Proceedings of the

Fifth International Joint Conference on

Autonomous Agents

and

Multiagent Systems

Future University-Hakodate, Japan
May 8-12, 2006

Edited by

Peter Stone, Gerhard Weiss

Table of Contents

INTRODUCTION..... **xxi**
INDUSTRY TRACK INTRODUCTION..... **xxiv**
CONFERENCE OFFICIALS..... **xxv**
SENIOR PROGRAM COMMITTEE..... **xxvii**
PROGRAM COMMITTEE..... **xxviii**
AUXILIARY REVIEWERS..... **xxxi**
INDUSTRY TRACK OFFICIALS..... **xxxiii**
SPONSORSHIP AND SUPPORT..... **xxxiv**

Invited Talks **1**

Towards Emergence of Communication: A Cognitive Developmental Robotics Approach 3
Minoru Asada
Making Strong Business Case for Multiagent Technology 10
Steve S. Benfield Jim Hendrickson, Daniel Galanti

SESSION: Simulation and modeling..... **17**

Full papers:
Agent Interface Enhancement: Making Multiagent Graphical Models Accessible..... 19
Yang Xiang, Kun Zhang
Power and Negotiation - Lessons from Agent-Based Participatory Simulations..... 27
Paul Guyot, Alexis Drogoul, Shinichi Honiden
RPD-Enabled Agents Teaming with Human for Multi-Context Decision Making..... 34
Xiaocong Fan, Bingjun Sun, Shuang Sun, Michael McNeese, John Yen
Camera-based Observation of Football Games for Analyzing Multi-agent Activities..... 42
Michael Beetz, Suat Gedikli, Bernhard Kirchlechner, Alexis Maldonado
Modeling one Human Decision-Maker with a Multi-Agent System: the CODAGE approach..... 50
Jean-Daniel Kant, Samuel Thiriot
Efficient Agent-Based Models for Non-Genomic Evolution..... 58
Nachi Gupta, Adrian Agogino, Kagan Tumer
Cognitive Agents Based Simulation for Decisions Regarding Human Team Composition..... 65
Bruno Gonçalves, Sofia Esteves, António Lopes, Luís Botelho

Performance of Pheromone Model for Predicting Traffic Congestion……..……..... 73
Yasushi Ando, Osamu Masutani, Hirotoshi Iwasaki, Yoshiaki Fukazawa, Shinichi Honiden
Modeling Agents and Interactions in Agricultural Economics……..... 81
Daisuke Torii, Toru Ishida, Francois Bousquet
Learnable Behavioural Model for Autonomous Virtual Agents: Low-Level Learning……..... 89
Toni Conde, Daniel Thalmann
A Flexible Framework for Representing Personality in Agents……..... 97
André Campos, Emanuel Santos, Anne Canuto, Rodrigo Soares, João Alchieri

Short papers:

Emotion-based Norm Enforcement and Maintenance in Multi-Agent Systems: Foundations and Petri Net Modeling……..... 105
Julia Fix, Christian von Scheve, Daniel Moldt
An Agent-Based Simulation of Pedestrian Dynamics: from Lane Formation to Auditorium Evacuation..... 108
Marcelo Toyama, Ana Bazzan, Roberto da Silva
Modeling Uncertain Domains with Polyagents……..... 111
Van Parunak, Sven Brueckner
TEAKS: Simulation of Human Performance at Work to Support Team Configuration……..... 114
Juan Martínez-Miranda, Arantza Aldea, Rene Bañares-Alcántara, Matias Alvarado
Modelling Dynamics of Cognitive Agents by Higher-Order Potentialities……..... 117
Tibor Bosse, Jan Treur
Approaches for Resolving the Dilemma between Model Structure Refinement and Parameter Calibration in Agent-Based Simulations……..... 120
Manuel Fehler, Franziska Klugl, Frank Puppe
Multiagent Simulation Meets the Real World……..... 123
Toru Ishida
Congestion Tolls as Utility Alignment Between Agent and System Optimum……..... 126
Ana Bazzan, Robert Junges
Agent-based modeling of human education data……..... 129
Yuqing Tang, Simon Parsons, Elizabeth Sklar
Study of Social Consciousness in Stochastic Agent-Based Simulations: Application to Supply Chains..... 132
Thierry Moyaux, Brahim Chaib-draa, Sophie D'Amours

SESSION: Logics for agent systems 135

Full papers:

Complexity and Succinctness of Public Announcement Logic……..... 137
Carsten Lutz
A complete and decidable security-specialised logic and its application to the TESLA protocol…….. 145
Alessio Lomuscio, Bożena Woźna
On the Logic of Coalitional Games……..... 153
Thomas Ågotnes, Wiebe van der Hoek, Michael Wooldridge
Model checking knowledge, strategies, and games in multi-agent systems……..... 161
Franco Raimondi, Alessio Lomuscio
Model Checking for Multivalued Logic of Knowledge and Time……..... 169
Wojciech Penczek, Beata Konikowska
Temporal Qualitative Coalitional Games……..... 177
Thomas Ågotnes, Wiebe van der Hoek, Michael Wooldridge
Reasoning about Action and Cooperation……..... 185
Luigi Sauro, Jelle Gerbrandy, Wiebe van der Hoek, Michael Wooldridge
Logics for Propositional Control……..... 193
Jelle Gerbrandy
On the Complexity of Practical ATL Model Checking……..... 201
Wiebe van der Hoek, Alessio Lomuscio, Michael Wooldridge
Knowing How to Play: Uniform Choices in Logics of Agency……..... 209
Andreas Herzig, Nicolas Troquard

Short paper:

Verifying space and time requirements for resource-bounded agents……..... 217
Natasha Alechina, Piergiorgio Bertoli, Chiara Ghidini, Mark Jago, Brian Logan, Luciano Serafini
The Interpreted System Model of Knowledge, Belief, Desire and Intention……..... 220
Kaile Su, Xiangyu Luo, Abdul Sattar, Mehmet Orgun
Operationalisation of Norms for Usage in Electronic Institutions……..... 223
Huib Aldewereld, Andrés García-Camino, Frank Dignum, Pablo Noriega, Juan Antonio Rodríguez-Aguilar, Carles Sierra
Mutual Enrichment Through Nested Belief Change……..... 226
Laurent Perrussel, Jean-Marc Thévenin, Thomas Meyer
An Architecture of a Normative System……..... 229
Guido Boella, Leendert van der Torre
What Agents Can Achieve under Incomplete Information……..... 232
Wojtek Jamroga, Thomas Ågotnes
Flexible agent protocols via temporal and resource-based reasoning……..... 235
Duc Pham, James Harland

Run-Time Model Checking of Interaction and Deontic Models for Multi-Agent Systems……..... 238
Nardine Osman, Dave Robertson, Christopher Walton
Modeling mental states in the analysis of multiagent systems requirements……..... 241
Alexei Lapouchnian, Yves Lespérance
A Stochastic Language for Modeling Opponent Agents……..... 244
Gerardo Simari, Amy Sliva, Dana Nau, Venkatramanan Subrahmanian

SESSION: Agent planning and search 247

Full papers:

Decentralized planning under uncertainty for teams of communicating agents……..... 249
Matthijs Spaan, Geoffrey J. Gordon, Nikos Vlassis
Integrating Parallel Interactions into Cooperative Search……..... 257
Efrat Manisterski, David Sarne, Sarit Kraus
A Multi-Agent System that Facilitates Scientific Publications Search……..... 265
Aliaksandr Birukou, Enrico Blanzieri, Paolo Giorgini
Security in multiagent systems by policy randomization……..... 273
Praveen Paruchuri, Milind Tambe, Fernando Ordóñez, Sarit Kraus
Real-Time Adaptive A*……..... 281
Sven Koenig, Maxim Likhachev
Winning back the cup for distributed POMDPs: Planning over continuous belief spaces……..... 289
Pradeep Varakantham, Ranjit Nair, Milind Tambe, Makoto Yokoo
Resource Allocation Among Agents with Preferences Induced by Factored MDPs……..... 297
Dmitri Dolgov, Edmund Durfee
Multi-Agent Based Peer-to-Peer Information Retrieval Systems With Concurrent Search Sessions……..... 305
Haizheng Zhang, Victor Lesser
A Hierarchical Approach to Efficient Reinforcement Learning in Deterministic Domains……..... 313
Carlos Diuk, Michael Littman, Alexander Strehl

Short papers:

A Smart Home Agent for plan recognition……..... 320
Bruno Bouchard, Abdenour Bouzouane, Sylvain Giroux
Exploring Sensor Networks using Mobile Agents……..... 323
Daniel Massaguer, Chien-Liang Fok, Nalini Venkatasubramanian, Gruia-Catalin Roman, Chenyang Lu
Stochastic Planning for Weakly-Coupled Distributed Agents……..... 326
AnYuan Guo, Victor Lesser

Dynamic Cooperation Using Resource Based Planning……..... 329
Utku Erdoğdu, Faruk Polat
Real-Time Target Evaluation Search……..... 332
Cagatay Undeger, Faruk Polat
Organizational Self-Design in Semi-dynamic Environments……..... 335
Sachin Kamboj, Keith Decker
A Decision-Theoretic Approach to File Consistency in Constrained Peer-to-Peer Device Networks……..338
David Roberts, Sooraj Bhat, Charles Isbell, Brian Cooper, Jeffrey S. Pierce
Solving POMDPs Using Quadratically Constrained Linear Programs……..... 341
Christopher Amato, Daniel S. Bernstein, Shlomo Zilberstein

SESSION: Argumentation and negotiation 345

Full papers:

An Argumentation-based Approach for Practical Reasoning……..... 347
Iyad Rahwan, Leila Amgoud
Predicting partner's behaviour in agent negotiation……..... 355
Jakub Brzostowski, Ryszard Kowalczyk
Tractable Negotiation in Tree-structured Domains……..... 362
Yann Chevaleyre, Ulle Endriss, Nicolas Maudet
Predicting People's Bidding Behavior in Negotiation……..... 370
Ya'akov Gal, Avi Pfeffer
Support-based Distributed Search……..... 377
Peter Harvey, Chee Fon Chang, Aditya Ghose
Adaptive Agent Negotiation via Argumentation……..... 384
Antonis Kakas, Pavlos Moraitis
Monotonic Concession Protocols for Multilateral Negotiation……..... 392
Ulle Endriss
Negotiating using Rewards……..... 400
Sarvapali Ramchurn, Carles Sierra, Lluis Godo, Nick Jennings

Short papers:

Negotiation Coalitions in Group-Choice Multi-Agent Systems……..... 408
Tom Wanyama, Behrouz Far
Negotiation-based coalitions in the physical world……..... 411
Majid Ali Khan, Ladislau Bölöni
A formal model for designing dialogue strategies……..... 414

Leila Amgoud, Nabil Hameurlain

A Scalable Method for Online Learning of Non-linear Preferences Based on Anonymous Negotiation Data 417

D.J.A. Somefun, J.A. La Poutré

Towards Formal Model of Adversarial Action in Multi-Agent Systems……..... 420

Michal Pěchouček, Jan Tožička, Martin Rehák

Negotiating Efficient Outcomes over Multiple Issues……..... 423

Sabyasachi Saha, Sandip Sen

Managing Social Influences through Argumentation-Based Negotiation……..... 426

Nishan Karunatillake, Nick Jennings, Iyad Rahwan, Sarvapali Ramchurn

An expressive approach to fuzzy constraint based agent purchase negotiation……..... 429

Miguel López-Carmona, Juan R. Velasco

Symbolic Negotiation Revisited……..... 432

Peep Küngas, Mihhail Matskin

A Multi-Issue Negotiation Protocol among Competitive Agents and its Extension to A Nonlinear Utility Negotiation Protocol……..... 435

Takayuki Ito, Mark Klein

SESSION: Robotics..... 439

Full papers:

Multi-Robot Learning with Particle Swarm Optimization……..... 441

Jim Pugh, Alcherio Martinoli

Multi-Model Motion Tracking under Multiple Team Member Actuators……..... 449

Gu Yang, Manuela Veloso

Multi-Agent Strategic Modeling in a Robotic Soccer Domain……..... 457

Andraz Bezek, Matjaz Gams, Ivan Bratko

On the Response of EMT-based Control to Interacting Targets and Models……..... 465

Zinovi Rabinovich, Jeffrey Rosenschein

Multiple UAV Task Allocation using Negotiation……..... 471

P.B. Sujit, Arpita Sinha, Debasish Ghose

Short papers:

Distributed Automatic Target Recognition Using Multi-Agent UAV Swarms……..... 479

Prithviraj Dasgupta

Action Awareness - Enabling Agents to Optimize, Transform, and Coordinate Plans……..... 482

Freek Stulp, Michael Beetz

Programming Modular Robots with the TOTA Middleware……..... 485

Marco Mamei, Franco Zambonelli

RADIC - A Generic Component for the Integration of Existing Reactive and Deliberative Layers..... 488

Matthias Scheutz, James Kramer

A Constraint Optimization Framework for Fractured Robot Teams……..... 491

Mary Koes, Katia Sycara, Illah Nourbakhsh

SESSION: Computational complexity in agent systems 495

Full papers:

Junta Distributions and the Average-Case Complexity of Manipulating Elections……..... 497

Ariel Procaccia, Jeffrey Rosenschein

The Communication Complexity of Coalition Formation Among Autonomous Agents……..... 505

Ariel Procaccia, Jeffrey Rosenschein

The Cost of Social Agents……..... 513

Guido Governatori, Antonino Rotolo, Vineet Padmanabhan

Failures of the VCG Mechanism in Combinatorial Auctions and Exchanges……..... 521

Vincent Conitzer, Tuomas Sandholm

Agent Interaction in Distributed MDPs and its Implications on Complexity……..... 529

Jiaying Shen, Raphen Becker, Victor Lesser

A Technique for Reducing Normal Form Games to Compute a Nash Equilibrium……..... 537

Vincent Conitzer, Tuomas Sandholm

Short papers:

Information-theoretic approaches to branching in search……..... 545

Andrew Gilpin, Tuomas Sandholm

The complexity of model checking concurrent programs against CTLK specifications……..... 548

Alessio Lomuscio, Franco Raimondi

SESSION: Cooperation and coordination 551

Full papers:

Distributed Navigation in an Unknown Physical Environment……..... 553

Arnon Gilboa, Ariel Felner, Amnon Meisels

Accident or Intention: That Is the Question (in the Noisy Iterated Prisoner's Dilemma)……..... 561

Tsz-Chiu Au, Dana Nau

On using Multi-agent Systems in Playing Board Games……..... 569

Stefan Johansson

Solution Sets for DCOPs and Graphical Games……..... 577
Jonathan Pearce, Rajiv Maheswaran, Milind Tambe
Towards Autonomous Mobile Agents with Emergent Migration Behaviour……..... 585
Tino Schlegel, Peter Braun, Ryszard Kowalczyk
Artifacts for Time-Aware Agents……..... 593
Cédric Dinont, Emmanuel Druon, Philippe Mathieu, Patrick Taillibert
Total Performance by Local Agent Selection Strategies in Multi-Agent Systems……..... 601
Toshiharu Sugawara, Satoshi Kurihara, Toshio Hirotsu, Kensuke Fukuda, Shinya Sato, Osamu Akashi
Effect of joining decisions on peer clusters……..... 609
Stéphane Airiau, Sandip Sen, Prithviraj Dasgupta
History-based Traffic Control……..... 616
Gabriel Balan, Sean Luke
Communication Management Using Abstraction in Distributed Bayesian Networks……..... 622
Jiaying Shen, Victor Lesser
Toward an Understanding of the Impact of SoftwarePersonal Assistants on Human Organizations……..... 630
Steven Okamoto, Paul Scerri, Katia Sycara
Robust Recognition of Physical Team Behaviors Using Spatio-temporal Models……..... 638
Gita Sukthankar, Katia Sycara
How Autonomy Oriented Computing (AOC) Tackles a Computationally Hard Optimization Problem…… 646
Xiao-Feng Xie, Jiming Liu
Social Coordination without Communication in Multi-Agent Territory Exploration Tasks……..... 654
Paul Schermerhorn, Matthias Scheutz
Analyzing characteristics of task structures to develop GPGP coordination mechanisms……..... 662
Wei Chen, Keith Decker

Short papers:

Norm-Oriented Programming of Electronic Institutions……..... 670
Andres García-Camino, Juan Antonio Rodríguez-Aguilar, Carles Sierra, Wamberto Vasconcelos
Learning to Identify Winning Coalitions in the PAC Model……..... 673
Ariel Procaccia, Jeffrey Rosenschein
A Deployed Multi-Agent Framework for Distributed Energy Applications……..... 676
Geoff James, Dave Cohen, Robert Dodier, Glenn Platt, Doug Palmer
Behaviosites: A Novel Paradigm for Affecting Distributed Behavior……..... 679
Amit Shabtay, Zinovi Rabinovich, Jeffrey Rosenschein
How can agent know the global information without close coordination?..... 682
Satoshi Kurihara, Kensuke Fukuda, Shinya Sato, Toshiharu Sugawara
Strategic voting when aggregating partially ordered preferences……..... 685

Francesca Rossi, Maria Silvia Pini, Kristen Brent Venable, Toby Walsh

Role-Based Semantics for Agent Communication……..... 688

Guido Boella, Rossana Damiano, Joris Hulstijn, Leendert van der Torre

Teaching New Teammates……..... 691

Doran Chakraborty, Sandip Sen

SESSION: Learning and evolution 695

Full papers:

Efficient agents for Cliff Edge environments with a large set of decision options……..... 697

Ron Katz, Sarit Kraus

A Novel Method for Automatic Strategy Acquisition in N-player Non-zero-sum Games……..... 705

Steve Phelps, Marek Marcinkiewicz, Simon Parsons, Peter McBurney

Robust Game Play Against Unknown Opponents……..... 713

Nathan Sturtevant, Michael Bowling

Probabilistic Policy Reuse in a Reinforcement Learning Agent……..... 720

Fernando Fernández, Manuela Veloso

Learning from Induced Changes in Opponent (Re)Actions in Multi-Agent Games……..... 728

Pieter Jan't Hoen, Sander Bohte, Han La Poutré

Learning the Required Number of Agents for Complex Tasks……..... 736

Sébastien Paquet, Brahim Chaib-draa

Extensive Games with Possibly Unaware Players……..... 744

Joseph Halpern, Leandro Rêgo

Learning against multiple opponents……..... 752

Thuc Vu, Rob Powers, Yoav Shoham

Selecting Informative Actions Improves Cooperative Multiagent Learning……..... 760

Liviu Panait, Sean Luke

The Use of Agents in Human Learning Systems……..... 767

Elizabeth Sklar, Debbie Richards

Reinforcement Learning for Declarative Optimization-Based Drama Management……..... 775

Mark Nelson, David Roberts, Charles Isbell, Michael Mateas

Short papers:

Learning to Cooperate in Multi-Agent Social Dilemmas……..... 783

Enrique Munoz de Cote, Andrea Bonarini, Alessandro Lazaric, Marcello Restelli

An Adaptive Plan-Based Dialogue Agent: Integrating Learning into a BDI Architecture…….. 786

Anh Nguyen, Wayne Wobcke

Multi-agent Reinforcement Learning Algorithm to Handle Beliefs of Other Agents' Policies and Embedded Beliefs……..... 789
Takaki Makino, Kazuyuki Aihara
Rule Value Reinforcement Learning for Cognitive Agents…..... 792
Chris Child, Kostas Stathis
Learning Executable Agent Behaviors from Observation……..... 795
Andrew Guillory, Hai Nguyen, Tucker Balch, Charles Isbell
RV $_{\sigma(t)}$: A unifying approach to performance and convergence in online multiagent learning……..... 798
Bikramjit Banerjee, Jing Peng
Lenient Learners in Cooperative Multiagent Systems……..... 801
Liviu Panait, Keith Sullivan, Sean Luke
Can Good Learners Always Compensate for Poor Learners?..... 804
Keith Sullivan, Liviu Panait, Gabriel Balan, Sean Luke
Learning Regularities with a Constructivist Agent……..... 807
Filipo Perotto, Luís Otávio Álvares
Improving Reinforcement Learning with Context Detection……..... 810
Bruno C. da Silva, Eduardo W. Basso, Filipo S. Perotto, Ana L.C. Bazzan, Paulo M. Engel
Software agents that learn through observation……..... 813
Jaçanã Machado, Luís Botelho

SESSION: Task and resource allocation 817

Full papers:

Resource Selection Games with Unknown Number of Players……..... 819
Itai Ashlagi, Dov Monderer, Moshe Tennenholtz
Adaptive Distributed Resource Allocation and Diagnostics Using Cooperative Information-Sharing Strategies……..... 826
Partha Dutta, Nick Jennings, Luc Moreau
Forming Efficient Agent Groups for Completing Complex Tasks……..... 834
Efrat Manisterski, Esther David, Sarit Kraus, Nick Jennings
Gradient Field-Based Task Assignment in an AGV Transportation System……..... 842
Danny Weyns, Nelis Boucké, Tom Holvoet
Learning the Task Allocation Game……..... 850
Sherief Abdallah, Victor Lesser
Continuous Refinement of Agent Resource Estimates……..... 858
David Morley, Karen Myers, Neil Yorke-Smith
How Equitable Is Rational Negotiation?..... 866

Sylvia Estivie, Yann Chevaleyre, Ulle Endriss, Nicolas Maudet

Mathematical Programming for Deliberation Scheduling in Time-limited Domains……..... 874

Jianhui Wu, Edmund Durfee

Optimal Decision-Making With Minimal Waste: Strategyproof Redistribution of VCG Payments……..... 882

Ruggiero Cavallo

Short papers:

Distributed Lagrangean Relaxation Protocol for the Generalized Mutual Assignment Problem……..... 890

Katsutoshi Hirayama

Auctioning Transformable Goods……..... 893

Andrea Giovannucci, Juan Antonio Rodríguez-Aguilar, Jesús Cerquides

SESSION: Ontologies and web services 897

Full papers:

ANEMONE: An effective minimal ontology negotiation environment……..... 899

Jurriaan van Diggelen, Robbert-Jan Beun, Frank Dignum, Rogier van Eijk, John-Jules Meyer

Agents for e-Business Applications……..... 907

Alessandro Negri, Agostino Poggi, Michele Tomaiuolo, Paola Turci

Automated Semantic Web Service Discovery with OWLS-MX……..... 915

Matthias Klusch, Benedikt Fries, Mahboob Alam Khalid, Katia Sycara

Ontology-Guided Learning to Improve Communication between Groups of Agents……..... 923

Mohsen Afsharchi, Behrouz Far, Jörg Denzinger

A Context-Aware Approach for Service Selection Using Ontologies……..... 931

Murat Şensoy, Pinar Yolum

Short papers:

TANDEM - a Design Method for Integrating Web Services into Multi-Agent Systems……..... 939

Steve Goschnick, Sandrine Balbo, Leon Sterling, Christine Sun

Reconciling Ontological Differences by Assistant Agents……..... 943

Kendall Lister, Leon Sterling, Kuldar Taveter

Formalizing Typicality of Objects and Context-sensitivity in Ontologies ……..... 946

Ching-Man Au Yeung, Ho-fung Leung

Service-Oriented and Autonomous Distribution and Provision of Multimedia Contents ……..... 949

Fuyuki Ishikawa, Nobukazu Yoshioka, Shinichi Honiden

Deriving Axioms Across Ontologies……..... 952

Dejing Dou, Drew McDermott

Web Presentation System using RST Events……..... 955
Arturo Nakasone, Helmut Prendinger, Mitsuru Ishizuku

SESSION: Believable agents 959

Full papers:

Learning Empathy: A Data-Driven Framework for Modeling Empathetic Companion Agents……. 961
Scott McQuiggan, James Lester

Embodied mobile agents……..... 969
Bill Tomlinson, Man Lok Yau, Eric Baumer

U-DIRECTOR: A Decision-Theoretic Narrative Planning Architecture for Storytelling Environments..... 977
Bradford Mott, James Lester

Evaluating a Computational Model of Social Causality and Responsibility……..... 985
Wenji Mao, Jon Gratch

Short papers:

A Model of Emotions for Situated Agents……..... 993
Van Parunak, Robert Bisson, Sven Brueckner, Robert Matthews, John Sauter

Balancing the needs of players and spectators in agent-based commentary systems……..... 996
Daniel Fielding, Brian Logan, Steve Benford

SESSION: Architectures: BDI and MDPs 999

Full papers:

Hierarchical Planning in BDI Agent Programming Languages: A Formal Approach……..... 1001
Sebastian Sardina, Lavindra de Silva, Lin Padgham

Modular BDI Architecture…….…….…….…..... 1009
Peter Novák, Jürgen Dix

Awareness as a vital ingredient of teamwork……..... 1017
Barbara Dunin-Kęplicz, Rineke Verbrugge

Exact Solutions of Interactive POMDPs Using Behavioral Equivalence……..... 1025
Bharanee Rathnasabapathy, Prashant Doshi, Piotr Gmytrasiewicz

On Proactivity and Maintenance Goals…….....1033
Simon Duff, James Harland, John Thangarajah

On the relationship between MDPs and the BDI architecture……..... 1041
Gerardo Simari, Simon Parsons

Short papers:

Modelling Situations in Intelligent Agents……..... 1049

John Thangarajah, Lin Padgham, Sebastian Sardina

Beliefs, Desires and Intentions Through the Environment……..... 1052

Tom Holvoet, Paul Valckenaers

User Defined Monitoring Strategies for BDI Agent Programs……..... 1055

Timothy William Cleaver, Abdul Sattar, Raihana Ferdous

Mixed-integer Linear Programming for Transition-independent Decentralized MDPs……..... 1058

Jianhui Wu, Edmund Durfee

SESSION: Scalability, security, and performance analysis 1061

Full papers:

Performance Analysis of Untraceability Protocols for Mobile Agents Using an Adaptable Framework……. 1063

Rafał Leszczyna, Janusz Górski

Scalable and Reliable Data Delivery in Mobile Ad Hoc Sensor Networks……..... 1071

Bin Yu, Paul Scerri, Katia Sycara, Yang Xu, Michael Lewis

Efficient Agent-Based Cluster Ensembles……..... 1079

Adrian Agogino, Kagan Tumer

ReAgents: Behavior-based Remote Agents and Their Performance……..... 1087

Eugene Hung, Joseph Pasquale

Short papers:

MLBP: MAS for Large-Scale Biometric Pattern Recognition……..... 1095

Ram Meshulam, Shulamit Reches, Aner Yarden, Sarit Kraus

Using Queuing Theory to Predict Organizational Metrics……..... 1098

Bryan Horling, Victor Lesser

Agent Dependability as an Architectural Issue……..... 1101

Peter C. Lockemann, Jens Nimis

Quorum Sensing on Mobile Ad-Hoc Networks**……**..... 1104

Maxim Peysakhov, Chris Dugan, Jay Modi, William Regli

Performance Evaluation of Open-Source Multiagent Platforms……..... 1107

Luis Mulet, Jose Miguel Such, Juan Miguel Alberola, Vicent Botti, Agustin Espinosa, Ana Garcia, Andres Terrasa

Structural Evaluation of Agent Organizations……..... 1110

Davide Grossi, Frank Dignum, Virginia Dignum, Mehdi Dastani, Lambèr Royakkers

Comparing Market and Token-Based Coordination……..... 1113

Yang Xu, Paul Scerri, Katia Sycara, Michael Lewis

SAgent: A Security Framework for JADE……..... 1116

Vandana Gunupudi, Stephen Tate

Coordinating simple and unreliable agents……..... 1119

Kagan Tumer

The Emergence of Global Properties from Local Interactions……..... 1122

Daniel Yamins

SESSION: Auctions and electronic markets 1125

Full papers:

Sequences of Take-It-or-Leave-It Offers: Near-Optimal Auctions Without Full Valuation Revelation……1127

Tuomas Sandholm, Andrew Gilpin

False-name-proof Combinatorial Auction Protocol: Groves Mechanism with Submodular Approximation…….1135

Makoto Yokoo, Toshihiro Matsutani, Atsushi Iwasaki

Reducing Price Fluctuation in Continuous Double Auctions through Pricing Policy and Shout Improvement….1143

Jinzhong Niu, Simon Parsons, Elizabeth Sklar, Kai Cai

Instantiating the Contingent Bids Model of Truthful Interdependent Value Auctions……..... 1151

Takayuki Ito, David Parkes

Designing a Successful Trading Agent for Supply Chain Management……..... 1159

Minghua He, Alex Rogers, Xudong Luo, Nicholas R. Jennings

Reducing Costly Information Acquisition in Auctions……..... 1167

Kate Larson

Computationally Efficient and Revenue Optimized Auctioneer's Strategy for Expanding Auctions…… 1175

Onn Shehory, Eran Dror

A Robust Combinatorial Auction Mechanism against Shill Bidders……..... 1183

Tokuro Matsuo, Takayuki Ito, Robert Day, Toramatsu Shintani

Mertacor: A Successful Autonomous Trading Agent……..... 1191

Panos Toulis, Dionisis Kehagias, Pericles Mitkas

Short papers:

A Comparison between mechanisms for Sequential Compute Resource Auctions……..... 1199

Andrew Byde

Robust Mechanisms for Information Elicitation……..... 1202

Aviv Zohar, Jeffrey Rosenschein

A Comparative Study of Sequential and Simultaneous Auctions……..... 1205

Shaheen Fatima

Competing Sellers in Online Markets: Reserve Prices, Shill Bidding, and Auction Fees……..... 1208
Enrico Gerding, Alex Rogers, Rajdeep Dash, Nick Jennings
Evaluating Bidding Strategies for Simultaneous Auctions……..... 1211
Teddy Candale, Sandip Sen

SESSION: Trust and reputation 1215

Full papers:
Certified Reputation: How an Agent Can Trust a Stranger……..... 1217
Trung Dong Huynh, Nick Jennings, Nigel R. Shadbolt
Trust and Honour in Information-based Agency……..... 1225
Carles Sierra, John Debenham
Using CHI-Scores to Reward Honest Feedback from Repeated Interactions……..... 1233
Radu Jurca, Boi Faltings
Learning Trust Strategies in Reputation Exchange Networks……..... 1241
Karen Fullam, K. Suzanne Barber
Monopolising Markets by Exploiting Trust……..... 1249
Sandip Sen, Dipyaman Banerjee

Short papers:
Developing a Composite Trust Model for Multi-agent Systems……..... 1257
Xiaoqing Zheng, Zhaohui Wu, Huajun Cheng, Yuxin Mao
Trust-based Agent Community for Collaborative Recommendation……..... 1260
Jianshu Weng, Chunyan Miao, Angela Goh, Zhiqi Shen, Robert Gay
Learning to commit in repeated games……..... 1263
Stéphane Airiau, Sandip Sen
Effect of deceptive referrals on system stability……..... 1266
Ikpeme Erete, Teddy Candale, Sandip Sen

SESSION: Agent programming 1269

Full papers:
Goal-Oriented Modularity in Agent Programming……..... 1271
M. Birna van Riemsdijk, Mehdi Dastani, John-Jules Meyer, Frank de Boer

Short papers:
Prototyping 3APL in the Maude Term Rewriting Language……..... 1279

M. Birna van Riemsdijk, Frank de Boer, Mehdi Dastani, John-Jules Meyer

A Jabber-based Multi-Agent System Platform……..... 1282

Miguel Escrivá Gregori, Javier Palanca Cámara, Gustavo Aranda Bada, Ana Garcia, Vicente Julian, Vicent Botti

Goal Types in Agent Programming……..... 1285

Mehdi Dastani, M. Birna van Riemsdijk, John-Jules Meyer

Belief Revision for AgentSpeak Agents……..... 1288

Natasha Alechina, Rafael Bordini, Jomi Fred Hübner, Mark Jago, Brian Logan

Plan Patterns for Declarative Goals in AgentSpeak……..... 1291

Jomi Fred Hübner, Rafael Bordini, Michael Wooldridge

A Service-Oriented Language for Programming Mobile Agents……..... 1294

Hervé Paulino, Luís Lopes

ActorNet: An Actor Platform for Wireless Sensor Networks……..... 1297

YoungMin Kwon, Sundresh Sameer, Kirill Mechitov, Gul Agha

SESSION: Agent-oriented engineering and methodologies 1301

Full papers:

Integrating Agent-Oriented Methodologies with UML-AT……..... 1303

Rubén Fuentes-Fernandez, Jorge Gómez-Sanz, Juan Pavón

Short papers:

Designing Agent Chips……..... 1311

Insu Song, Guido Governatori

Multiagent Systems as Software Architecture……..... 1314

Danny Weyns, Tom Holvoet, Kurt Schelfthout

Automated Analysis and Verification of Agent Behavior……..... 1317

Tibor Bosse, Dung Lam, K. Suzanne Barber

The hidden dangers of experimenting in distributed AI……..... 1320

Annika Smit, Masja Kempen, André Meyer, Niek Wijngaards

Agent-Community based Peer-to-Peer Information Retrieval - an Evaluation……..... 1323

Tsunenori Mine, Akihiro Kogo, Makoto Amamiya

An Overview of Business Process Adaptations via Protocols……..... 1326

Nirmit Desai, Amit Chopra, Munindar Singh

Multi-agent based Community Computing System Development with the Model Driven Architecture…….. 1329

Yuna Jung, Jungtae Lee, Minkoo Kim

Enhancing Agent-Oriented Models with Aspects……..... 1332

Alessandro Garcia, Christina Chavez, Ricardo Choren

SESSION: Agent communication 1335

Full papers:

On the Semantics of Conditional Commitment……..... 1337

Shakil Khan, Yves Lespérance

Contextualizing Commitment Protocols……..... 1345

Amit Chopra, Munindar Singh

A Utility based sensing and communication model for a glacial sensor network……..... 1353

Paritosh Padhy, Rajdeep Dash, Kirk Martinez, Nick Jennings

Specifying and Resolving Preferences Among Agent Interaction Patterns……..... 1361

Ashok Mallya, Munindar Singh

Short papers:

An Ontology of Social Control Tools……..... 1369

Philippe Pasquier, Roberto Flores, Brahim Chaib-draa

Modelling the Links Between Social Commitments and Individual Intentions……..... 1372

Philippe Pasquier, Brahim Chaib-draa

The MAPa Language of Agent Dialogues……..... 1375

MaríaAdela Grando, Christopher Walton

Convergence Analysis for Collective Vocabulary Development……..... 1378

Jun Wang, Les Gasser, Jim Houk

Learning a Common Language Through an Emergent Interaction Topology……..... 1381

Samarth Swarup, Kiran Lakkaraju, Les Gasser

Organizational Petri Nets for Protocol Design and Enactment……..... 1384

Stéphanie Combettes, Chihab Hanachi, Christophe Sibertin-Blanc

SESSION: Optimization and constraint processing 1387

Full papers:

Nogood-based Asynchronous Distributed Optimization (ADOPT-ng)……..... 1389

Marius Silaghi, Makoto Yokoo

MDPOP: Faithful Distributed Implementation of Efficient Social Choice Problems……..... 1397

Adrian Petcu, Boi Faltings, David Parkes

Examining DCSP Coordination Tradeoffs……..... 1405

Michael Benisch, Norman Sadeh

Multiply-Constrained Distributed Constraint Optimization……..... 1413
Emma Bowring, Milind Tambe, Makoto Yokoo

Short papers:

The DynCOAA Algorithm for Dynamic Constraint Optimization Problems……..... 1421
Koenraad Mertens, Tom Holvoet, Yolande Berbers
Experimental analysis of privacy loss in DCOP algorithms……..... 1424
Rachel Greenstadt, Jonathan Pearce, Emma Bowring, Milind Tambe
No-Commitment Branch and Bound Search for Distributed Constraint Optimization……..... 1427
Anton Chechetka, Katia Sycara
An Agent Based General Solution Model For the Course Timetabling Problem……..... 1430
Yan Yang, Raman Paranjape, Luigi Benedicenti
Hierarchical Variable Ordering for Distributed Constraint Optimization……..... 1433
John Davin, Jay Modi

Demonstration Session..... 1437

SESSION 1

SeSAm: Implementation of Agent-Based Simulation Using Visual Programming..... 1439
F. Klugl, R. Herrler, M. Fehler
CSC: Criticality-Sensitive Coordination..... 1441
Pedro Szekely , Marcel Becker , Stephen Fitzpatrick , Gergely Gati , David Hanak , Jing Jin , Gabor Karsai , Rajiv T. Maheswaran , Bob Neches , Craig M. Rogers , Romeo Sanchez , Chris van Buskirk
Synergistic Integration of Agent Technologies for Military Simulation 1443
Sean Owens, Paul Scerri, Robin Glinton, Bin Yu, Katia Sycara
Asynchronous Chess Competition 1445
Nathaniel Gemelli, Robert Wright, James Lawton, Andrew Boes
Deployment of A-globe Multi-Agent Platform 1447
David Sislak, Martin Rehák, Michal Pěchouček, Dušan Pavlíček
UAV Airspace Management System UAMS 1449
Regis Vincent, Osher Yagdar, Andrew Agno
Multi-Agent Coordination and Control Testbed for planning and scheduling strategies 1451
Paul Verstraete, Paul Valckenaers, Hendrik Van Brussel, Karuna Hadeli, Bart Saint Germain
A Texas Hold'em poker player based on automated abstraction and real-time equilibrium computation 1453
Andrew Gilpin, Tuomas Sandholm

SESSION 2

Decision-Making Support for Maritime Business..... 1455
Jonathan Himoff

Participatory, Embodied, Multi-Agent Simulation..... 1457
Paulo Blikstein, William Rand, Uri Wilensky

CONOISE-G: Agent-Based Virtual Organisations..... 1459
Jigar Patel, W.T. Luke Teacy, Nicholas R. Jennings, Michael Luck, Stuar t Chalmers, Nir Oren, Timothy J. Norman, Alun Preece, Peter M. D. Gray, Gareth Shercliff, Patrick J. Stockreisser, Jianhua Shao, W. Alex Gray, Nick J. Fiddian, Simon Thompson

Automated Argument System based on Logic of Multiple-Valued Argumentation..... 1461
Toshiko Wakaki, Takanori Mukai, Hiromitsu Miyashita, Hajime Sawamura, Kumiko Matsunaga, Taro Fukumoto, Katsumi Nitta

Computational Mechanism Design for Multi-Sensor Information Fusion..... 1463
A. Rogers, R.K. Dash, N.R. Jennings, S. Reece, S. Roberts

Component Agent Framework for domain-Experts (CAFnE) Toolkit..... 1465
Gaya Jayatilleke John Thangarajah Lin Padgham Michael Winikoff

Conflict Negotiation Among Personal Calendar Agents..... 1467
Pauline M. Berry, Cory Albright, Emma Bowring, Ken Conley, Kenneth Nitz, Jonathan P. Pearce, Bart Peintner, Shahin Saadati, Milind Tambe, *Tomás Uribe, Neil Yorke-Smith*

Synthetic Humans in Emergency Response Drills..... 1469
Daniel Massaguer, Vidhya Balasubramanian, Sharad Mehrotra, Nalini Venkatasubramanian

ITSUMO : an Intelligent Transportation System for Urban Mobility..... 1471
Bruno Castro da Silva, Rober t Junges , Denise de Oliveira, Ana L.C. Bazzan

***Industry Track*..... 1473**

SESSION: IT1

Towards a Reference Model for Intelligent Agent Systems..... 1475
Pragnesh Jay Modi, Spiros Mancoridis, William M. Mongan, William Regli, Israel Mayk

Automating Deep Space Network Scheduling and Conflict Resolution 1483
Mark D. Johnston, Bradley J. Clement

Using Multiagent Teams to Improve the Training of Incident Commanders..... 1490
Nathan Schurr, Pratik Patil, Fred Pighin, Milind Tambe

Autonomous Agents for Air-Traffic Deconfliction..... 1498
Michal Pěchouček, David Sislak, , Dušan Pavlíček, Miroslav Uller

SESSION: IT2

Automating Supply Chain Negotiations using Autonomous Agents: a Case Study in Transportation Logistics.. 1506
Sander van der Putten, Valentin Robu, Han La Poutré, Annemiek Jorritsma, Margo Gal

Multi-Agent Logistics i-Scheduler for Road Transportation..... 1514
Jonathan Himoff, George Rzevski, Petr Skobelev

Commercializing a Multiagent-Supported Collaborative System..... 1522
Leen-Kiat Soh, Hong Jiang

SESSION: IT3

The Role of Agents in Enterprise System Management..... 1530
Onn Shehory

Implantable Medical Devices as Agents and Part of Multiagent Systems..... 1534
Touby Drew, Maria Gini

A Hotel Information System implementation using MAS technology……..... 1542
Armando Robles P., Pablo Noriega B.V., Marco Julio Robles P., Hector Hernandez T., Victor Soto Ramírez, Edgar Gutierrez S.

Commercial Applications of Agents: Lessons, Experiences and Challenges..... 1549
Roxana A. Belecheanu, Steve Munroe, Michael Luck, Tim Miller, Peter McBurney

SESSION: IT4

A Computer-Supported Cooperative Learning System with Multiagent Intelligence..... 1556
Leen-Kiat Soh, Nobel Khandaker, Xuliu Liu, Hong Jiang

Deploying a Personalized Time Management Agent..... 1564
Pauline Berry, Ken Conley, Melinda Gervasio, Bart Peintner, Tomás Uribe, Neil Yorke-Smith

Utility-based Collaboration among Autonomous Agents for Resource Allocation in Data Centers..... 1572
Rajarshi Das, Ian N. Whalley, Jeffrey O. Kephart

ACTIVE: Agile Coordinator Testbed Integrated Virtual Environment..... 1580
G. Emami, J. Cheng, D. Cornwell, M. Feldhousen, C. Long, V. Malhotra, I. Starnes, L. Kerschberg, A. Brodsky, X. Zhang

***Author Index*..... 1588**

INTRODUCTION

The spread of Internet connectivity and advancement of computer hardware increases the power of information technologies to perform practical services in many areas of our everyday life. Agent technologies play essential roles in the design of service-oriented systems such as computer-mediated collaboration, education and training, electronic commerce, information retrieval, and data mining.

Agents and multiagent systems also provide a new way of thinking--conceptualizing performance elements as autonomous changes the paradigm for analyzing problems, designing systems, and dealing with complexity, distribution, and interactivity. Concept of agents and multiagent architecture are shaping landscapes of basic research areas including social simulation, games, pervasive and ubiquitous computing, robotics, and user interfaces.

The Autonomous Agents and Multiagent Systems (AAMAS) conference series brings together researchers from around the world to share the latest advances in the field. It was initiated in 2002 as a merger of three highly successful related events: the International Conference on Autonomous Agents (AGENTS), the International Conference on Multi-Agent Systems (ICMAS), and the International Workshop on Agent Theories, Architectures, and Languages (ATAL). The AAMAS conference series provides a single, high-profile forum for research in the theory and practice of autonomous agents and multiagent systems. AAMAS 2002, the first of the series, was held in Bologna, followed by AAMAS 2003 in Melbourne, AAMAS 2004 in New York, and AAMAS 2005 in Utrecht. AAMAS 2006 is held in Hakodate, Japan.

The five-day event consists of 6 tutorials, 32 workshops, a main conference track, a separate industry paper track with 15 papers, demonstration sessions with 17 demonstrations, an exhibition, and a doctoral mentoring program with 24 students. These and other conference elements are the result of great efforts on the part of many dedicated volunteers. We are grateful to all the AAMAS-06 conference officials and those who worked with them to put together such a successful research meeting.

AAMAS 2006 received 553 submissions for the main conference. Each paper was reviewed by at least three program committee members (guided by a senior program committee member). 127 papers were accepted as full 8-page papers, and an additional 123 were accepted as short 3-page papers (acceptance rates of 23.0% for full papers and 45.2% overall).

Several innovations were added to this year's conference program and review process. In order to make the review process more interactive, authors were given the opportunity to respond to their reviews prior to the discussion phase and final decisions. In many cases, the reviews, and sometimes the decisions, were altered as a result. In addition, we introduced awards for "best program committee member" and "best senior program committee member" in recognition of the exceptional time and effort contributed by our program committee.

At the conference itself, for the first time, the emphasis was on poster presentations as the main form of technical exchange. All accepted papers were presented at poster sessions, with just about half of the full papers presented orally in two parallel tracks. The oral presentations were chosen primarily so as to provide a broad representative of the accepted papers. The goal of this format was to allow for more focused oral sessions while maximizing the opportunity for one-on-one detailed technical exchanges at the posters.

From among this year's complete program, the following papers were selected as nominees for the best paper and best student paper awards:

Resource Selection Games with Unknown Number of Players
Itai Ashlagi, Dov Monderer, Moshe Tennenholtz

Optimal Decision-Making With Minimal Waste: Strategyproof Redistribution of VCG Payments
Ruggiero Cavallo

A Technique for Reducing Normal Form Games to Compute a Nash Equilibrium
Vincent Conitzer, Tuomas Sandholm

Instantiating the Contingent Bids Model of Truthful Interdependent Value Auctions
Takayuki Ito, David Parkes

Automated Semantic Web Service Discovery with OWLS-MX
Matthias Klusch, Benedikt Fries, Mahboob Alam Khalid, Katia Sycara

Learning Empathy: A Data-Driven Framework for Modeling Empathetic Companion Agents
Scott McQuiggan, James Lester

Toward an Understanding of the Impact of Software Personal Assistants on Human Organizations
Steven Okamoto, Paul Scerri, Katia Sycara

A complete and decidable security-specialised logic and its application to the TESLA protocol
Bozena Wozna, Alessio Lomuscio

The program chairs would like to thank all of the PC and SPC members for their hard work putting together this year's program. Overall, the reviews were quite detailed and the reviewers were very responsive in the discussion phase, often carefully considering the author responses and altering reviews accordingly. From among the group, a few PC members stand out as having written particularly detailed reviews and providing constructive, helpful feedback both to the authors and to the program chairs. These individuals have been nominated for the best PC member award:

Gal Kaminka
Barbara Keplicz
David Parkes
Simon Parsons
Jeff Rosenschein

Additionally, a few SPC members stand out as leading particularly useful discussions and providing useful input back to the authors and the program chairs. These individuals have been nominated for the best SPC member award:

Vincent Conitzer
Luke Hunsberger
Shaul Markovitch
Karl Tuyls
Danny Weyns

Congratulations (and thank you!) to all of the nominees.

Hideyuki Nakashima and Michael Wellman (general chairs)
Peter Stone and Gerhard Weiss (program chairs)

March 2006

INDUSTRY TRACK INTRODUCTION

Agents and multi-agent systems have a role to play in commercial computing and this role will grow. Today the world is interconnected, tomorrow the world will only be more so. This loosely coupled distributed environment is exactly the environment for which agents were created. In parallel with this growth of networked computation is an increase in the sophistication that we, as users, personal or business, expect from our software systems. If it has a processor, we want it connected to the world and doing something useful for us. Agent frameworks, algorithms, or methodologies that support flexible, dynamic, intelligent, or autonomous management of information or resources are potentially commercial discriminators today and are assuredly future discriminators. The agent paradigm can and will make a difference and have impact on commercial computing.

The AAMAS industry track is a special track at the AAMAS conference that runs in parallel with the regular AAMAS scientific track. The track is dedicated to fostering commercial, or real world, impact for agent technologies. The track features presentations, panels, invited talks, and demonstrations from industrial participants, giving them an opportunity to showcase the state of the art in agent technology. The industry track aims to foster mutually beneficial links between those engaged in foundational scientific research and those working to make autonomous agents and multi-agent systems a commercial reality.

This year's track follows on the successful industry track held at AAMAS05 in Utrecht, the Netherlands. The AAMAS05 track featured 15 high quality papers, an invited talk by Dr. Mark Greaves of DARPA (Defense Advanced Research Projects Agency), and a panel chaired by Dr. Tom Wagner on Developing Impact for Agents. The AAMAS05 papers discussed the application of agent technology to commercial problems, presented case studies, or assessed technology adoption issues. The selection of papers at AAMAS05 was competitive and the best papers were reported in the IEEE Journal of Intelligent Systems.

This year the conference features 15 high quality papers that present applications of multi-agent systems or discuss deployment related aspects of agent technology. Authors include leaders from research and academe but also commercial stakeholders and key decision makers from agent oriented software companies. Specific application areas include the use of agents in space exploration and defense, logistics and supply chain management, medical applications and enterprise information systems. This year's papers also present more technical applications in coordination, collaboration and multi-agent learning. The proceedings also include the lessons, experiences and challenges generated during the operation of AgentLink III – a Co-ordination Action funded by the European Union for Agent Based Computing. The AAMAS06 invited speaker is Steve Benfield of Agentis Software. The industry panel is chaired by Dr. Brad Clement of NASA JPL.

We are glad to see continued strong interest in the industry track. Agents can and will have impact, and it is our job as a community to foster this impact.

Tom Wagner and Michal Pechoucek (Industry Track Co-Chairs) March 2006

CONFERENCE OFFICIALS

General Chairs

- Hideyuki Nakashima, Future University - Hakodate, Japan
- Michael Wellman, University of Michigan, USA

Program Chairs

- Peter Stone, The University of Texas at Austin, USA
- Gerhard Weiss, Technical University Munich, Germany

Local Organization Chair

- Ei-ichi Osawa, Future University - Hakodate, Japan

Local Treasurer

- Hitoshi Matsubara, Future University - Hakodate, Japan

Finance Chair

- Liz Sonenberg, University of Melbourne, Australia

Workshops Chair

- Jiming Liu, Hong Kong Baptist University, China

Tutorials Chair

- Yves Lesperance, York University, Canada

Demos Chairs

- Ana Lucia C. Bazzan, Universidade Federal do Rio Grande do Sul, Brasil
- Regis Vincent, SRI International, USA

Industry Track Chairs

- Michal Pechoucek, Gerstner Laboratory and Czech Technical University, CZ
- Tom Wagner, Defense Advanced Research Projects Agency (DARPA), USA

Doctoral Mentoring Co-chairs

- Ahlem Ben Hassine, NICT, Japan
- Toru Ishida, Kyoto University, Japan

Publicity Chair

- Chengqi Zhang, University of Technology, Australia

Sponsorship Chairs

- **Europe:** Steven Willmott, Universitat Politecnica de Catalunya, Spain
- **East Asia:** Akihiko Ohsuga, Toshiba Co., Japan
- **US & Canada:** Van Parunak, Altarum Institute, USA

Publications Chair

- Takayuki Ito, Nagoya Institute of Technology, Japan

Student Scholarship Chairs

- Eduardo Alonso, City University, London, UK
- Sandip Sen, University of Tulsa, USA

Exhibits Chair

- Keiji Suzuki, Future University - Hakodate, Japan

SENIOR PROGRAM COMMITTEE

- Elisabeth Andre, Germany
- Tucker Balch, USA
- Mike Bowling, Canada
- Monique Calisti, Switzerland
- Keith Decker, USA
- Marie DesJardins, USA
- Virginia Dignum, The Netherlands
- Jurgen Dix, Germany
- Edmund Durfee, USA
- Maria Gini, USA
- Paolo Giorgini, Italy
- Amy Greenwald, USA
- Brian Henderson-Sellers, Australia
- Adele Howe, USA
- Gal Kaminka, Israel
- Barbara Keplicz, Poland
- Sven Koenig, USA
- Sarit Kraus, Israel
- Kate Larson, Canada
- Kevin Leyton-Brown, Canada
- Alessio Lomuscio, University College London, UK
- Maja Mataric, USA
- Nicolas Maudet, France
- Itsuki Noda, Japan
- Lin Padgham, Australia
- Ana Paiva, Portugal
- David Parkes, USA
- Simon Parsons, USA
- Michal Pechoucek, Czech Republic
- Joelle Pineau, Canada
- Daniel Polani, UK
- David Pynadath, USA
- Martin Riedmiller, Germany
- Jeff Rosenschein, Israel
- Paul Rybski, USA
- Jaime Sichman, Brazil
- Carles Sierra, Spain
- Betsy Sklar, USA
- Tomoichi Takahashi, Japan
- Kagan Tumer, USA
- Mike Wooldridge, UK
- Franco Zambonelli, Italy

PROGRAM COMMITTEE

- Adrian Agogino, USA
- Thomas Ågotnes, UK
- H. Levent Akin, Turkey
- Klaus-Dieter Althoff, Germany
- Leila Amgoud, France
- Luis Antunes, Portugal
- Ruth Aylett, UK
- Jacky Baltes, Canada
- Bikramjit Banerjee, USA
- Bernhard Bauer, Germany
- Joseph Beck, USA
- Michael Beetz, Germany
- Federico Bergenti, Italy
- Michael Berger, Germany
- Douglas Blank, USA
- Rafael Bordini, UK
- Vicent Botti, Spain
- Felix Brandt, Germany
- Frances Brazier, The Netherlands
- Jan Broersen, The Netherlands
- Chris Brooks, USA
- Brett Browning, USA
- Hans-Dieter Burkhard, Germany
- Giacomo Cabri, Italy
- Stefano Carpin, Germany
- Marco Carvalho, USA
- Cristiano Castelfranchi, Italy
- Marc Cavazza, UK
- Lawrence Cavedon, Australia
- Brahim Chaib-draa, Canada
- Stephan Chalup, Australia
- Xiaoping Chen, China
- Sonia Chernova, USA
- Carlos Chesnevar, Spain
- Yann Chevaleyre, France
- Keith Clark, UK
- Bradley Clement, USA
- Robin Cohen, Canada
- John Collins, USA
- Thomas Collins, USA
- Vincent Conitzer, USA
- Dan Corkill, USA
- Massimo Cossentino, Italy
- Jeffrey Cox, USA
- Jacob Crandall, USA
- Stephen Cranefield, New Zealand
- Elisabeth Crawford, USA
- Torbjorn Dahl, UK
- Rajarshi Das, USA
- Mehdi Dastani, The Netherlands
- Esther David, UK
- Paul Davidsson, Sweden
- Fiorella de Rosis, Italy
- John Debenham, Australia
- Joerg Denzinger, Canada
- Giovanna Di Marzo, Switzerland
- M. Bernardine Dias, USA
- Frank Dignum, The Netherlands
- Mark D'Inverno, UK
- Kurt Dresner, USA
- Alexis Drogoul, France
- Paul Dunne, UK
- Thomas Eiter, Austria
- Amal El Fallah-Seghrouchni, France
- Ulle Endriss, The Netherlands
- Tara Estlin, USA
- Rino Falcone Italy
- Maria Fasli, UK
- Shaheen Fatima, UK
- Felix Fischer, Germany
- Klaus Fischer, Germany
- Michael Fisher, UK
- Nicoletta Fornara, Switzerland
- Aram Galstyan, USA
- Alessandro Garcia, UK
- Andrew Garland, USA
- Les Gasser, USA
- Matthew Gaston, USA
- Brian Gerkey, USA
- Mohammad Ghavamzadeh, Canada
- Chiara Ghidini, Italy
- Nigel Gilbert, UK
- Andrew Gilpin, USA
- Marie Pierre Gleizes, France
- Piotr Gmytrasiewicz, USA
- Llu・・Godo, Spain
- Daniela Godoy, Argentina
- Claudia Goldman, Israel
- Steve Goschnick, Australia
- Jon Gratch, USA
- Dominic Greenwood, Switzerland
- Gery Gutnik, Israel
- Meirav Hadad, Israel
- Salima Hassas, France
- Andreas Herzig, France
- Henry Hexmoor, USA
- Benjamin Hirsch, Germany
- Eric Horvitz, USA
- Michael Huhns, USA
- Luke Hunsberger, USA
- Ullrich Hustadt, UK
- Luca Iocchi, Italy
- Charles Isbell, USA
- Mitsuru Ishizuka, Japan
- Takayuki Ito, Japan
- Nobuhiro Ito, Japan

- Wojtek Jamroga, Germany
- Sverker Janson, Sweden
- Odest Jenkins, USA
- Rune Jensen, Denmark
- Lewis Johnson, USA
- Christopher Jones, USA
- Catholijn Jonker, The Netherlands
- Anupam Joshi, USA
- Anthony Karageorgos, Greece
- Kamal Karlapalem, India
- Hidenori Kawamura, Japan
- Stefan Kirn, Germany
- Matthias Klusch, Germany
- Jelle Kok, The Netherlands
- Manuel Kolp, Belgium
- Stefan Kopp, Germany
- Manolis Koubarakis, Greece
- Gerhard Kraetzschmar, Germany
- Daniel Kudenko, UK
- Satoshi Kurihara, Japan
- Koichi Kurumatani, Japan
- Kazuhiro Kuwabara, Japan
- Yoshitaka Kuwata, Japan
- Han La Poutre, The Netherlands
- Jerome Lang, France
- Nuno Lau, Portugal
- Jamie Lawrence, UK
- James Lawton, USA
- Jaeho Lee, Korea
- Christian Lemaitre, Mexico
- Kristina Lerman, USA
- Yves Lesperance, Canada
- Victor Lesser, USA
- James Lester, USA
- Churn-Jung Liau, Taiwan
- Michael Littman, USA
- Wei Liu, Australia
- Yaxin Liu, USA
- Michael Luck, UK
- Marco Mamei, Italy
- Vladimir Marik, Czech Republic
- Shaul Markovitch, Israel
- Alcherio Martinoli, Switzerland
- Peter McBurney, UK
- John-Jules Meyer, The Netherlands
- Andre Meyer, The Netherlands
- Jay Modi, USA
- Pavlos Moraitis, France
- Joerg Mueller, Germany
- Ranjit Nair, USA
- Hideyuki Nakanishi, Japan
- Yukiko Nakano, Japan
- Tomoharu Nakashima, Japan
- Daniele Nardi, Italy
- Son Nguyen Hung, Poland
- Matthias Nickles, Germany
- Fariba Sadri, UK
- Monica Nicolescu, USA
- Katsumi Nitta, Japan
- Pablo Noriega, Spain
- Timothy Norman, UK
- Masayuki Numao, Japan
- Oliver Obst, Germany
- James Odell, USA
- Keith O'Hara, USA
- Masayuki Ohta, Japan
- Eugenio Oliveira, Portugal
- Andrea Omicini, Italy
- Charlie Ortiz, USA
- Sascha Ossowski, Spain
- Eric Pacuit, The Netherlands
- Sebastien Paquet, Canada
- Sunju Park, Korea
- Van Parunak, USA
- Marc Pauly, USA
- Shamimabi Paurobally, UK
- Adrian Pearce, Australia
- Catherine Pelachaud, France
- Wojciech Penczek, Poland
- Anna Perini, Italy
- Paolo Petta, Austria
- Hanh Pham, USA
- Steve Phelps, UK
- Jeremy Pitt, UK
- Enric Plaza, Spain
- Pascal Poupart, Canada
- Rob Powers, USA
- Henry Prakken, The Netherlands
- Chris Preist, UK
- Helmut Prendinger, Japan
- Bob Price, Canada
- Mikhail Prokopenko, Australia
- Riccardo Pucella, USA
- Michael Quinlan, Australia
- Zinovi Rabinovich, Israel
- Iyad Rahwan, United Arab Emirates
- Omer Rana, UK
- Rojas Raul, Germany
- Daniel Reeves, USA
- Martin Rehak, Czech Republic
- Luis Paulo Reis, Portugal
- Alessandro Ricci, Italy
- Debbie Richards, Australia
- Thomas Rist, Germany
- Juan Antonio Rodriguez-Aguilar, Spain
- Thomas Roefer, Germany
- Nico Roos, The Netherlands
- Avi Rosenfeld, Israel
- Michael Rovatsos, UK
- Zachary Rubinstein, USA
- Matthew Rudary, USA
- Javier Ruiz-de-Solar, Chile
- Zsofia Ruttkay, The Netherlands
- Mario Verdicchio, Italy

- Sebastian Sardina, Australia
- Alberto Sardinha, Brazil
- Sanem Sariel, Turkey
- David Sarne, USA
- Ichiro Satoh, Japan
- Ken Satoh, Japan
- Paul Scerri, USA
- Ute Schmid, Germany
- Renate Schmidt, UK
- Marco Schorlemmer, Spain
- Sandip Sen, USA
- Marek Sergot, UK
- Onn Shehory, Israel
- Saeed Shiry, Iran
- Marius Silaghi, USA
- Guillermo Simari, Argentina
- Von-Wun Soo, Taiwan
- Alvaro Soto, Chile
- Diana Spears, USA
- Donald Steiner, USA
- Leon Sterling, Australia
- Matthew Stone, UK
- Ashley Stroupe, USA
- Nathan Sturtevant, Canada
- Gita Sukthankar, U.S.
- Niranjan Suri, USA
- Behnke Sven, Germany
- Marcin Szczuka, Poland
- Yasutake Takahashi, Japan
- Ikuo Takeuchi, Japan
- Milind Tambe, USA
- Brian Tanner, Canada
- Ahmed Tawfik, Canada
- Takao Terano, Japan
- Gerry Tesauro, USA
- Daniel Thalmann, Switzerland
- John Thangarajah, Australia
- Simon Thompson, UK
- Robert Tolksdorf, Germany
- Francesca Toni, UK
- Paolo Torroni, Italy
- Jan Treur, The Netherlands
- Paola Turci, Italy
- Karl Tuyls, The Netherlands
- Wiebe van der Hoek, UK
- Hans van Ditmarsch, New Zealand
- Rogier van Eijk, The Netherlands
- Sieuwert van Otterloo, The Netherlands
- Laszlo Varga, Hungary
- Eloisa Vargiu, Italy
- Wamberto Vasconcelos, UK
- Richard Vaughan, Canada
- Javier Vazquez-Salceda, Spain
- Katja Verbeeck, Belgium
- Rineke Verbrugge, Netherlands
- Bart Verheij, The Netherlands
- Ioannis Vetsikas, UK
- Hannes Vilhj 逍拯 sson, USA
- Regis Vincent, USA
- Mirko Viroli, Italy
- Giuseppe Vizzari, Italy
- Bao Vo, Australia
- Alfredo Weitzenfeld, Mexico
- Danny Weyns, Belgium
- Shimon Whiteson, USA
- Marco Wiering, The Netherlands
- Andrew Williams, USA
- Steven Wilmott, Spain
- Michael Winikoff, Australia
- Cees Witteveen, The Netherlands
- Wayne Wobcke, Australia
- Bozena Wozna, UK
- Peter Wurman, USA
- Tomohisa Yamashita, Japan
- Holly Yanco, USA
- Makoto Yokoo, Japan
- R. Michael Young, USA
- Yingqian Zhang, Germany
- Lina Zhou, USA
- Martin Zinkevich, Canada

AUXILIARY REVIEWERS

- Frederick Asselin
- Marcelo Armentano
- Philippe Balbiani
- Alejandra Barrera
- Silvio Becher
- Roxana Belecheanu
- Heni Ben Amor
- Elisabetta Bevacqua
- Sooraj Bhat
- Aliaksandr Birukou
- Sander Bohte
- Alexander Bolotov
- Tibor Bosse
- Nelis Boucke
- Gavin Brown
- Volha Bryl
- Pieter Buzing
- David Cabanillas
- Marcela Capobianco
- Henrique Cardoso
- Carlos Carrascosa
- Michal Chalamish
- Dave Cleaver
- Roberta Coelho
- Toni Conde
- Nikolaus Correll
- Vincent Corruble
- Julie Cowart
- Vasa Curcin
- Jiangbo Dang
- Jurriaan van Diggelen
- Yannis Dimopoulos
- Phan Minh Dung
- Rosemary Emery-Montemerlo
- Christian Erbelding
- Mark Esteva
- Pedro Faria
- Alessandro Farinelli
- Al Fedoruk
- Luca Ferrari
- Michael Fink
- Noria Foukia
- Florian Fuchs
- Kobi Gal
- Andres Garcia-Camino
- Alejandra Garcia-Rojas
- Benoit Gaudou
- Jean-Pierre George
- Valentin Goranko
- Giorgio Grisetti
- Davide Grossi
- Tarik Hadzic
- Christian Hahn
- James Harland
- Peter Hebden
- Annerieke Heuvelink
- Koen Hindriks
- Ian Hodkinson
- Michael Holmes
- Mark Hoogendoorn
- Jian Alan Huang
- Jingshan Huang
- James Geoff
- Giovambattista Ianni
- Stefan Johansson
- Steven de Jong
- Vicente Julian
- Nidhi Kalra
- Ken Kaneiwa
- Roni Katz
- Ryan Kelly
- Mary Koes
- Michael Kolb
- Rob Kremer
- Hidekazu Kubota
- Srikanth Kallurkar
- Willem Labuschagne
- Kiran Lakkaraju
- Rogerio de Lemos
- Raz Lin
- Kendall Lister
- Thomas Lochmatter
- Kevin Lochner
- Dominique Longin
- Emiliano Lorini
- Peter-Paul van Maanen
- Cristian Madrigal-Mora
- Andreia Malucelli
- Maurizio Mancini
- Wenji Mao
- Rudolph Mappus IV
- Pedrito Maynard-Zhang
- Tarek Melliti
- Carlos Merida-Campos
- Koen Mertens
- Sebastian Mhatre
- Greg Mori
- Luis Mota
- Steve Munroe
- Jan Murray
- Nide Naoyuki
- Mark Nelson
- Brendan Neville
- Luis Nunes
- Lyndon Nixon
- Magalie Ochs
- Ikki Ohmukai
- Masashi Okamoto
- Anna Okhmatovskaia
- Marcos de Oliveira
- Michael Papasimeon
- Praveen Paruchuri
- Jonathan Pearce
- Loris Penserini
- Jan Persson
- Julien Pettre
- Giovanni Pezzulo
- Gauthier Picard
- Michael Pirker
- Pierrick Plamondon
- Marc Ponsen
- Viara Popova
- Elena Paslaru
- Jim Pugh
- Raffaele Quitadamo
- Franco Raimondi
- Birna van Riemsdijk
- David Roberts
- Valentin Robu
- Juan Antonio Rodriguez-Aguilar
- Pierre Roduit
- Alex Rogers
- Alexander Romanovsky
- Tom Roper
- Rosaldo Rossetti
- Luca Sabatucci
- Bastin Tony Roy Savarimuthu
- Klaus Schild
- Roman Schindlauer
- Klaus Schmid
- Valeria Seidita
- Christian Seitz
- Juan Manuel Serrano
- Alexei Sharpanskykh
- Carles Sierra
- Viviane Silva
- Esteban Leon Soto
- Nicolas Soubouret
- Matthijs Spaan
- Kostas Stathis
- Renze Steenhuisen
- Arnon Sturm
- Angelo Susi
- Samarth Swarup
- Kuldar Taveter
- Peter Tiedemann
- Ingo Timm
- Dmytro Thykonov
- Dmitri Toropov
- Leon van der Torre
- Luca Tummolini
- Andrew Trotman
- Douglas Vail
- Pradeep Varakantham
- Tim Verwaart

- Frederic Vexo
- Bao Vo
- Yung-Chun (Justin) Wan
- Jun Wang
- Michael Watzke
- Mathijs de Weerdt
- Nicola Zannone
- Ingo Zinnikus
- Vittorio Amos Ziparo

INDUSTRY TRACK OFFICIALS

Program Chairs

- Michal Pechoucek, Gerstner Laboratory and Czech Technical University, CZ
- Tom Wagner, Defense Advanced Research Projects Agency (DARPA), USA

Program Committee

- David Allsopp, QinetiQ
- Fabio Bellifemine, Telecom Italia Lab
- Michael Berger, Siemens Corporate Technology
- Jeff Bradshaw, IHMC
- Sven Brueckner, Altarum Institute
- Bernard Burg, Panasonic Research
- Stefan Bussmann, Daimler Chrysler
- Monique Calisti, Whitestein Technologies
- Todd Carrico, Cougaar Software
- Patricia Charlton, Motorola
- Adam Cheyer, SRI International
- Jonathan Dale, Fujitsu
- Ian Dickinson, Hewlett-Packard Laboratories
- Ravi Gorthi, SETLabs, Infosys Technologies Limited
- Mark Greaves, Vulcan Inc.
- Jon Himoff, Magenta Technology
- Martin Hofmann, Lockheed Martin
- David Kinny, Agentis
- Robert Kohout, Strategic Analysis
- James Lawton, US Air Force Research Laboratory
- Dickson Lukose, DL Informatique, Malaysia
- Vladimir Marik, Rockwell Research Center
- David Martin, SRI
- Peter McBurney, University of Liverpool
- Nanjangud Narendra, IBM
- Paul O'Brien, BT Group
- James Odell, James Odell Associates USA
- Van Dyke Parunak, Altarum Institute
- Michal Pechoucek, Gerstner Laboratory, Czech Technical University
- Chris Preist, Hewlett Packard Labs, Bristol
- Chris Reed, University of Dundee
- Ralph Ronnquist, AOS Australia
- Jeff Rosenschein, Hebrew University and co-founder of AgentSoft
- David Sadek, France Telecom, R&D Division
- Onn Shehory, IBM ? Haifa Research Labs
- John Shepherdson, British Telecommunications
- Petr Skobelev, Magenta Technology CTO
- Donald Steiner, Quantum Leap
- Niranjan Suri, IHMC
- Simon Thompson, British Telecom
- Tom Wagner, DARPA

SPONSORSHIP AND SUPPORT

Sponsors

- ACM SIGART
- The International Foundation for Multiagent Systems (IFMAS)
- The International Workshop on Agent Theories, Architectures, and Languages (ATAL)

Corporate Supporters

- Toshiba, Japan
- Midoriya Electric, Japan
- Microsoft Research, USA
- DENSO IT LABORATORY, Japan
- Magenta Technology / Magenta Corporation Ltd., UK
- Microsoft, Japan
- Fujitsu, Japan
- Communication Technologies, Japan
- Hitachi, Japan
- Victor Company of Japan, Limited (JVC)
- Pioneer Corporation, Japan
- SONY, Japan
- Nihon Unisys, Ltd., Japan
- Yokogawa Electric, Japan
- Altarum Institute, USA
- Lostwax Media Ltd., UK
- Soar Technology, USA
- Whitestein Technologies, Switzerland

Other Supporters

- Special Interest Group at Multi-Agent and Japan Society for Software Science and Technology, Japan
- Defense Advanced Research Projects Agency, USA
- Intelligent Agent Society of Japan, Japan
- National Science Foundation (NSF), USA
- Future University-Hakodate, Japan

Logos


SIG-MACC
TOSHIBA
MIDORIYA
ELECTRIC CO., LTD.
Microsoft
Research
DENSO
IT LAB
MAGENTA
TECHNOLOGY
Microsoft
DARPA
Intelligent Agent Society of Japan
FUJITSU
COMTEC
STRIKES BACK
HITACHI
Inspire the Next
JVC
The Perfect Experience
Pioneer
SONY
UNISYS
YOKOGAWA
ALTARUM
lost wax
Soar Technology
Thinking inside the box.
WHITESTEIN
Technologies

Invited Talks

Towards Emergence of Communication: A Cognitive Developmental Robotics Approach

Minoru Asada
JST ERATO Asada Synergistic Intelligence Project and Dept. of Adaptive Machine Systems
Graduate School of Eng., Osaka University, Suita, Osaka 565-0871, Japan
asada@jeap.org

ABSTRACT

This paper presents an introduction of the JST ERATO Asada Synergistic Intelligence Project (hereafter, SI) and its preliminary studies on emergence of communication. The project aims at building cognitive developmental artificial agents (humanoids), understanding natural agents (humans), and their mutual feedbacks. The project consists of four groups: (1) Physio-SI: whole body dynamic motions such as walking, running, and jumping, (2) Perso-SI: cognitive developmental robotics including body image, imitation, and language communication, (3) Socio-SI: emergence of communication and society by androids, and (4) SI-mechanism: neuroscientific supports for Physio, Perso, and Socio-SIs. As a preliminary study, vowel acquisition based on visual and auditory mutual imitation is given. It aims at modelling mother-infant interaction in vowel acquisition based on the findings in infant studies, such as lip shape imitation and bias from mother's vocalization onto the infant auditory perception.

1. INTRODUCTION

The recent advances in information and robot technology (in short, IRT) enable us to build humanoid robots that have a large number of DoFs and various kinds of many sensors. Human-like motions are realized on these robots, and the shape and appearance have become closer and closer to us. [12, 4].

However, the current robotics lacks the faculties of language communication with ordinary people and of intelligent behavior generation in various situations such as at home. The fundamental relationship between humans and robots would become more important since these robots would be introduced into our lives in near future, and therefore the mechanisms of adaptation and development of both humans and robots should be taken into account in order to find the correct direction of the technology in future.

"Synergistic Intelligence (hereafter, SI)," the title of JST (Japan Science and Technology Agency) ERATO (Exploratory Research for Advanced Technology) Asada project, emerges intelligent behaviours through the interaction with environment including humans. Synergistic effects with brain science, neuroscience, cognitive science, and developmental psychology are expected. SI is one approach to a new discipline called "Humanoid Science" that aims at providing a new way of understanding ourselves and a new design theory of humanoids through mutual feedback between the design of human-like robots and human-related science.

"Humanoid Science" under which a variety of researchers from robotics, AI, brain science, cognitive science, psychology and so on are seeking for new understanding of ourselves by constructivist approaches, that is expected to produce many applications.

SI adopts a methodology called "Cognitive Developmental Robotics" (hereafter, CDR) [1] that consists of the design of self-developing structures inside the robot's brain, and the environmental design: how to set up the environment so that the robots embedded therein can gradually adapt themselves to more complex tasks in more dynamic situations. Unstructured terrains are opponents for adaptive walkers to negotiate with in order to generate dynamic motions. The caregiver's behaviour to a robot is one environmental design issue since parents, teachers, and other adults adapt themselves to the needs of children according to each child's level of maturity and the particular relationship they have developed with that child.

One of the most formidable issues in SI is "Nature vs. Nurture": to what extent should we embed the structure, and to what extent should we expect the development triggered by the environment? A symbolic issue is " Language Acquisition. "How can robots emerge the symbol in the social context? What is the essential element in this process?

This paper presents an introduction of SI and its preliminary studies on emergence of communication. The project aims at building cognitive developmental artificial agents (humanoids), understanding natural agents (humans), and their mutual feedbacks (see Fig.1). The project consists of four groups: (1) Physio-SI: whole body dynamic motions such as walking, running, and jumping, (2) Perso-SI: cognitive developmental robotics including body image, imitation, and language communication, (3) Socio-SI: emergence of communication and society by androids, and (4) SI-mechanism: neuroscientific supports for Physio, Perso, and Socio-SIs. As a preliminary study, vowel acquisition based on visual and auditory mutual imitation is given. It aims at modelling mother-infant interaction in vowel acquisition based on the findings in infant studies, such as lip

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

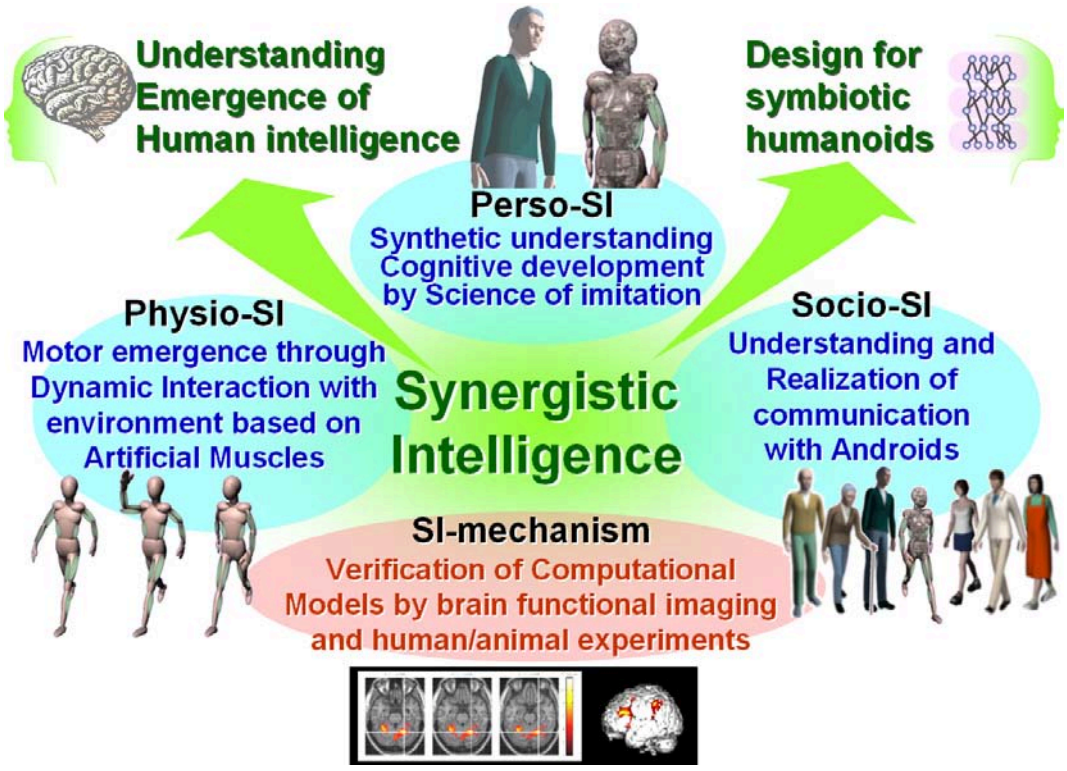


Figure 1: An overview of Synergistic Intelligence Project

shape imitation and bias from mother's vocalization onto the infant auditory perception.

2. AN OVERVIEW OF SI

In this project, we will establish a new discipline that explains the essential relationship between humans and robots in future symbiotic society. Systematic and continuous modelling of " embodiment, " " autonomy, " and " sociality " from a viewpoint of emergence and development will be performed, and constructivism will be shown by realizing these models on real robots. Fundamental principles for adaptive and developable robots will be revealed and may contribute to human society.

An expected goal is realization of three years old humanoid that can show the capabilities of 1) seamless transition among running, jumping, walking, and stop, and whole body motion such as rope skipping, 2) naming game with toy playing and novel behavior generation from imitation, and 3) self-expression, attention, and conversation. The rest of this section gives overviews of four groups of SI.

2.1 Physically Synergistic Intelligence

Physio-SI in short, aims at understanding emergence of human intelligence from dynamic connection between body structure and environment with artificial muscles.

The agent behavior is determined by not only its dynamics but also environmental one if its compliance is comparable to that of the environment. This interaction may help the agent to be adaptive and robust against changes of the environment, which cannot be taken for the rigid agents with high-ratio geared motors. In this project, we utilize such physical interactions to realize dynamic skills such as walking, jumping, and running, and try to figure out the methodology to emerge synergistic intelligence:

- **Walking on uneven terrains utlizing antagonistic driven legs:** By utilizing flexibility provided by antagonistic driven legs and ballistics of the body, we will derive a methodology to realize adaptive walking on uneven terrains Preliminary results are shown in [11].
- **Realization of dynamic behaviors:** We will realize dynamic behaviors such as jumping and running by utlizing compliance and dymamic power provided by the antagonistic drive.
- **Smooth transition between dynamic behaviors:** We will propose a framework to deal with different modes of locomotion, walking and running, and develop a "Motor Skill Emerging Humanoid" that can realize these locomotion modes.
- **Cognitive approach to locomotion:** From the standpoint of building a Motor SKill Emerging Humanoid, we approach to the synergistic intelligence mechanisms and will understand the foundation of emergence of dynamic behaviors.

2.2 Interpersonally-Synergistic Intelligence

Fusing together the capabilities of real world behavior and communication will be the critical step towards the truly humanoid robot intelligence. Interpersonally-Synergistic Intelligence group (Perso-SI in short) aims at constructing

synthetic robotic models that account for the process of cognitive development starting from bodily movements of fetuses and reaching rich meaningful communication of infants. This will bridge the research of the other two groups, namely, Physio-Synergistic Intelligence and Socio-Synergistic Intelligence.

Our developmental cognitive model would account for (but not limited to) the following cognitive functionalities:

1. Acquisition of brain internal representation of self body through whole body sensory-motor learning,
2. Learning about objects and tools,
3. Emergence and development of behavior imitation,
4. Concept/imagery formation by abstraction of behaviors, and
5. Fusing together the behavior concepts/imitation with interactions through gestures, vocalizations, and facial expressions.

Some of our models will be based on the neuroscientific models provided by the Synergistic Intelligence Mechanisms group; The model of object manipulation for topic 2, imitation learning for topic 3, and motor imagery for topic 4.

Two of the hardware platforms required for the above research are; (a) whole-body sensory-motor learning robot, and (b) behavior imitation learning robot. We will develop these robots and use them for the above described research on cognitive models. Robot (a) will have a whole body tactile sensing skin and whole body motion capabilities, and will be used for experiments such as body image acquisition by sensory integration learning. Robot (b) will emphasize object manipulation capabilities and their integration with audio-visual sensing, and will be used for experiments on imitation learning, concept acquisition, and fusing with communication. In addition, we will construct (c) a computer simulation of sensory-motor learning of fetuses, in order to carry out learning experiments and model validation for very early stages of sensory-motor learning.

2.3 Socially Syngenetic intelligence

This group studies developmental mechanisms of communication and evaluations of intelligence based on human subjects and the human society. We, humans, anthropomorphize the object for conversation. This is an essential human ability and a hint for investigating principles of communicative and intelligent machines by engineering and scientific approaches. This idea guides us to a new cross-interdisciplinary framework called Android Science [4]. In this framework, we tackle the following issues

- Development of Man-Made Man (M3: M cube): In order to realize robots that can be accepted by the human society, we need to develop a human-like robot. The robot developed in this project is different from other robot-like humanoids and very humanlike androids (**??**). The robot M3 is going to have about 50 pneumatic actuators, soft skin covering the whole body, and various humanlike sensors with a child size.
- Mechanisms for establishing social relationships with humans: We, humans, have various interactive behaviors for establishing fundamental relationships with others as humans, for example eye movement while looking at each other, personal distance, shard attention, synchronized movement while interacting, and reactions with tactile sensation. These human behaviors are precisely measured and the numerical models are developed [5]. Then, these programmable models of the behaviors are going to be implemented into M3.
- Constructive approach for understanding social mechanisms of humans and robots: We are going to implement more complicated and social behaviors to the robot that has the fundamental relationships with humans. This is the constructive approach [3] that starts from the basic elements and gradationally implements more complex behaviors. Finally, we acquire knowledge on what is a human and what is the human society by using the develop robot.

2.4 Synergistic Intelligence Mechanism

SI-Mech in short, will conduct the following three issues:

1. study of neural correlates of interactive control mechanism between hands and objects,
2. investigation of the brain mechanism of imitation and imaging, and
3. construction of an acquisition model for fundamental functions of communication.

In study of neural correlates of interactive control mechanism between hands and objects, we will use tracing of line drawing tasks under various conditions to identify the neural circuits involved in smooth manipulation of hands and objects. We will also clarify the relationship between some motor and mental disorders and the neural substrates. Furthermore, Nagai et al. [7] indicated that patients with Williams syndrome can trace the drawings of objects, but can not copy them. To investigate this dissociation between tracing and copying, we will use functional imaging and neuropsychological methods to clarify the difference in brain regions involved in these behaviors.

On the other hand, mechanisms of prediction and imitation play important roles in acquiring functions of communication. Imazu et al. [2] studied neural correlates of tool (chopstick) use with functional neuroimaging and revealed the following finding: actual use of chopstick activates the right cerebellum, whereas pantomiming and imagery of chopstick use activate the left inferior parietal lobule. This result suggests that different neural networks are involved in actual use, and pantomime and imagery of tool use. This could explain the dissociation observed in patients with ideomotor apraxia: they unable to pantomime with tools, despite their ability to manipulate tools actually.

For the purpose of 'the investigation of the brain mechanism of imitation and imaging', we will first examine those of action in the human brain with functional MRI. Second we will make a research on both the deficit and the learning process of action imitation of autistic children who have communication deficits, modeling the relationship between action imitation and communication function. In addition, we are going to compare the aforesaid study of William's syndrome with that of the autism and clarify the essence of communication function.

Furthermore, the 'motor sequence prediction learning hypothesis' on the basis of the brain imaging studies as well as the anatomical ones has been proposed. There are two main assumptions in the hypothesis. First, it is assumed that the functions of left BA44 and BA45 in Broca's area differ from each other; left BA45 is involved in motor sequence prediction and left BA44 is implicated in motor sequence generation, interacting with the basal ganglia respectively. Second, left BA45 makes contact with the temporal lobe through long-distance association fibers to integrate semantic and syntactic information. Finally, in terms of 'the construction of an acquisition model for fundamental functions of communication', we plan to construct an acquisition model through brain imaging studies based on the 'motor sequence prediction learning hypothesis'.

3. VOWEL ACQUISITION BASED ON VISUAL AND AUDITORY MUTUAL IMITATION

A preliminary study of SI is briefly described, which attempts to model mother-infant interaction in vowel acquisition. A pioneering constructivist approach to building a robot that reproduces a developmental process of infants' vowel acquisition has been conducted by Yoshikawa et al. [13] inspired by the observation in infant study. They have constructed a mother-infant interaction model with robot learning capability and parrot-like teaching by caregiver. However, the robot has not listened his/her own voice, therefore nor actively explored more natural vowels similar to the caregiver.

The study presented in this section extends the previous work in the following manners seeking for more natural interaction. First, a lip is added to the robot to imitate the lip shape of the caregiver in order to accelerate the learning process by constraining the initial exploration area in the formant space which is a well-known sound feature space to distinguish vowels [9]. From the lip shape imitation, the initial locations of vowels are obtained. Second, the pentagon the caregiver's vowels construct in the formant space is utilized as the desired vowels for the robot by shifting this pentagon to the centroid of the initial vowel locations of the robot. Third, mutual imitation between the robot and the caregiver is introduced in order to obtain more natural vowels hypothesizing that the caregiver imitates the robot voice but unconsciously the imitated voice is close to one of his/her own vowels. Through this process, the desired positions specified at the second step are gradually shifted, expecting to more natural ones.

3.1 Vowel imitation based on mother-infant interatction model

An overview of the whole system is shown in Fig. 2 where a vocal robot with articulation and auditory functions interacts with a caregiver who shows lip shape to the robot, and both imitate their voices each other. We slightly modified the vocal robot in [13] by adding the lip structure that has two degrees of freedom corresponding to opening/closing in horizontally and vertically. The shape change of the vocal tract is controlled by four motors instead of five ones in the previous work.

The caregiver has two roles: the first one is to show the lip shape to the robot so that it can imitate the lip shape

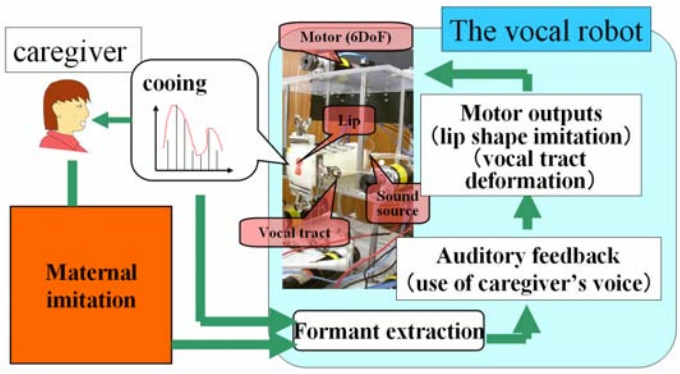


Figure 2: An overview of the whole system

since there are some evidences on the relationship between the lip shape and the vowel utterance. Carre [10] shows the formant changes owing to opening/closing of the lip by his simulation. Patterson and Werker [8] found that the infants have knowledge on the relationship between the lip shape and phoneme. Then, we assume that the robot can imitate the lip shape corresponding to each vowel from the caregiver to initialize its vowels in the formant space for further exploration.

The second one is maternal imitation, that is, imitation of the robot voices. Since the robot also imitates the caregiver's imitation, mutual imitation is expected to lead the robot voices to more natural one. The implicit assumption behind this process is that the caregiver tries to imitate the robot voices, but unconsciously his or her voices are close to his or her own vowels owing to the embodiment (sensorimotor constraint). As a result, the robot voices gradually change to more natural ones.

The vocal robot has six degrees of freedom, two of which are for lip opening/closing. First, we examine the utterance capability of the robot. The motor command is normalized into three levels 0 (free, no deformation), 0.5 (middle), and 1.0 (the maximum deformation), therefore we have 729 (3^6) utterrances. Among them, we picked up 711 utterrances after removing unstable onses.

Next, the effect of the lip shape is examined, that is, two of six DoFs corresponding lip opening/closing is fixed to imitate the human lip shape, and other four DoFs are combined (81 utterances, 3^4). Fig.3 shows the formant distribution of the robot utterances when the lip shape is imitated. The top row indicates the formant distributions of utterances when the lip shape is fixed to imitate that of human's vowel utternaces /a/, /i/, /u/, /e/, and /o/, respectively. The bottom row shows the lip shapes of human and the robot, respectively.

The number of utterances constrained by lip shape is 394 (11 unstable utterances are removed) from 711 without the constraint. This tells that by lip shape imitation, the number of candidates for vowel utterances can be reduced to 55%.

3.2 Learning to acquire more natural vowels through mutual imitation

From the above discussion, the exploration process to find more natural vowel utterance seems easy since the robot can

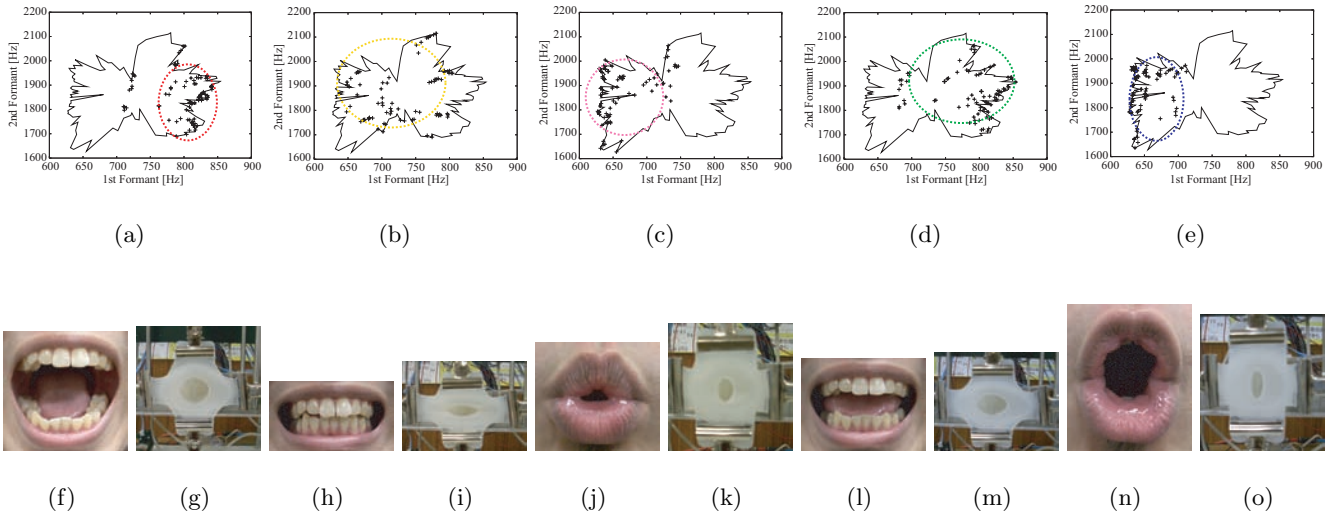


Figure 3: Formant distribution of the utterances (top: /a/, /i/, /u/, /e/, /o/) from lip shape imitation (bottom)

follow the trajectory between its initial position (L/j/, j=a, i, u, e, or o) to the desired vowel position (C/j/', j=a, i, u, e, or o) under the condition that the lip shape is fixed. However, there is no guarantee that the desired vowels can be heard as natural ones. Then, we introduce mutual imitation to shift the desired position so that the robot can acquire more natural ones by hypothesizing that the caregiver's imitation is biased by his/her own sensorimotor constraint, that is, embodiment, and therefore unconsciously implies the direction of natural utterance in the formant space.

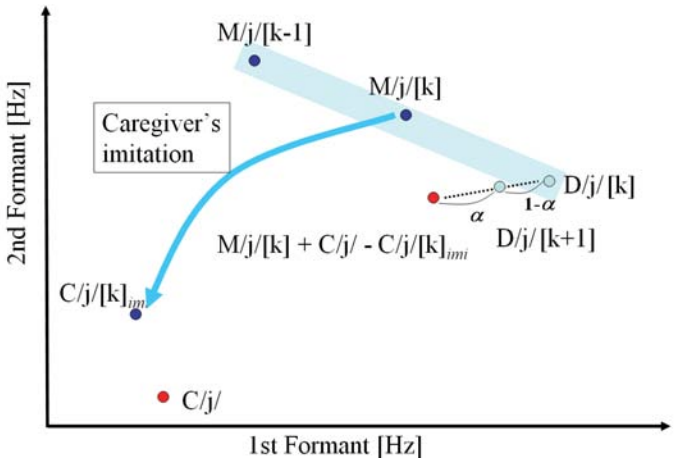


Figure 4: Mutual imitation process on the formant space

Fig.4 explains how mutual imitation can improve the robot utterance. Let D/j/[k], M/j/[k], C/j/[k]$_{imi}$ (i=a, i, u, e, or o) be the desired position, the current position and the utterance that caregiver imitates at the time=k, respectively. Initially, D/j/[0] = C/j/' and M/j/[0] = L/j/, and C/j/ is the caregiver's original vowel.

1. The robot finds the the current position M/j/[k] on the trajectory between M/j/[k-1] and D/j/[k], and utters this position to the caregiver.
2. The caregiver imitates the robot utterance M/j/[k], but the imitated utterance C/j/[k]$_{imi}$ is close to his/her own vowel C/j/ owing to the sensorimotor constraint.
3. A difference vector from C/j/[k]$_{imi}$ to C/j/ is supposed to indicate the direction to more natural vowel utternace.
4. Then, the desired position of the utterance is set at the position M/j/[k] + C/j/ - C/j/[k]$_{imi}$ from the current position.
5. To avoid a big change, a new desired position D/j/[k+1] is set as follows: D/j/[k+1] = α D/j/[k] + (1-α) (M/j/[k] + C/j/ - C/j/[k]$_{imi}$).
6. Repeat the above process until M/j/ does not change (often, due to the limit of the utterance capability).

We apply the learning algorithm to the real robot and a caregiver with α = .7. Here, we stop the learning at the 500th step since almost no changes happened in the robot utterance. Fig.5 shows the results of all vowels where thin folded lines of D/j/ indicate how they are modified according to the mutual imitation. The initial D/j/s are shown as nodes of a pentagon connected by dotted black lines and the final ones as nodes of a pentagon connected by dotted yellow-green lines. The acquired vowels are shown as nodes of a pentagon connected by dotted sky-blue lines. The initial nodes are black and the final ones colored. The human vowels are shown as larger marks such as +, *, and so on, while the robot ones as smaller ones.

Since it is difficult to show how natural the robot utterances are, we prepared the robot utterances without mutual

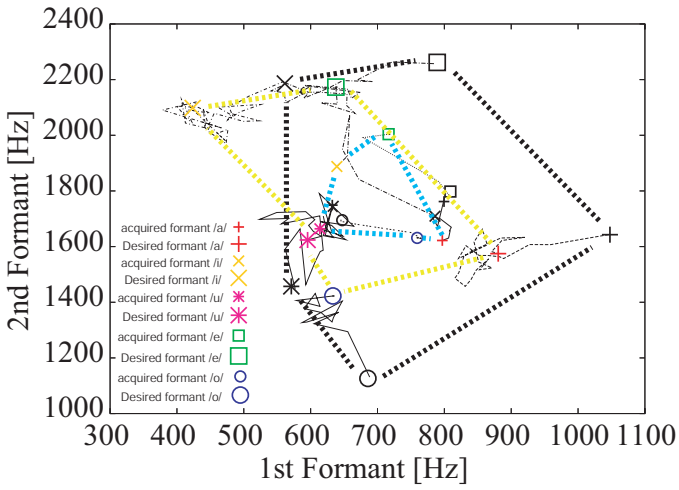


Figure 5: Desired and acquired vowels by a robot at the learning step 500

imitation, that is, the robot acquired the vowels supposing the initial desired positions D/j/[0] untl the end of learning, and asked about 40 naive people (no knowledge on vowel learning). As a result, 70% agreed that the utterances with mutual imitation are more natural than that without mutual imitation.

3.3 Discussion

Currently, we emulated the lip shape imitation by hand. In a real situation, the robot should capture the caregiver's face through its TV camera and extract the lip shape information. Then, it should send the motor commands to imitate the observed lip shape. To do that, the robot should know the correspondence between the body parts such as eyes, nose, and lip, and also how to control its body part, in this case lip motion. Neonatal imitation tells us such a capability [6] though it has been controversy. However, its mechanism has not been revealed yet. Development of boy representation and motor control is one of the key issues in cognitive developmental robotics, and we will attack this problem in near future.

Another issue is how the caregiver behaviors affect the learning results. In this paper, the number of the caregiver is just one, but we have not recorded how he responded (imitated) to the robot. The analysis of such interaction data would be helpful to understand the developmental process of vowel acquisition, and also to build the design policy of the learning robots.

In our study, we adopted the following hypotheses that the lip shape information helps the infant's vowel acquisition and that the caregiver's imitation tends to similar to his/her own vowels. Based on these, we build the interaction model and showed the experimental results. The verification for these hypotheses is not complete. Collaboration with infant study and developmental psychology seems necessary not only to verify them but also to refine or modify the model.

4. CONCLUSION

The brief introdcution of SI project is given and one preliminary study was shown toward emergence of communication. More studies on body representation, imitation, and symbol emergence will be done in our project and hopefully, we like to build a humanoid that can show intelligent, adaptive behaviors including dynamic motions such as runnig and junmping, conversation skill though poor utterance, and facial expression to communicate with us.

5. ACKNOWLEDGMENTS

First of all, the author appreciates the group leaders of the project: Profs. Hiroshi Ishiguro and Koh Hosoda at Osaka University, Prof. Toshio Inui at Kyoto University, and Prof. Yasuo Kuniyoshi at University of Tokyo for the invaluable discussion with them. Next, Dr. Yuichiro Yoshikawa and Mr. Katsushi Miura are appreciated for their preparation of the paper on the preliminary study.

6. REFERENCES

[1] M. Asada, K. F. MacDorman, H. Ishiguro, and Y. Kuniyoshi. Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robotics and Autonomous System*, 37:185–193, 2001.

[2] S. Imazu, T. Sugio, S. Tanaka, and T. Inui. Differences between actual and imagined usage of chopsticks. *An fMRI study. Cortex*, 92(11):in press, 2005.

[3] H. Ishiguro. Toward interactive humanoid robots - a constructive approach to developing intelligent robots. In *Proc. 1st Int. Joint Conf. Autonomous Agents and Multi-Agent Systems AAMAS2002)*, page Invited talk, 2002.

[4] H. Ishiguro. Android science. In *Proc. A CogSci 2005 Workshop on Toward Social Mechanisms of Android Science*, page 25 and 26 July, 2005.

[5] T. Kanda, H. Ishiguro, M. Imai, and T. Ono. Development and evaluation of interactive humanoid robots. *Proceedings of the IEEE*, 92(11):1839–1850, 2004.

[6] A. N. Meltzoff and M. K. Moore. Explaining facial imitation: A theoretical model. *Early Development and Parenting*, pages 179–192, 1997.

[7] C. Nagai, M. Iwata, R. Matsuoka, and M. Kato. Visual recognition disabilities in williams syndrome - why can they trace but cannot copy? -. *Japanese Journal of Neuropsychology*, 17:36–44, 2001.

[8] M. L. Patterson and J. F. Werker. Two-month-old infants match phonetic information in lips and voice. *Developmental Science*, 6(2):191–196, 2003.

[9] R. K. Potter and J. C. Steinberg. Toward the specification of speech. *Journal of the Acoustical Society of America*, 22:807–820, 1950.

[10] C. R. Prediction of vowel systems using a deductive approach. In *In Proceedings of the International Conference on Spoken Language Processing 96*, pages 434–437, 1996.

[11] K. H. T. Takuma and M. Ishikawa. Design and control of a 3d biped robot actuated by antaginistic pairs of pneumatic muscles, Sep. 2005.

[12] K. Terada, Y. Ohmura, and Y. Kuniyoshi. Analysis and control of whole body dynamic humanoid motion – towards experiments on a roll-and-rise motion. In *Proc. of IEEE/RSJ International Conference on*

Intelligent Robots and Systems 2003 (IROS '03), pages (CD–ROM), 2003.

[13] Y. Yoshikawa, M. Asada, K. Hosoda, and J. Koga. A constructivist approach to infants' vowel acquisition through mother-infant interaction. *Connection Science*, 15(4):245–258, Dec 2003.

Making a Strong Business Case for Multiagent Technology

Steve S. Benfield
CTO, Agentis Software
1200 Ashwood Parkway
Suite 410, Atlanta, GA 30338
+1 678.368.4105
sbenfield@
agentisoftware.com

Jim Hendrickson
VP Global Services, Agentis Software
1200 Ashwood Parkway
Suite 410, Atlanta, GA 30338
+1 650.759.8995
jhendrickson@
agentissoftware.com

Daniel Galanti
Dir of Engineering, Agentis Software
Reconquista 865 - Piso 6
Buenos Aires, Argentina
+54.11.4317.6880
dgalanti@
agentissoftware.com

ABSTRACT
In this paper, we describe lessons learned in commercializing and implementing agent-based systems in real-world business situations. Most notably we discuss gains in programmer productivity achieved by using AdaptivEnterprise, a commercial IDE and J2EE runtime engine based on BDI agents. In a wide range of complex business applications, we show that the use of BDI technology incorporated within an enterprise-level architecture can improve overall developer productivity by an average 350%. For java coding alone, the increase in productivity was over 500%. In competitive bids for large scale enterprise applications, AdaptivEnterprise was able to reduce total cost of ownership between two- and three-fold. In addition, the agent approach allowed the business to change and extend solutions quickly and helped bridge the gap between the business side and IT development.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *multiagent systems*.

General Terms
Management, Economics

Keywords
Agentis, AdaptivEnterprise, Function Point Analysis

1. INTRODUCTION
Agentis Software offers a commercial environment and methodology, called AdaptivEnterprise, for implementing agent-oriented goal-seeking applications. To date, Agentis has either implemented or assisted customers in implementing 15 applications. Of these, 13 are currently in production and two are pending deployment.

AdaptivEnterprise is both an IDE for building agents as well as a Java runtime engine for deploying the agents in a scalable, reliable, and secure manner. AdaptivEnterprise deploys agents in a pure J2EE-environment and has been commercially available for several years.

2. MAJOR LESSONS LEARNED
Agentis has learned several major lessons in commercializing agent technology. They are:

- Agent technology is applicable to a wide range of industry problems;
- The primary business value of agent technology is reduced complexity;
- Agent technology provides compelling ROI and productivity improvement;
- Designing with agents is easier for business analysts than conventional technologies;
- An iterative methodology is critical to building successful Agent-Oriented Systems; and
- IT Programmers can easily make the transition from pure Object-Orientation to Agent-Orientation

3. AGENT TECHNOLOGY IS APPLICABLE TO A WIDE RANGE OF INDUSTRY PROBLEMS
Our solution, AdaptivEnterprise, is currently in use in a wide variety of applications across a range of industries including logistics, finance, energy, telephony, transportation, and housing administration.

The range of implemented applications includes:

- Datacenter compliance and monitoring (Sarbanes-Oxley Compliance)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

COPYRIGHT 2006 ACM 1-59593-303-4/06/0005...$5.00.

- Multi-jurisdictional tax calculations
- Global package tracking
- Housing allocation
- Service configuration
- Dynamic pricing
- Customs documentation cleansing and classification
- Document workflow

4. REDUCING COMPLEXITY: THE #1 VALUE OF AGENTS

The most common theme throughout these applications is that they are "hard" to build. All are complex and are not the typical transactional applications that corporate IT groups normally build. Many of these applications are more typically delivered as customizable packages by third-party vendors. However, these applications have complex requirements and enough "exceptions" to make implementing an off-the-shelf package unfeasible without significant customization efforts.

Another common theme of these applications is change. Requirements for these applications change very rapidly and in often complex ways. New services and pricing schemes are added, companies are merged, government regulations change, etc. The ability to be able to quickly add or change functionality is critical. Because major control logic is abstracted in an agent-based system, adding and changing functionality is very easy.

These applications are also event and exception driven. That is, the system needs to respond to ongoing events and exceptions. Because the response to an event can vary depending on the current state of the system, the loosely coupled nature of agent control logic (processes) makes event and exception processing much simpler to code.

Finally, speed to market has been critical in the applications we've been involved in. Given the mission critical nature of these projects, there are serious financial and business implications involved in getting these applications completed. The reason many of our customers adopted Agent technology is that they were looking for a faster and easier way to get these applications built. Some projects were awarded to Agentis because we promised to deliver the application quickly, not because the customer understood Agent-based systems.

5. MEASURING ROI FOR PROJECTS

While the technical reasons for using agents might be apparent for various situations, in business settings, Return on Investment (ROI) is a primary decision criterion. Businesses will take a chance on a technology through a proof-of-concept project (POC) without regard for ROI but, for real implementations, the ROI must be compelling. Agent technologies are not pervasive in businesses and are foreign to many IT departments. In addition, there is a good bit of confusion about what agents are.

Measuring true ROI is complicated—and can be expensive to measure in full detail. In addition, it is sometimes hard to isolate the ROI of agent technology itself because many factors go into the success and the total cost of a project, including software licenses, analysis and design, testing, GUI construction, server infrastructure, project management, consulting/labor costs, etc.

Likewise, one cannot compare the ROI of two projects and assume the one with the lowest ROI was the most successful. Projects differ in complexity, internal politics, purpose, stakeholders and, of course, core functionality. Even two projects with similar functionality can have vastly different ROIs because of factors external to the products/languages used, such as changing requirements, ineffective management, etc.

However imprecise ROI measurements tend to be, there are several generally accepted measures of a project's "success". The first is productivity—how productive was the IT staff at implementing the solution versus other technology choices or development teams? The second is cost—how much did the project cost versus alternative estimates?

Given the fact that the applications that agents are best at solving tend to be complex critical applications, the expected ROI should be high. One specific project resulted in a 173% ROI and is detailed at the end of this paper.

5.1 Estimating Agent Productivity with Function Point Analysis

Function Point Analysis (FPA) is a technique published in 1979 by Allan J. Albrecht to measure the "size" of a software application and predict the level of effort associated with building it.

A Function Point (FP), according to the International Function Point Users Group (IFPUG.org), is a unit of "system functionality" measured from a functional, or user, point of view. It is independent of the computer language, development methodology, technology or capability of the project team used to develop the application.

According to IFPUG:

- FPA can be used to size software applications accurately. Sizing is an important component in determining productivity (outputs/inputs).
- Function Points can be counted by different people, at different times, to obtain the same measure within a reasonable margin of error.
- Function Points can be used to determine whether a tool, a language, or an environment is more productive when compared with others.

Because FPA is a fairly objective methodology for analyzing software complexity, it is useful for measuring

the productivity of differing software teams, methodologies, frameworks, and languages.

One of Agentis' customers, a large international logistics firm, uses FPA as a standard for measuring projects and IT productivity. Recently, they performed an analysis of their Agent-Oriented projects to determine whether to increase the role of Agents in the organization.

Based on six applications encompassing a total of 7,356 function points (FP) and 10 person-years of development, AdaptivEnterprise delivered an average of 2.11 function points per day for each fully completed project. Average productivity for all other non-agent-based Java projects within the organization is 0.45 FP per day.

Based on this analysis, agent-oriented methodology, architecture, and development delivered a 368% improvement in overall project productivity. (2.11 FP Actual/0.45 FP Expected)

Improvements on specific projects ranged from a high of 513% improvement to a low of 273% improvement. The largest single project had 3,850 FP and the smallest project had 304 FP.

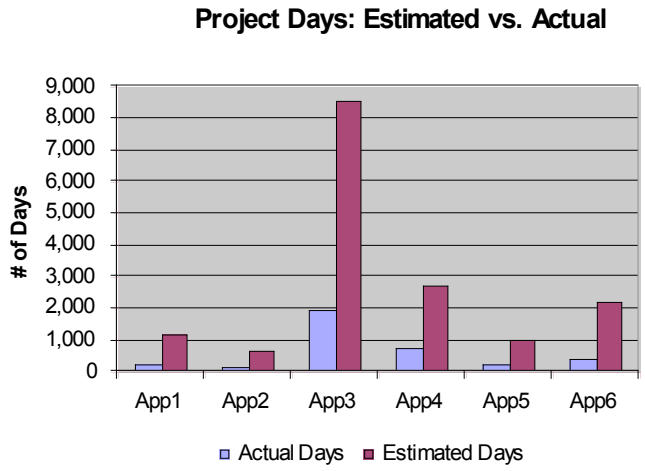


Figure 1: Estimated vs. Actual Project Days

5.2 Agent-Orientation's Effect on Coding

All the FP numbers presented to this point have been for the complete project—which includes requirements gathering, analysis and design, testing, and implementation. When Agent-oriented development is compared at a pure coding level, the productivity improvements are even greater.

Since AdaptivEnterprise is a 100% pure Java and J2EE solution, we compared the agent-based applications against other Java applications in the organizations. The historical average FP coding productivity for the organization's Java developers is 0.95 FP/day.

We only have coding metrics for four of the projects and their mean coding productivity with AdaptivEnterprise was 4.97FP/day. This represents a 423% improvement (4.97FP / 0.95 Average Java FP) over the standard for the organization.

Finally, a non-FP related data point is provided by an application for another customer involving a migration of an existing COBOL workflow application with 120,000 lines of code. The application was re-engineered into a Java-based system with seven agents in less than three months with two full-time and one part-time developer.

6. DESIGNING WITH AGENTS IS EASIER FOR BUSINESS ANALYSTS

How do you take a business process and map it into software? This is a central problem that business is still striving to solve in application development. Translating requirements into a working system is as much a science as it is an art. Humans who work with business processes speak one language while programmers and systems analysts speak another. IT staff are much more concerned with wiring specific systems and functions together to form a specific process. Business analysts describe roles, patterns, workflows, and decision trees.

We have found that by discussing systems in terms of data, goals, and plans, the conceptual modeling of systems becomes much easier. First, the terminology is easier for both IT and business to understand. Agent analysis lends itself very well to iterative requirements gathering. Just as it is easy to code new plans to an existing Agent-based application, it is also easier to add new plans during the conceptual phase of an application. Likewise, moving from the conceptual phase of analysis to the concrete design phase is aided because agent-oriented analysis produces goals and plans that can then be specifically designed and coded as discrete work elements.

At Agentis, we've used a modified version of the Rational Unified Process along with agile development techniques such as Scrum or XP. As pointed out previously, time to market is important with these applications. Because of that, a classic waterfall or even a strict RUP-based UML modeling approach is not always possible or desirable for these projects.

6.1 Agility Is Critical

A typical situation that we encountered was a rapid escalation in the problem space. Many organizations that are introduced to Agents for the first time are a bit skeptical. To mitigate any possible risks involved with bringing Agent-technology on board, many projects start as smaller proofs-of-concept and are quickly escalated into full-blown formal projects once Agent technology and concepts are proven.

Besides being able to tackle complex problems, Agentis' experience is also one of agility. Agent-based approaches allow us to respond very quickly to changes in the

requirements of a system and to changes to systems currently in production. For example, the application that covers 3,850 function points started life as a 350+ function point application. The benefits of the system were immediately seen by the customer and the application quickly escalated into a major, mission-critical endeavor. Because the BDI agent-model scales easily, it was fairly straightforward to move the system from a small effort to a large one—more agents were added along with more goals and more plans.

Several customers have stated that agent-oriented methodology coupled with a GUI design environment for building agents facilitated RAD capabilities with their complex systems—something that was previously not thought possible, since RAD tends to be associated with smaller systems with more CRUD (Create, Read, Update, Delete) transaction functionality.

6.2 Agent Technology Must Play Well With Others

Both the methodology as well as the actual implementation of an agent-based system must "play well with others". At minimum, a system will need to interact with an existing relational (SQL) database—most likely by storing long running state & transaction information.

Many organizations are moving toward Service Oriented Architecture (SOA) by implementing standard language and platform neutral mechanisms for exposing computing resources through WSDL or SOAP. At minimum, the agent system must be able to consume SOAP-based web services and provide agent-based web services to other systems via web services standards.

Likewise, organizations are also starting to use BPEL as a process model definition and execution environment. Agents must be able to call BPEL processes and BPEL processes must be able to invoke agents.

Having an Agent technology based on J2EE or .NET mitigates most of these integration concerns at a code level. Having GUI and IDE support for these interactions is critical for productivity.

6.3 Agent Technology Can Be Used By IT Programmers

Agentis normally helps customers build their first application. In cases of large applications, a third-party integrator might be brought in to provide programming resources. Projects are normally led by Agentis staff with extensive agent experience. However, most of the coding is done by "normal" Java developers. Agentis has trained over 100 developers in the AdaptivEnterprise product and all of our customers have taken over the maintenance of their applications with several of them continuing to build new applications with no direct input from Agentis.

Typical developer training is three days of classroom training on the product and 1½ days on the methodology. We have added concept courses for the Business Analyst and Architects. Developers can be productive after the training and proficient in 4 to 6 weeks of project work.

7. CASE STUDIES

7.1 Multi-Jurisdiction Tax Application for a Large Energy Company

Collection of sales and use tax is complicated because each city, municipality, county, and state has their own taxation policies. In addition, many large corporations are given tax breaks and have specific agreements in place with various government agencies covering the types of items that are taxable and at which special rate. Finally, because the customer in this case is an energy utility, many special taxation rules apply that are above and beyond typical use tax calculations.

The customer had acquired an existing tax package and had spent almost 18 months of time and several millions of dollars implementing and customizing the application. However, the tax package was unable to provide accurate tax information because of the complexities involved and required heavy user intervention, resulting in large amounts of overtime pay spent to ensure the tax calculations were correct. Agentis was brought in to help fix the problem.

The customer was attracted to the speed of development that Agentis promised as well as the goal and plan nature of the agent architecture which was easy for the customer to understand. Likewise, speed of application maintenance and change was critical. Tax laws and regulations change constantly across many tax jurisdictions. The agent-oriented application allows them to add and modify tax calculation plans very easily.

Between overtime spending, maintenance spending on the discontinued tax application, and ability to meet quality of service requirements of the government, the customer saved just under $6 million in direct costs and administrative revenue for collecting & administrating the taxes.

ROI for Tax Application	
Direct Savings & Revenue Stream	$5,812,000
Project Costs	$ 1,171,000
Useful Life	5 Years
Net Present Value	$ 2,005,555
ROI	173.06%

Overall the ROI on the project was calculated at 173% over five years resulting in an internal rate of return of 43%. In

addition to the cost savings, the customer can now offer value-added services to their customers such as tax reduction strategies, etc. In addition, the new system provides an audit trail for additional Sarbanes-Oxley (SOX) compliance.

The project was so successful that the customer has contacted Agentis on how the two companies can become closer strategic partners for further application development.

7.2 Global Package Tracking, Tracing, and Recovery

The customer is a global logistics and shipping company handling billions of packages a year. As the organization grows and acquires other logistics firms, a global solution was needed to handle package tracking, tracing, and recovery. As packages move through the system, millions of events are generated which change the state of the system and of individual shipments.

The application was designed to handle up to 7,500 concurrent users over 100 countries and needed to handle over 10,000 traces per day. A trace is a physical trace of a shipment—not just a check on 'where is my package'. Many times a trace results in people physically looking for a shipment or entering details about the shipment and its current status: examples include being detained for customs, weather delays, transit strikes, etc. Because of this, human intervention is usually required in the process as well as timely communication to the customer on the shipment status. To support this complex workflow scenario, the system supports the following:

- Role-based and team-based workflow;
- Integration to existing enterprise services for authorization and authentication, shipment searching, customer location and service level agreements;
- Real-time subscription to changes in physical location and workflow state;
- Automatic workflow task allocation to most suitable Customer Service Representative (CSR) and most suitable team; and
- Extensive workflow monitoring: notifications, alerts, reminders, state change markers.

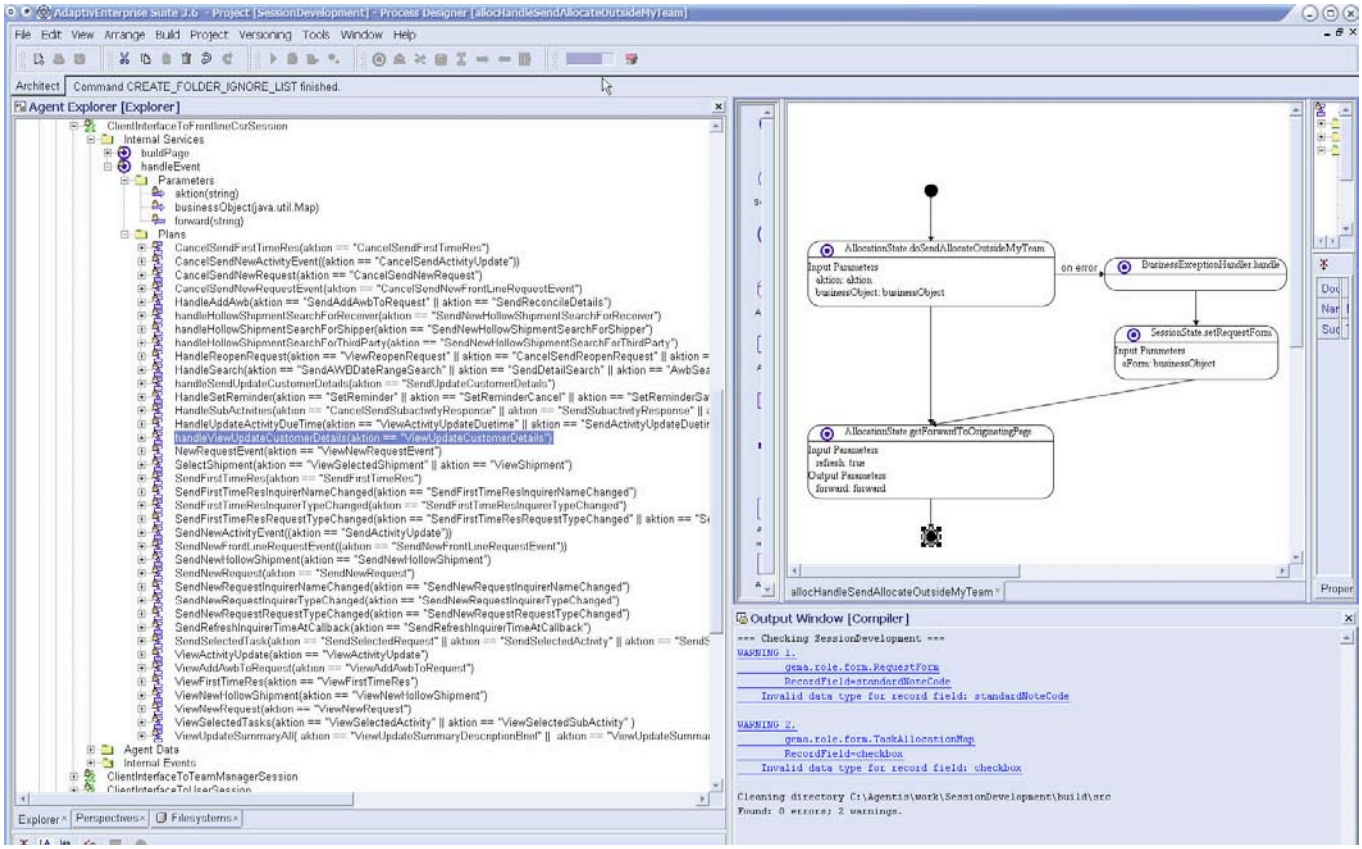


Figure 2: An event-handling agent with 37 possible executable plans.

The system is currently deployed to over 30 CPUs serving almost 3,000 concurrent users.

Business benefits, as stated by the customer, are:

- Reduced cost of operations because of automation of tasks and increased call center throughput;
- Enterprise-level adaptability because of agent reuse and flexibility and SOA implementation; and
- New platform for customer self-service as well as new service delivery channels

The reasons the customer chose Agentis/Agent Technology were:

- Full BDI/Goal Directed Agent Framework
  - 100% J2EE solution; use of EJBs, MDBs, JMS
  - Both synchronous and asynchronous inter-agent communication
- Extreme adaptability through Agents
  - Low lifecycle costs: maintenance is adaptation
  - Unprecedented application longevity
- Agents are EJBs
  - Enterprise friendly and very scalable
- Visual Agent Development Environment
  - Oriented toward Business Analysts
  - Comprehensive develop/test/deploy environment

The system was competitively bid with the customer. The total implementation costs to date are roughly $6MM. Other competitive solutions using a combination of packaged software along with customization had a estimated total ownership cost of $15MM to $25MM. The customer was able to get the functionality that they wanted in half the time they wanted at less than half the cost.

8. Summary

Agent technology is ready for mainstream corporate use and can be successfully commercialized. It has proven itself to be highly reliable and scalable and also well suited for complex business applications involving many moving parts, fast changing requirements, and tight timelines. Agent-based solutions have been used many times to solve problems customers thought were not solvable—at least not within any reasonable timeframe or budget constraints.

However, pure agent technology is not enough. The technology must be presented in a way that "typical" programmers can understand and use—and it must integrate into their current development efforts.

Likewise, a proven methodology that addresses the needs of business analysts as well as developers is also required.

ucing
ystem
.

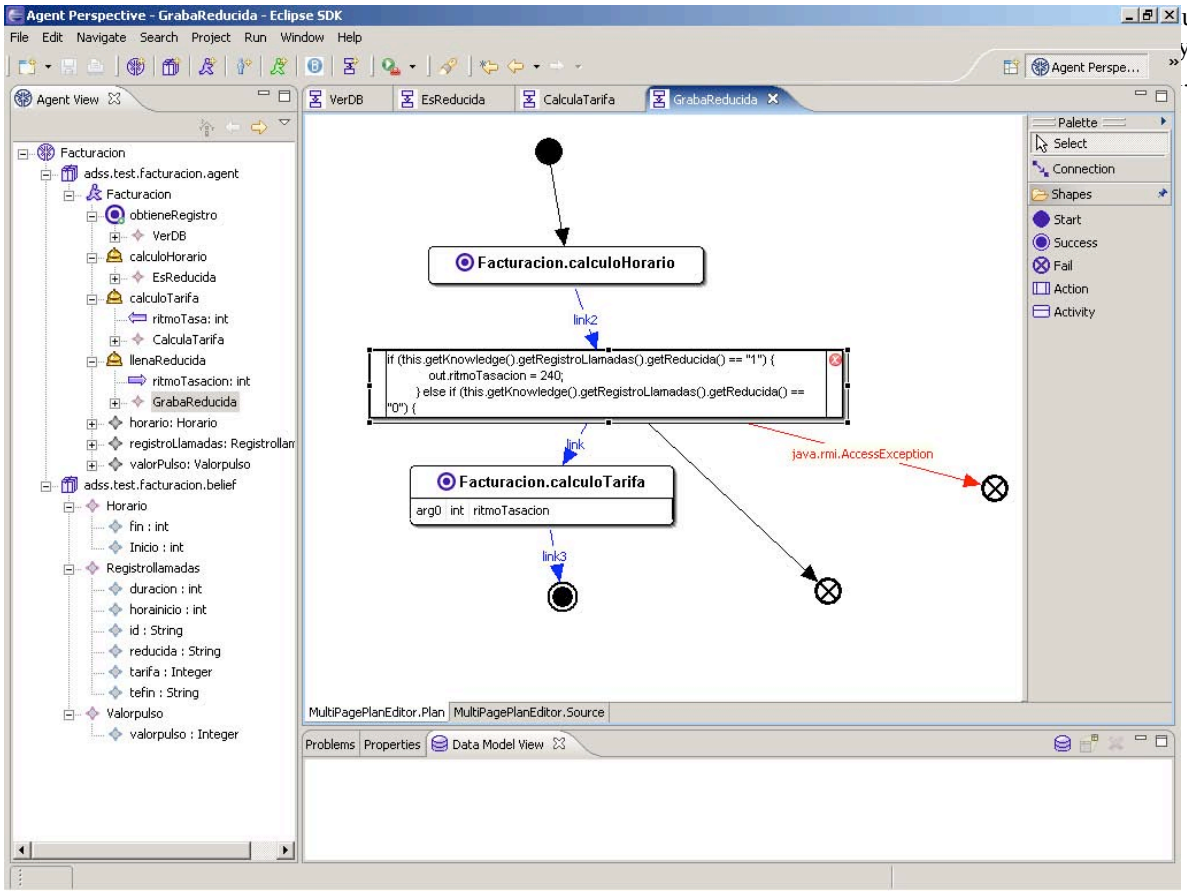


Figure 3: Plans are modeled with UML.

Simulation and modeling

Agent Interface Enhancement: Making Multiagent Graphical Models Accessible

Yang Xiang
University of Guelph, Canada
yxiang@cis.uoguelph.ca

Kun Zhang
University of Guelph, Canada
zhangk@uoguelph.ca

ABSTRACT

Multiagent probabilistic reasoning with multiply sectioned Bayesian networks requires interfacing agent subnets (the modeling task) subject to a set of conditions. To specify the interfaces such that they are both valid and efficient is non-trivial. We present multiagent interface enhancement that relieves agent designers from the burden of model efficiency so that multiagent graphical models become more accessible.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms

Keywords

Agent-Based Modeling, Cooperation and Coordination Among Agents, Graphical Models, MSBNs, Multiagent Architectures, Uncertain Reasoning

1. INTRODUCTION

Built upon the success of Bayesian Networks (BNs) [4], multiply sectioned Bayesian networks (MSBNs) [7] provide a rigorous framework for reasoning about uncertain domains in cooperative multiagent systems (MASs). Each agent holds its partial perspective (a subnet over a subdomain) of a domain, reasons autonomously as well as through limited communication with other agents. From a few high level requirements, (1) exact probabilistic measure of agent belief, (2) agent communication by belief over small sets of shared variables, (3) a simpler agent organization, (4) DAG domain structuring, and (5) joint belief admitting agents' belief on private variables and combining their beliefs on shared variables, it has been shown [9] that the resultant representation of a cooperative multiagent system is an MSBN.

To ensure autonomous, exact and efficient inference, subnets in an MSBN must satisfy a set of representational constraints. This is the modeling task. Specification of such a multiagent graphical model that is both valid and efficient is non-trivial and places a major burden on MAS designers. A key factor in modeling is the specification of agent interfaces, which are subject to hypertree, d-sepset, privacy, and sparse linkage tree conditions (to be introduced below). The main contribution of this work is a method to automate interface modeling so that MAS designers are relieved from the burden of model efficiency (the sparse linkage tree condition) and partially from the burden of privacy. The direct consequence is the simplification of modeling task and the improved accessibility of the MSBN framework to MAS designers.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. THE ENHANCEMENT APPROACH

Before agents can aid graphical modeling, the MAS has to be created. We assume that for any large domain where an MAS is to be built, there is a natural subdomain division reasonably obvious to the human agent designers. The natural division divides the domain into overlapping subdomains, one for each agent, and thus forms the initial agent interfaces. For instance, in equipment diagnosis [7], each subsystem forms a natural subdomain and its input from and output to another subsystem form the interface between the two. In supply chain collaborative design [8], each component to be designed by a supplier forms a natural subdomain. The initial interfaces, however, may not support efficient inference. We let agents cooperate to enhance interfaces to improve efficience. Not requiring initial interfaces to be efficient removes the burden from agent designers and eases their modeling task.

One of the advantages of the MSBN framework is that once the modeling of MSBN-based MAS is complete, internal details of each agent (its subnet) is kept private throughout model verification, model compilation, and run-time inference. As we are now dealing with multiagent modeling, it is necessary to relax this privacy constraint *slightly* (but not completely). Each subnet is built by its agent designer. We assume that the designer can classify variables in the subnet into three sets: private, public, and preferably private. The *private* set should be kept so absolutely. The *public* set is included in the initial agent interface. The *preferably private* set is initially private, but the agent is allowed to make some elements public if it believes that the disclosure improves inference efficience. The agent is required, however, to keep the disclosed subset as small as possible.

We argue that the three-set assumption respects privacy realistically. Consider tightened airport security check after 9/11. Travelers who are subject to much more thorough body search can be regarded as giving up more privacy than before. However, many accept this as a price paid for a safer flight. The three-set assumption treats privacy as a form of utility and efficience as another, and allows them to be traded as desired by agent designer.

Furthermore, consider what would happen if human agent designers, instead of agents, are trying to construct efficient agent interfaces. For a complex domain, it is unlikely for each designer to classify its subdomain into private and public variables, and the resultant agent interfaces will be automatically both valid and efficient. More likely, they would start with initial interfaces and incrementally modify them, which effectively discloses some initially private variables to other agent designers. The three-set assumption enables this incremental process to be mirrored by agents.

We are unaware of other work on improving agent interfaces for multiagent graphical models. The most related are the information bottleneck (IB) approach [6] and its extension [1]. IB constructs a set T of variables that are stochastic functions of one set Y of variables and provide information on another set X. In general, T *approximates* information in X. Our enhanced interface conveys *exactly* all relevant information from one agent to another.

3. OVERVIEW OF MSBNS

An MSBN M is a collection of Bayesian subnets, one from each agent, representing probabilistic dependence of a *domain* divided into multiple *subdomains*. Agents cooperate to reason about the state of the domain [7]. To ensure distributed, exact inference, subnets satisfy conditions described below:

Let $G_i = (N_i, E_i)$ $(i = 0, 1)$ be two graphs. G_0 and G_1 are *graph-consistent* if subgraphs of G_0 and G_1 spanned by $N_0 \cap N_1$ are identical. Given graph-consistent graphs $G_i = (N_i, E_i)$ $(i = 0, 1)$, the graph $G = (N_0 \cup N_1, E_0 \cup E_1)$ is the *union* of G_0 and G_1, denoted by $G = G_0 \sqcup G_1$. Given a graph $G = (N, E)$, a division of N into N_0 and N_1 such that $N_0 \cup N_1 = N$ and $N_0 \cap N_1 \neq \emptyset$, and subgraphs G_i of G spanned by N_i $(i = 0, 1)$, G is *sectioned* into G_0 and G_1. Sectioning is used to relate graphical models at individual agents in a cooperative multiagent system.

DEFINITION 1. *Let $G = (N, E)$ be a connected graph sectioned into subgraphs $\{G_i = (N_i, E_i)\}$. Let the subgraphs be organized into an undirected tree Ψ where each node is uniquely labeled by a G_i and each link between G_k and G_m is labeled by the non-empty* `interface` *$N_k \cap N_m$ such that for each i and j, $N_i \cap N_j$ is contained in each subgraph on the path between G_i and G_j in Ψ. Then Ψ is a* `hypertree` *over G. Each G_i is a* `hypernode` *and each interface is a* `hyperlink`*. A pair of hypernodes connected by a hyperlink is said to be* `adjacent`*.*

Each hyperlink is an agent *interface*. Agents communicate by exchanging beliefs over their interfaces. An interface must be a *d-sepset*:

DEFINITION 2. *Let G be a directed graph such that a hypertree over G exists. A node x contained in more than one subgraph with its parents $\pi(x)$ in G is a* `d-sepnode` *if there exists at least one subgraph that contains $\pi(x)$. An interface I is a* `d-sepset` *if every $x \in I$ is a d-sepnode.*

The overall structure of an MSBN is a hypertree MSDAG:

DEFINITION 3. *A* `hypertree MSDAG` *$\mathcal{G} = \bigsqcup_i G_i$, where each G_i is a DAG, is a connected DAG such that (1) there exists a hypertree Ψ over $\mathcal{G}$, and (2) each hyperlink in Ψ is a d-sepset.*

Graphically, a hyperlink separates the hypertree MSDAG into two subtrees. Semantically, this corresponds to conditional independence given the d-sepset. An MSBN is defined as follows with an example shown in Figure 1.

DEFINITION 4. *An MSBN M is a triplet $M = (\mathcal{N}, \mathcal{G}, \mathcal{P})$. $\mathcal{N} = \bigcup_i N_i$ is the* `domain` *where each N_i is a set of variables. $\mathcal{G} = \bigsqcup_i G_i$ (a hypertree MSDAG) is the* `structure` *where nodes of each DAG G_i are labeled by elements of N_i. Let x be a variable and $\pi(x)$ be all the parents of x in $\mathcal{G}$. For each x, exactly one of its occurrences (in a G_i containing $\{x\} \cup \pi(x)$) is assigned $P(x|\pi(x))$, and each occurrence in other DAGs is assigned a constant table. $\mathcal{P} = \prod_i P_i(N_i)$ is the* `jpd`*, where each $P_i(N_i)$ is the product of probability tables associated with nodes in G_i. Each triplet $S_i = (N_i, G_i, P_i)$ is called a* `subnet` *of M. Two subnets S_i and S_j are said to be* `adjacent` *if G_i and G_j are adjacent on the hypertree MSDAG.*

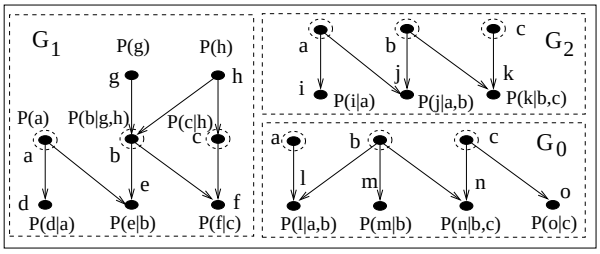


Figure 1: A trivial MSBN where each d-sepnode is shown with a dashed circle. The hypertree has the structure $G_1 - G_0 - G_2$ and each d-sepset is $\{a, b, c\}$.

To compute intra-agent messages effectively, each agent compiles its subnet into a junction tree (JT) [2], where variables are grouped into *clusters* with intersection of adjacent clusters referred to as *separators*. To integrate computation of intra-agent messages with computation of inter-agent messages seemlessly, the compilation is performed cooperatively. A key step is *cooperative triangulation* [7], during which each agent performs *constrained triangulation* through node elimination in partial order: A set N of nodes is eliminated according to a partial order (N_1, N_2) where $N = N_1 \cup N_2$, if nodes in N_1 are eliminated before nodes in N_2 are eliminated. We also refer to the triangulation as *constrained by* N_2, or *unconstrained* if $N_2 = \emptyset$.

For efficient computation of inter-agent messages, agents compile each d-sepset into a JT, called a *linkage tree*. See Figure 2 for linkage tree L_1 between T_0 and T_1. Each cluster in L_1 is called a *linkage*. Linkage $\{b, c\}$ is an message channel between cluster $\{b, c, f\}$ in T_1 and cluster $\{b, c, n\}$ in T_0. Its state space size (SSS) is 16 if the dimension (number of possible values) of variables a, b and c is 4, and that of L_1 is 32 which determines the efficience of inter-agent messaging. In general, the sparser the linkage tree (with many small linkages), the more efficient it is. If L_1 were made of a single linkage $\{a, b, c\}$, its SSS would be 64.

4. WHY NOT CENTRALIZE COMPILATION

An alternative to enhancing interfaces in MSBN is to build a centralized domain model, compile it into a JT, and separate the JT into subtrees one for each agent. Although it provides the most efficient run-time representation, it suffers from two major drawbacks compared to the approach we take.

First of all, the alternative requires that all domain variables and their dependence relations to be disclosed to the centralized model builder. There will be no agent privacy at all. In contrast, our approach does not disclose any private variables and dependence relations among them. It only discloses a subset of preferably private variables and does so in a cautious manner.

Secondly, even if the agent privacy were not a concern at all, the alternative will disrupt natural subdomain division among agents.

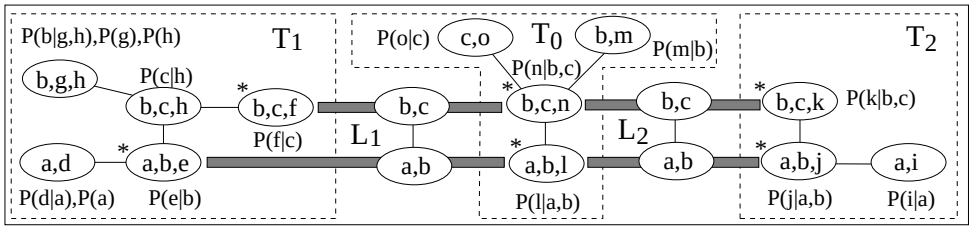


Figure 2: JTs and linkage trees obtained from Figure 1.

We illustrate this problem with a trivial MSBN shown in Figure 3. In (1), a trivial centralized domain model is shown as a BN. Sup-

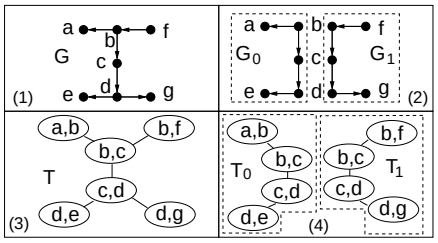


Figure 3: Illustration of natural subdomain disruption.

pose that it is naturally divided into two overlapping subdomains as shown in (2), where variables b, c and d form the interface. The BN in (1) is compiled into JT in (3). If this JT is to be separated into two, there are five possible ways, each separating the JT into two subtrees along one of the five separators (e.g., the separator between clusters $\{b, c\}$ and $\{b, f\}$ is $\{b\}$). None of them corresponds to the natural subdomains $\{a, b, c, d, e\}$ and $\{b, c, d, f, g\}$. The agents thus constructed lose their semantic coherence (the need for such coherence is also touched upon in [3]). On the other hand, according to the MSBN framework, the subdomain models in (2) will each be compiled into a JT in (4), which preserves the natural division while allowing efficient inference.

5. ANALYSIS

Inference in MSBNs is efficient mostly because agent interfaces, the d-sepsets, are represented as linkage trees, which enables potential factorization. Intuitively, this means that d-sepnodes are dispersed among a number of clusters in the local JT. Hence, the more scattered the d-sepnodes among clusters, the more efficient the inference. In the following, we analyze how adding more d-sepnodes can achieve such effect. Proposition 1 shows under what condition adding a node to d-sepset can disperse d-sepnodes into different clusters.

PROPOSITION 1. *Let S be the subnet of an agent, T be its local JT from S, and x and y be d-sepnodes contained in a cluster C in T. If a unique path exists between x and y in S, all other nodes on the path are non-d-sepnodes, and one of them, v, is head-to-tail or tail-to-tail, then adding v to d-sepset can disperse x and y into different clusters, unless it is prevented by other agents.*

Proof: Since the path contains at least v and it is head-to-tail or tail-to-tail, x and y will not be connected during moralization. Once v is added to the d-sepset, during constrained triangulation by the local agent, x can be eliminated before y and v, or y can be eliminated before x and v (neither is possible if v is not added to the d-sepset). In either case, x and y will not be connected by any fill-in. Hence, unless a moral link or a fill-in between x and y is added by another agent, x and y cannot be in the same cluster in the new local JT. □

The only way the condition in Proposition 1 can be violated is where the unique path between x and y consists of a single head-to-head node v. If any arc on the path is reversed or any additional nodes exist on the path, the condition in Proposition 1 will be true. Because a moral link will connect x and y, no matter what the rest of the d-sepset is, x and y must be contained in the same cluster. This means that Proposition 1 captures *all* possible cases where a non-d-sepnode can be successfully added to d-sepset to disperse x and y.

The following corollary generalizes Proposition 1 to the case where multiple paths exist between x and y in the local subnet.

COROLLARY 1. *Let S be the subnet of an agent, T be its local JT from S, and x and y be d-sepnodes contained in a cluster C in T. For each path between x and y in S, if all other nodes on the path are non-d-sepnodes, and one of them, v, is head-to-tail or tail-to-tail, then adding such v from each path to d-sepset can disperse x and y into different clusters, unless it is prevented by other agents.*

When d-sepnodes are dispersed into different clusters, they are effectively dispersed into different linkages in the linkage tree. On one hand, this reduces sizes of corresponding linkages. As individual linkages become smaller, the state space of the linkage tree is reduced and hence the improved efficience. On the other hand, according to Corollary 1, dispersing d-sepnodes requires enlarging the d-sepset. The new d-sepnodes increase the state space in each linkage they participate. To achieve the net effect of reduction in linkage tree state space, new d-sepnodes must be chosen carefully.

In general, when addition of a single non-d-sepnode disperses multiple pairs of d-sepnodes, the negative effect of the addition is smaller than its positive effect and the linkage tree state space is decreased.

The above analysis has emphasized the topological aspect of the issue. However, since it is the total state space of the linkage tree that directly affects inference efficience, not cardinalities of linkages, another important factor is the dimensions of variables involved. When a linkage of three variables each of dimension 3 is dispersed, by adding a binary variable v to d-sepset, into three linkages each made of v and another variable (see Figure 5 (d) later for an example), SSS is reduced from 3*3*3 = 27 to 3*2*3=18. However, if v has a dimension of 4, SSS is increased to 3*4*3 = 36.

In general, when a new d-sepnode v disperses existing d-sepnodes whose dimensions are higher than that of v, the linkage tree SSS is decreased.

6. SEARCH FOR ENHANCEMENT

To improve interface efficience, we add non-d-sepnodes to a d-sepset to make the linkage tree sparser and reduce its SSS. We refer to a non-empty set of non-d-sepnodes in a subdomain as an *enhancement*, whose effectiveness depends also on enhancements from other agents. Hence, the optimal set of enhancements (two

per d-sepset), which reduces the sum of all linkage tree state space sizes to the maximal extent, generally can only be obtained by exhaustive multiagent search. If each agent has $O(k)$ (where k is very large as shown below) alternative enhancements for each of its d-sepsets, then the search space of n agents has a complexity of $O(k^{2n})$. In the following, we consider several approximations with reduced complexity.

1. Each pair of adjacent agents performs cooperative search over the space of all pairs of enhancements (one from each agent), and then adds the pairwise top enhancement to their d-sepset.
2. Each agent searches for local top enhancements (there may be ties). Cooperative search is then performed by all agents over the space of all combinations of local top choices.
3. Each agent searches for local top enhancements. Then each pair of adjacent agents searches for pairwise top enhancements over the space of all pairs of local top choices.

The first approximation restricts cooperative search to pairwise. The computation has a complexity of $O(n\ k^2)$. The second approximation first reduces the number of alternative enhancements at each agent per each d-sepset from k to a few, say, k'. Then global cooperative search is performed with $O(k'^{2}\ ^{n})$ complexity. The third approximation has the lowest complexity of $O(n\ (k + k'^2))$. In this work, we study the effectiveness of this approximation and will investigate the other approximations in future research.

The top level algorithm below is run by the MAS coordinator to activate cooperative interface enhancement.

ALGORITHM 1 (COENHANCEINTERFACE).
choose an agent A_* arbitrarily;
call A_* to run EnhanceInterface with null parameters;

The following recursive algorithm is run by each agent. Without losing generality, we denote the agent executing the algorithm as A_0. The execution is activated by a caller, denoted as A_c, which is either an adjacent agent of A_0 or the MAS coordinator. If A_0 has additional adjacent agents, they are denoted $A_1, ..., A_m$. The subdomains of $A_c, A_0, ..., A_m$ are $N_c, N_0, ..., N_m$, respectively, their subnets are $G_c, G_0, ..., G_m$, the interface between A_0 and A_c (if an agent) is denoted D_c, and the interfaces between A_0 and $A_1, ..., A_m$ are denoted $D_1, ..., D_m$. The caller A_c passes two parameters into A_0: $NewD_c$ - a set of new d-sepnodes and $Arcs_c$ - a set of arcs (each either connecting elements in $NewD_c$ or connecting an element of $NewD_c$ with an existing d-sepnode). In return, A_0 will send parameters $NewD_c^{\rightarrow}$ and $Arcs_c^{\rightarrow}$ to A_c, where the arrow denotes output by A_0.

ALGORITHM 2 (ENHANCEINTERFACE). *When A_0 is called by A_c with parameters $NewD_c$ and $Arcs_c$, it does the following:*

compile G_0 into JT T_0^* by unconstrained triangulation;
for $i = 1, ..., m$ and $i = c$ if A_c is an agent, do
  compile G_0 into JT T_i through triangulation constrained by $(N_0 \setminus D_i, D_i)$;
  perform $UpdateDsepset(G_0, D_i, T_i, T_0^*)$ and denote return value by $NewD_i^{\rightarrow}$;
  denote arcs in G_0 relevant to $NewD_i^{\rightarrow}$ as $Arcs_i^{\rightarrow}$;
for $i = 1, ..., m$, do
  call A_i to EnhanceInterface with $NewD_i^{\rightarrow}$ and $Arcs_i^{\rightarrow}$;
  receive $NewD_i$ and $Arcs_i$ from A_i;
for $i = 1, ..., m$ and $i = c$ if caller is an agent, do
  enhance D_i with $NewD_i$ and $NewD_i^{\rightarrow}$ if both are non-empty;
  update G_0 with $NewD_i$ and $Arcs_i$;
if A_c is an agent, send $NewD_c^{\rightarrow}$ and $Arcs_c^{\rightarrow}$ to A_c;

EnhanceInterface consists of three *for* loops. In the first *for* loop, local subnet is compiled into a JT by constrained triangulation relative to each d-sepset. Using this JT and that obtained from unconstrained triangulation, the local top d-sepset enhancement is obtained by UpdateDsepset (see below for more on this step). Through the second *for* loop, the enhancement operation is propagated along the hypertree organization of the MAS, until it reaches agents located at the leaves of the hypertree. The results of local search by UpdateDsepset will then be propagated back towards A_*, through the last statement of the algorithm. In the third *for* loop, the agent enhances the d-sepsets and its local subnet.

The return value $NewD_i^{\rightarrow}$ from local search UpdateDsepset can contain an unique top enhancement. It can also contain a set of top enhancements (that tie). In that case, the search strategy 3 dictates a pairwise cooperative search, which can be performed by modifying the above algorithm as follows:

It may appear that the task to decide which pair of local top enhancements are pairwisely best requires cooperation of both agents. We show below that a simpler operation suffices.

We consider agent A_0 and its caller agent A_c. At the end of the first *for* loop of EnhanceInterface, A_0 has received $NewD_c$ and computed $NewD_c^{\rightarrow}$. For each enhancement in $NewD_c$ and each in $NewD_c^{\rightarrow}$, A_0 can perform constrained triangulation using the corresponding enhanced d-sepset. A linkage tree will then be derived. According to [7] [1], this linkage tree is equivalent to what would be derived by A_c. Hence, the best pair of enhancements can be computed by A_0 only, with the following statements added at the end of the first *for* loop:

search for best pair of enhancements with A_c based on $NewD_c$ and $NewD_c^{\rightarrow}$, breaking ties arbitrarily;
update $NewD_c$ and $NewD_c^{\rightarrow}$ accordingly by deleting other enhancements;

To inform A_c of the result, the last statement of the algorithm needs to be modified into the following:

if A_c is an agent, send $NewD_c^{\rightarrow}$ and $Arcs_c^{\rightarrow}$ to A_c as well as the pairwisely best enhancement in $NewD_c$;

7. LOCAL HEURISTIC SEARCH

For an agent to search for the locally best enhancement to a d-sepset, the alternatives must be identified. This can be done by examining the local subnet or the local JT. Since linkage tree is generated from local JT and preserves conditional independence encoded in the JT (Proposition 7.6 [7]), we identify enhancements based on the local JT.

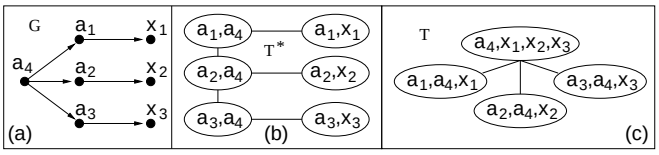


Figure 4: (a) Subnet G. (b) The JT T^* from G by unconstrained triangulation. (c) The JT T from triangulation constrained by $\{x_1, x_2, x_3\}$.

Consider the trivial subnet in Figure 4 (a). Given the d-sepset $\{x_1, x_2, x_3\}$, the local JT in (c) can be constructed. From the local JT, the linkage tree can be derived that has a single linkage $\{x_1, x_2, x_3\}$.

[1] Section 7.6.2 on two-agent cooperative triangulation and Section 8.3 on equivalence of linkage trees derived by adjacent agents.

To identify d-sepset enhancements, we consider JT T^* in (b) obtained by unconstrained triangulation. It preserves all conditional independence relations in (a) while T in (c) does not. Hence, T^* provides hints as to what conditional independence relations of (a) that has been removed in (c) can be utilized to improve the efficience of the d-sepset and the resultant linkage tree.

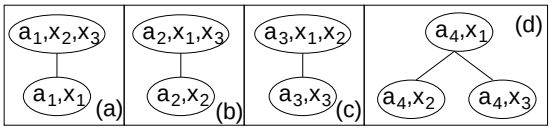


Figure 5: (a) The linkage tree obtained by adding a_1 to the d-sepset. (b) Obtained by adding a_2. (c) Obtained by adding a_3. (d) Obtained by adding a_4.

For example, $\{a_1\}$ is a separator in T^* between clusters $\{a_1, a_4\}$ and $\{a_1, x_1\}$, and it separates the cluster that contains the d-sepnode x_1 from clusters that contain the rest of the d-sepset. This suggests that $\{a_1\}$ is an alternative enhancement. After adding it to the d-sepset, the new linkage tree is shown in Figure 5 (a), where x_1 is indeed dispersed into a different linkage from the rest of d-sepset.

Following the same idea, $\{a_2\}$, $\{a_3\}$, $\{a_4\}$ are each an alternative enhancement. The corresponding linkage trees are shown in Figure 5 (b), (c) and (d), respectively. We define alternative enhancement formally below.

DEFINITION 5. *Let G be a subnet over N, D be its d-sepset with an adjacent subnet, and T^* be a JT of G obtained from unconstrained triangulation. An* `alternative enhancement` *E of D is the union of a set (possibly empty) of separators of T^* such that $E \cap D = \emptyset$.*

Given T^* with q clusters, in general, it has $O(2^q)$ enhancements, where q is bounded by the number of subdomain variables. We explore two techniques to improve computational efficience: leaf cluster removal and topological dominance. Below, we present the local search algorithm and then analyze two techniques. All d-sepnodes and non-d-sepnodes are relative to the d-sepset D.

ALGORITHM 3 (UPDATEDSEPSET).

Input: a subnet G over N and its d-sepset D with an adjacent subnet; a JT T from G by constrained triangulation with partial order $(N \setminus D, D)$; a JT T^* obtained from G by unconstrained triangulation;

perform the following recursively until no cluster can be deleted,
  for each leaf cluster Q of T^*,
    if Q contains no d-sepnodes, delete Q from T^*;
    else if Q is a leaf cluster in T with the same separator as in T^*,
      delete Q from T^*;
if T contains no separator with preferably private nodes only,*
  return $\emptyset$;

CanNewD = empty;
for each separator W of T with preferably private nodes only, do*
  add $\{W\}$ to CanNewD as an 1-separator member;
  copy T to T';*
  delete each occurrence of W in T' to split it into subtrees;
  if each subtree has a single cluster with d-sepnodes,
    label W as final;

k=1, more=true;
while more == true, do
  more = false;
  for each k-separator member M of CanNewD, do
    copy T to T';*
    for each separator R in M, do
      delete each occurrence of R in T' to split it into subtrees;
    if each subtree has a single cluster with d-sepnodes,
      label M as final;
    else
      for each 1-separator member $\{W\}$ of CanNewD, non-final,
          $W \notin M$, and $M \cup \{W\} \notin CanNewD$, do
        if there exists a subtree where W is a separator on the path
            between 2 clusters with d-sepnodes, do
          add $M \cup \{W\}$ to CanNewD as a k+1-separator member;
          more = true;
  k++;

for each member M of CanNewD, do
  perform constrained triangulation with M added to d-sepset;
  compute SSS α of the resultant linkage tree;
find member M^* of CanNewD with the smallest α value α^*;
β = SSS of the linkage tree derived from T;
if $\beta - \alpha^*$ is large than a threshold, return M^*;
else return $\emptyset$;

UpdateDsepset consists of four segments (separated by blink lines). The first removes leaf clusters in T^*. Proposition 2 below shows that these clusters are irrelevant and hence their removal reduces the space of alternative enhancements. The analysis is based on h-separation [7] in JTs:

DEFINITION 6. *Let H be a JT over N, and X, Y, Z be disjoint subsets of N such that no $x \in X$ and $y \in Y$ are contained in the same cluster in H. For $x \in X$ contained in cluster Q_x and $y \in Y$ contained in cluster Q_y, x and y are h-separated by Z if there exists a separator $S \subseteq Z$ on the path between Q_x and Q_y. X and Y are h-separated by Z if for every $x \in X$ and $y \in Y$, x and y are h-separated by Z.*

For instance, in Figure 4 (2), $\{x_1\}$ is h-separated from $\{x_2, x_3\}$ by $\{a_1\}$, written as $< x_1|a_1|x_2, x_3 >_{T^*}$ or simply $< x_1|a_1|x_2, x_3 >$. We are interested in h-separation relations $< X|Z|Y >$ relative to a given d-sepset D, where $X \subset D$, $Y \subset D$ and $Z \cap D = \emptyset$. We refer to such a relation as a *h-separation relative to D*.

PROPOSITION 2. *Let N be the domain of T^* and E be an alternative enhancement containing separators in T^*. After recursive deletion (relative to d-sepset D) of leaf clusters as specified in UpdateDsepset, let the resultant JT be T' whose domain is $N' \subset N$ and $E' = E \cap N' \subset E$. Let H denote the set of h-separations relative to D implied by E and H' be that implied by E'. Then $H = H'$.*

Proof: Intuitively, E and E' are the same except that E contains additional separators in T^* that are deleted by UpdateDsepset. We show that leaf cluster deletion by UpdateDsepset does not remove h-separations implied by E.

Let Q be a leaf cluster in T^* with separator V. We denote $Q \setminus V$ by C. The only h-separations relative to D that Q represents are in the form of $< C^-|V|U^- >$, where C^- is any subset of C and U^- is any subset of $N \setminus Q$.

The h-separations relative to D implied by E are in the form $< X|Z|Y >$ where $X \subset D$, $Y \subset D$, and $Z \cap D = \emptyset$. If Q contains no d-sepnodes, then C^- contains no d-sepnodes. Hence, removal of Q cannot remove any h-separations implied by E.

Next, consider the case where Q contains d-sepnodes. If Q is a leaf cluster in T (as defined in UpdateDsepset) and its separator in T is also V, then the h-separations represented by Q are also true in T. This means that these relations are not implied by E. Hence, removal of Q cannot remove any h-separations implied by E.

Since no h-relations in H is removed by leaf cluster deletion in UpdateDsepset, we obtain $H = H'$. □

Proposition 2 implies that leaf cluster deletion by UpdateDsepset does not introduce approximation to local search while improving its efficience. Proposition 3 below establishes that the early termination of UpdateDsepset is a correct action.

PROPOSITION 3. *If UpdateDsepset returns $\emptyset$ at the end of first segment, then the d-sepset D cannot be improved.*

Proof: When $T*$ contains no separator with preferably private nodes only, no legal separator Z can be found such that $< X|Z|Y >$ holds, where $X \subset D$, $Y \subset D$, and $Z \cap D = \emptyset$. □

8. TOPOLOGICAL DOMINANCE

The second and third segments of UpdateDsepset apply the technique of topological dominance or t-dominance to further trim the local search space. The concept of dominance has been used in game-theoretic research, e.g., [5]. We apply here from a topological perspective. In Figure 5 (d), each original d-sepnode has been isolated in a separate linkage. This suggests that adding more nodes to d-sepset may enlarge the linkage tree SSS. In fact, adding to $\{a_4\}$ any variable of a dimension no less than that of a_4 will do poor than $\{a_4\}$. We say that any such superset is *t-dominated* by $\{a_4\}$.

On the other hand, the enhancement $\{a_1\}$ does not t-dominate every proper superset. For instance, in the linkage tree produced from $\{a_1\}$ (Figure 5 (a)), x_2 and x_3 are contained in the same linkage. In the linkage tree produced from the superset $\{a_1, a_2\}$ (Figure 6 (a)), x_2 and x_3 are dispersed into different linkages. Hence, enhancement $\{a_1, a_2\}$ is not t-dominated by $\{a_1\}$.

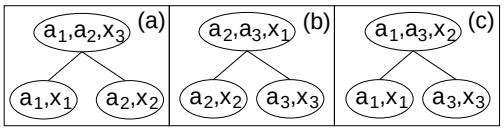


Figure 6: (a) The linkage tree obtained by adding $\{a_1, a_2\}$ to the d-sepset. (b) Obtained by adding $\{a_2, a_3\}$. (c) Obtained by adding $\{a_1, a_3\}$.

If an enhancement is not t-dominated by any proper subset, then it is called a *non-t-dominated* enhancement. For instance, $\{a_1, a_2\}$ is an non-t-dominated enhancement. If an non-t-dominated enhancement t-dominates every proper superset, it is called a *final* enhancement. Enhancement $\{a_4\}$ is final, but $\{a_1\}$ is not final.

Figure 6 shows linkage trees from three alternative enhancements each of which is obtained by including two separators in Figure 4 (b). All of them are non-t-dominated and final.

DEFINITION 7. *Let E be an alternative enhancement of a d-sepset D and H be the set of h-separations relative to D implied by E. Let $E' \supset E$ be another enhancement and H' the corresponding h-separations. E' is* $t - \texttt{dominated}$ *by E, if for each h-separation $< X|Z'|Y >$ in H', there exists $< X|Z|Y >$ in H, such that $Z' \supseteq Z$. If there exists no $E' \subset E$ such that E' t-dominates E, then E is* $\texttt{non} - \texttt{t} - \texttt{dominated}$. *If for all $E' \supset E$, E' is t-dominated by E, then E is* `final`.

Given the four separators in Figure 4 (b), there are a total of 15 alternative enhancements. Linkage trees from seven are shown in Figs. 5 and 6. The remaining eight are t-dominated.

As assumed, each agent classifies subdomain variables into private, public, and preferably private. For instance, the agent could classify a_4 as private, but a_1, a_2, a_3 as preferably private. This classification trims the set of non-t-dominated enhancements. In particular, only enhancements corresponding to Figure 5 (a), (b), (c), and Figure 6 (a), (b), (c) are now valid. Proposition 4 below shows under what condition, ignoring t-dominated enhancements causes no loss of accuracy to local search. We omit proofs for the remaining formal results due to space.

PROPOSITION 4. *Let AE be the set of all alternative enhancements of a d-sepset D for a given subnet. Let ND be the set of non-t-dominated elements of AE. Let TOP_{AE} be the set of elements of AE whose linkage tree state space sizes are minimal among elements of AE, and TOP_{ND} be the corresponding set relative to ND. If dimensions of non-d-sepnodes are identical, then $TOP_{AE} = TOP_{ND}$.*

The second and third segments of UpdateDsepset compute non-t-dominant enhancements. According to Proposition 4, local search over these enhancements is only accurate when preferably private variables have identical dimensions. When this condition does not hold, local search restricted by t-dominance becomes a heristics and introduces approximation.

The second segment identifies 1-separator non-t-dominated enhancements. The third segment identifies non-t-dominated enhancements made of multiple separators. Note that the concept of final enhancement is utilized to skip t-dominated enhancements and improve efficience. The following proposition shows that *final* enhancements are correctly identified. It can be proven by a direct application of Def. 7.

PROPOSITION 5. *Let E be an enhancement and Ψ be the set of subtrees resultant from deletion of each separator of E in T'. If each subtree in Ψ has a single cluster with d-sepnodes, then E t-dominates every enhancement $E' \supset E$.*

The following theorem establishes the significance of final enhancements and functionality of the second and third segments.

THEOREM 1. *At the completion of the third segment of UpdateDsepset, $CanNewD$ contains all and only non-t-dominated enhancements relative to d-sepset D.*

After local search space has been reduced to non-dominated enhancements, can it be trimmed without introducing further approximation? The following example answers the question negatively.

For variables in Figure 4 (a), denote dimensions of $a_1, \cdots, a_4$ as $m_1, \cdots, m_4$, respectively, and those for $x_1, \cdots, x_3$ as $n_1, \cdots, n_3$, respectively. Denote by k_1 the SSS of linkage tree from constrainted triangulation with d-sepset $\{x_1, x_2, x_3\}$ (Figure 4 (c)). Denote the SSS of linkage trees shown in Figure 5 as k_2, k_3, k_4, k_5, respectively, and those in Figure 6 as k_6, k_7, k_8, respectively. Then $k_1, \cdots, k_8$ can be computed as follows:

$$k1 = n_1 * n_2 * n_3,$$
$$k2 = m_1 * (n_2 * n_3 + n_1), \quad k3 = m_2 * (n_1 * n_3 + n_2),$$
$$k4 = m_3 * (n_1 * n_2 + n_3), \quad k5 = m_4 * (n_1 + n_2 + n_3),$$
$$k6 = m_1 * m_2 * n_3 + m_1 * n_1 + m_2 * n_2,$$
$$k7 = m_2 * m_3 * n_1 + m_2 * n_2 + m_3 * n_3,$$
$$k8 = m_1 * m_3 * n_2 + m_1 * n_1 + m_3 * n_3$$

Table 1: State space sizes $k_1, \cdots, k_8$ given variable dimensions. Upper half: variable dimensions. Lower half: state space sizes.

n_1	2	3	3	3	5	5	7	7
n_2	2	3	3	3	5	6	6	4
n_3	2	3	3	3	5	5	5	5
m_1	2	2	3	3	2	2	4	3
m_2	2	3	2	3	3	3	2	4
m_3	2	3	3	2	3	3	3	2
m_4	2	3	3	3	3	4	5	5
k_1	8	27	27	27	75	150	210	140
k_2	12	24	36	36	60	70	148	81
k_3	12	36	24	36	90	93	82	156
k_4	12	36	36	24	90	105	141	66
k_5	12	27	27	27	45	64	90	80
k_6	16	33	33	45	55	58	80	97
k_7	16	45	33	33	75	78	69	82
k_8	16	33	45	33	55	61	115	55

Table 1 shows values for $k_1, \cdots, k_8$ given dimensions $n_1, \cdots, n_3$, $m_1, \cdots, m_4$. The minimal linkage tree SSS for each row occurs at a different column. For instance, for the first row (with all variables binary), the optimal size is 8 and is located at the first column in the right half. For the second row, the optimal size 24 is located at the second column, and so on.

This example demonstrates that any non-t-dominated enhancement may be the best choice, given right variable dimensions. In order not to cause further approximation, all non-t-dominated enhancements must be evaluated. This is done in the fourth segment. For simplicity, we phrased return value as a single enhancement, although it can be multiple enhancements that tie or top the list.

9. SUBNET CONNECTEDNESS

When an agent A passes a set of new d-sepnodes to an adjacent agent A', it is possible that some new d-sepnodes are disconnected in the updated subnet of A'. These new d-sepnodes are dependent of the rest of the subdomain of A' (since they are connected to other d-sepnodes in the subdomain of A). Hence, their disconnectedness in the updated subnet of A' creates misconception of independence. They violate convention of graphical models where connection implies dependence and disconnection implies independence.

To avoid the misconception, for each new d-sepnode v, we need to ensure that, in the updated subnet of A', it is connected to at least one existing d-sepnode. When v is not already connected to existing d-sepnodes, arcs must be added to achieve the connection. The arcs added will make the new subnet denser, but should not make it unnecessarily so, because any increase in the density of the subnet will translate into an increase in the size of state space of the resultant linkage tree. Based on these criteria, we identify the following subtasks:

First, agent A needs to recognize when some new d-sepnodes are disconnected to existing d-sepnodes. A new d-sepnode v is connected to existing d-sepnodes if (1) v is directly connected to an existing d-sepnode, or (2) v is indirectly connected an existing d-sepnode through a path consisting entirely of new d-sepnodes. This subtask can be accomplished with the following algorithm:

ALGORITHM 4 (MARKCONNECTED).

mark each existing d-sepnode;
perform the following recursively,
  for each marked node, do
    if it has any adjacent new d-sepnode v, mark v;

The following proposition shows that any new d-sepnode left unmarked by MarkConnected is disconnected from the existing d-sepset. Its proof is trivial.

PROPOSITION 6. *After MarkConnected is performed, if there exist unmarked new d-sepnodes, then each such node is disconnected from the existing d-sepnodes.*

Second, once a disconnected new d-sepnode v is identified, agent A needs to propose arc(s) to be added, that is, which other d-sepnodes should v connect to and direction of arcs. We show below that inadequate arcs added may create either directed cycle or significantly increase the size of state space of the local JT.

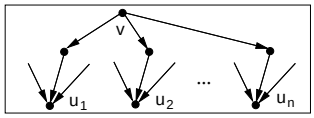


Figure 7: A subnet whose d-sepset with an adjacent subnet is $\{u_1, u_2, ..., u_n\}$.

In Figure 7, the existing d-sepset is $\{u_1, u_2, ..., u_n\}$. Enhancing the d-sepset with node v will significantly reduce the size of state space of the linkage tree. However, if v is added to the subdomain of the adjacent agent, it is disconnected. To make it connected to the existing d-sepset, it can be connected to a node u_i. If the arc (u_i, v) is added, it forms a directed cycle. On the other hand, if the arc (v, u_i) is added, the node u_i has one more parent. In the local JT, there is at least one cluster that contains u_i and all its parents. At least the state space of this cluster is enlarged by a factor of the dimension of v due to adding (v, u_i).

We propose a method for arc addition using the updated linkage tree in A. The following algorithm pairs a disconnected new d-sepnode with a connected d-sepnode (either existing or new).

ALGORITHM 5 (PAIRDISCONNECTED).

Input: existing d-sepset D, enhanced d-sepset D^+, subset X of D^+ marked by MarkConnected, and $Y = D^+ \setminus X$;

get linkage tree L with triangulation constrained by D^+;
find a cluster C such that $C \cap Y \neq \emptyset$ and $C \cap X \neq \emptyset$;
pick $v \in C$ such that $v \in Y$ and $u \in C$ such that $u \in X$;
return pair (v, u);

We show that as long as $Y \neq \emptyset$, there always exists a cluster C as described in PairDisconnected, and hence the pair (v, u) can always be found.

PROPOSITION 7. *Let D be the existing d-sepset of a subnet with an adjacent subnet, D^+ be the enhanced d-sepset, X be the subset of D^+ marked by MarkConnected, and $Y = D^+ \setminus X \neq \emptyset$. Then, PairDisconnected will succeed and return (v, u).*

For each pair of d-sepnodes (v, u) identified, agent A notifies agent A'. In response, A' adds new d-sepnode v to its subdomain and adds a new private binary variable c as the common child of v and u. This method has several desirable properties:

First, clearly the new node v in A' is now connected in the updated subdomain of A'. Second, since c is a private node of A', there is no arc to be added in A relative to v. Hence, the state space of its local JT will not be compromised. Third, since c has only two adjacent nodes in A', it will cause the formation of a cluster

$\{c, v, u\}$ in the local JT of A'. Since the cluster has only three variables, it increases the state space of the local JT only slightly.

Disconnected new d-sepnodes may be divided into several groups where nodes in each group is connected to each other. In such case, A needs to run MarkConnected and PairDisconnected once for each group.

10. PROPERTIES OF ENHANCEMENT

We analyze key properties of CoEnhanceInterface. First, after CoEnhanceInterface, each agent interface is a d-sepset.

PROPOSITION 8. *Let E be an enhancement of d-sepset D from agent A relative to an adjacent agent A'. Then $D \cup E$ is a d-sepset.*

Second, CoEnhanceInterface preserves the hypertree condition.

PROPOSITION 9. *Let a d-sepset D between two adjacent subnets of an MSBN M be enhanced to D^+. Then hypertree condition holds in updated structure of M.*

Propositions 8 and 9 guarantee that the outcome of enhancement is a valid MSBN. Finally, all dependence relations in the original MSBN are preserved after agent interface enhancement.

PROPOSITION 10. *Let a d-sepset D between two adjacent subnets of an MSBN M be enhanced to D^+. If the structure of M is an I-map over N, then the updated structure of M is still an I-map over N.*

From Proposition 10, it follows that the enhanced MSBN preserves exact inference.

11. EXPERIMENT

To evaluate the effectiveness of the algorithms, we simulated 10 MSBNs, each with 5 subnets. Each subnet has between 50 to 75 nodes and each variable has a dimension up to 4. Each d-sepset has 8 variables.

For improvement of local search efficience, results of UpdateDsepset in 14 subnets are shown below. The total number of enhancements for each subnet ranges from $2^{24} > 1.6 \times 10^7$ to $2^{51} > 2 \times 10^{15}$. Due to leaf cluster removal and topological dominance, the number of non-t-dominated enhancements per subnet is between 31 and 2495.

Subnet	n_0	n_1	n_2	n_3	n_4	n_5	n_6
$\lvert CanNewD \rvert$	71	355	239	1063	791	35	363
# enhancement	2^{24}	2^{51}	2^{24}	2^{41}	2^{24}	2^{24}	2^{44}
Subnet	n_7	n_8	n_9	n_10	n_11	n_12	n_13
$\lvert CanNewD \rvert$	725	31	35	77	31	119	2495
# enhancement	2^{24}	2^{33}	2^{24}	2^{24}	2^{43}	2^{26}	2^{35}

For overall efficience improvement, results of CoEnhanceInterface with pairwise search are shown below. For each MSBN, SSS for each linkage tree is counted and sum over the four linkage trees is shown. All randomly selected d-sepsets have the worst complexity. Enhancement reduces the sum to between 44% and 94%. As the price, the largest number of preferably private variables disclosed to other agents occurred in an agent with 3 adjacent agents and a subdomain of 75 variables. It disclosed a total of 12 variables (an average of 4 per d-sepset).

MSBN	m_0	m_1	m_2	m_3	m_4
Before enhancement	82944	82944	82944	82944	82944
After enhancement	34286	12398	4562	7220	7020
MSBN	m_5	m_6	m_7	m_8	m_9
Before enhancement	82944	82944	82944	82944	82944
After enhancement	7720	46098	14540	25860	6894

In addition, we simulated a digital system of 216 devices. To monitor it, we built a MSBN model of 5 agents whose subdomains are sized at 91, 202, 72, 149 and 91 variables, respectively. The initial four agent interfaces have a total linkage tree SSS sum of 11264 and it is reduced to 728 (6.5%) through enhancement.

12. CONCLUSIONS

Graphical models such as MSBNs provide a framework for exact and autonomous probabilistic reasoning for multiagent systems. However, construction of such models that are both valid and efficient is involved. This contribution allows MAS designers to start with a model that is not necessarily efficient and let agents to improve its efficiency automatically. This approach relieves designers from the burden of model efficiency and makes the MSBN-based MAS more accessible. Through formal analysis and experiments, we have shown that the proposed agent interface enhancement explores only a small portion of the local search space and joint search space, while improving the model efficiency significantly.

13. ACKNOWLEDGEMENT

Financial support from NSERC of Canada and NSF of USA is acknowledged.

14. REFERENCES

[1] N. Friedman, O. Mosenzon, N. Slonim, and N. Tishby. Multivariate information bottleneck. In J.S. Breese and D. Koller, editors, *Proc. 17th Conf. on Uncertainty in Artificial Intelligence*, pages 152–161, San Francisco, 2001. Morgan Kaufmann.

[2] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, (4):269–282, 1990.

[3] S.M. Mahoney and K.B. Laskey. Network engineering for complex belief networks. In *Proc. 12th Conf. on Uncertainty in Artificial Intelligence*, pages 389–396, 1996.

[4] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[5] J.S. Rosenschein and G. Zlotkin. *Rules of Encounter*. MIT Press, 1994.

[6] N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. In B. Hajek and R.S. Sreenivas, editors, *Proc. 37th Allerton Conf. on Communication, Control and Computation*, pages 368–377, 1999.

[7] Y. Xiang. *Probabilistic Reasoning in Multi-Agent Systems: A Graphical Models Approach*. Cambridge University Press, 2002.

[8] Y. Xiang, J. Chen, and W.S. Havens. Optimal design in collaborative design network. In *Proc. 4th Inter. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS'05)*, pages 241–248, 2005.

[9] Y. Xiang and V. Lesser. On the role of multiply sectioned Bayesian networks to cooperative multiagent systems. *IEEE Trans. Systems, Man, and Cybernetics-Part A*, 33(4):489–501, 2003.

Power and Negotiation: Lessons from Agent-Based Participatory Simulations

Paul Guyot
LIP6 - Université Pierre et Marie Curie - Paris VI
Boîte 169 - 4, place Jussieu, F-75252 Paris Cedex 05
paul.guyot@lip6.fr

Alexis Drogoul
IRD Bondy
32, avenue Henri Varagnat, F-93143 Bondy Cedex
alexis.drogoul@bondy.ird.fr

Shinichi Honiden
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430
honiden@nii.ac.jp

ABSTRACT

Participatory simulations are conducted to improve our knowledge of human behaviors, to help in solving conflicts, to shape interaction protocols between humans and to teach some aspects of collective management.

Agent-based participatory simulations differ from other kinds of participatory simulations including role playing games and experimental economics simulations. The control architecture of the agents, in these simulations, is more or less integrally replaced by a human player and the interactions between players are limited by the communication protocols designed for the agents, usually the exchange of electronic messages logged for further analysis. Such systems can be considered as ideal multi-agent systems featuring cognitive and intelligent agents. Previous work demonstrated that running this kind of simulations helps to design and improve multi-agent simulations.

In this paper, we present a series of agent-based participatory experiments studying negotiation in an abstract case of common resource pool management. The roles were designed in such a way that conflicts should emerge during the negotiations. Observing the behavior of human players, we noticed the apparition of power relations between players. We observed that this power in negotiations was unrelated to any a priori dependence between agents or between roles but was instead drawn from strategies and, more surprisingly, this power was built on an emerging ontology.

General Terms

Experimentation, Design

Keywords

Agent-based simulations, Participatory simulations, Negotiations, Power relations

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

The increasing use of computer-aided participatory simulations is related to an increasing coupling of participatory experiments with agent technologies.

One tradition of participatory simulations consists in combining role playing games with multi-agent systems [4, 5] within what was called the MAS/RPG methodology. Mainly concerned with common resource pool management, researchers from this tradition primarily use participation to better understand human behaviors. This methodology can be interpreted as a kind of participatory design, with an iterative process alternating role playing games and multi-agent simulations, aiming at building systems closer to the actual social phenomena.

Experimental economics, the oldest tradition of participatory simulations, now includes hybrid experiments with agents taking part in the simulation in order to validate and improve models or train participants [18]. The growing use of software agents as human peers fostered methodological studies such as the work of economists Jens Grossklags and Carsten Schmidt on the actual impact of revealing the presence of software agents [13].

From a multi-agent point of view, we define agent-based participatory simulations as simulations where human players take the control of agents. Unlike role playing games or experimental economics, these simulations are built with a multi-agent simulation design methodology. The model is agent-based and the interactions between the entities of these simulations are limited to agent-like interactions. Replacing the control architecture by human players builds up an idealistic multi-agent system with cognitive and intelligent agents. Previous work, based on a coalition formation problem, showed that the strategies elaborated by human players can overtake the distributed solving capabilities of the original multi-agent system [16].

The rationale of SimCommod agent-based participatory experiments was to study possible power relations in negotiations between ideal cognitive agents. The notion of power studied here is limited to negotiations and is based on Dahl's definition [10] : getting an agent to do something that it wouldn't do otherwise. It is of course not what happens when an agent sends a message to or calls a method of another agent.

Such power relations, among other aspects of negotiations, are extremely difficult to model and implement in multi-agent systems. Fifteen years ago, Cristiano Castelfranchi pinpointed the lack of social power in various models of

multi-agent systems [9]. Building on Castelfranchi's work, power in multi-agent systems has been historically associated with a dependence relation. Nevertheless, the tradition of participatory experiments shows that power is not limited to dependence.

This paper is organized as follows. We will first present the background of SimCommod participatory experiments including the notions of power and negotiations in both multi-agent systems and participatory simulations. We will then describe the experiments and how the model and the experimental protocol were tailored to analyze the ways to express power relations as agent interactions. Finally, we will present the results, with the strategies used to influence other players including the emergence of an ontology and we will compare them with power relations in participatory simulations not based on an agent methodology.

2. BACKGROUND

While any interaction between agents can be considered a negotiation, power attracts surprisingly very little interest in the agent community. Participatory simulations, by contrast, almost always feature roles with a priori power relations.

2.1 Negotiation and power in multi-agent systems

Negotiation can be defined as a discussion aimed at reaching an agreement. It can be translated in agents terminology as interactions between agents aimed at a cooperative behavior, ranging from concluding contracts to sharing beliefs. A working group at IWMAS98 [19] tried to characterize negotiation between agents and concluded that negotiation is an extremely broad term that covers most agent interactions. Indeed, our agents usually interact specifically to cooperate or share information.

Historically, negotiation between agents followed fixed protocols and agents that negotiated were designed at the same time. The protocol can be as simple as voting, which is considered a negotiation in the agent literature. With the standardization of agent communication languages, research can now focus on negotiation in open environments with autonomous agents of different origin and design. The very first difficulty when agents engineered by different programmers for different tasks need to negotiate is to allow them to perform an ontology alignment, i.e. to agree on what they negotiate about [22].

While they do provide a framework for collective behaviors, these negotiation protocols have very little in common with real-world situations of conflict. Unlike people, agents are designed to cooperate. Negotiation protocols that lack the notion of power are obviously inadequate to describe real-world conflicts.

Cristiano Castelfranchi introduced the notion of power in multi-agent systems by drawing a social power theory [9]. An implementation of this theory was proposed, relying on dependence relations between agents and on the ability to reason about others [21]. Dependence relation theory was later improved to address the case of dependences between more than two agents and to be more robust to possible changes in the strategies of other agents [7].

However, in participatory simulations, some of the power relations within the negotiations are not dependence relations.

Type	Expected outcome	Participants	Examples
Research	Design, improve and validate models	Stakeholders	SHADOC, SimCafé, PAT
Education	Teach the articulation between individual and collective behaviors	Students	Virus, ButorStar
Action	Favor negotiation between stakeholders	Stakeholders (exchanged roles)	Sylvopast

Figure 1: Typology of participatory simulations

2.2 Negotiation and power in participatory simulations

Participatory simulations can be sorted into three different kinds based on their expected outcomes (Fig. 1). Research participatory simulations aim at designing, improving and validating models of social behaviors. Education participatory simulations focus on training students and teaching them the articulation between individual and collective behaviors. Action participatory simulations try to favor negotiation between stakeholders by exchanging their roles.

In Sylvopast [11], an example of action simulation, participants are foresters or shepherds. As defined by their role, shepherds cannot perform forest management. They need to find incentives to convince the foresters to adapt the forest to their sheep. Officially, the foresters are independent and can ignore the requests of the shepherds. Actually, if the shepherds are eliminated after going bankrupt, the computer model will set fire to the forest. While there have been occurrences of this scenario during some experiments, stakeholders playing Sylvopast often know each other and share an history of common work in the forest. In spite of tensions between the two groups, they often succeed to cooperate.

ButorStar [17] is a typical example of an education participatory simulation as it was specifically designed for teaching purposes. Roles represent different stakeholders of the reedbeds: hunters, fishermen, breeders, reed harvesters and environmentalists. These roles are designed with conflicting interests and particularly, the economic interest of the other players is conflicting with the environmentalists who try to preserve the great bittern, an endangered species living in the reedbeds and after which the game is named (*Butor Etoilé*). In this role playing game, players have to cooperate to choose an irrigation level that affects everyone's outcome. The game master enters the final decision in the computer and, when no agreement emerged, the actual irrigation is the worst possible case for everyone.

In these examples, negotiation is a key element of the experiments. The models even punish negotiators who have failed to reach an agreement. The negotiation phase takes place in natural language, without the help of any computer. Many factors influence the outcome of the negotiation and are not logged or related to the game. We have no scientific means to analyze the power relation between players during such games. In fact, role playing game designers often intro-

duce a priori power relation between the roles they define. Interestingly, they notice the number of players with a given role affects the outcomes of the negotiation. For example, in ButorStar, two environmentalists rather than one will better succeed in preserving the great bittern because they are more convincing.

On the contrary, most experiments in economics, which are of the research type, do not channel negotiations through natural language. Instead, interactions and negotiations are based on abstract symbols for two reasons. First, the price, which is what is most often negotiated, is decided by using what economists call institutions, i.e. the mode of interactions between agents. Institutions usually consist in some auction protocol. Second, experimental economists use abstract symbols instead of representation of emotional concepts such as endangered species and players are anonymous to avoid conflicts with a previous relation [12, 8]. Economist James Andreoni argued that players behave differently depending on how things are called [1].

3. THE SIMCOMMOD EXPERIMENTS

The SimCommod experiments are agent-based participatory simulations that used the Simulación framework [15]. They inherit the conflicting situation of the e-ComMod model and they also borrow methodological aspects from experimental economics. The experiments took place in june 2005 in the Laboratoire d'Informatique de Paris VI and in september 2005 in the National Institute of Informatics, Tokyo.

3.1 Agent-based participatory simulations

Agent-based participatory simulations are multi-agent simulations where human players act as the control architecture of agents. In experiments based on the Simulación framework, each agent is a network entity. Each player sits at a computer and interacts with other players through the computer only (Fig. 2).

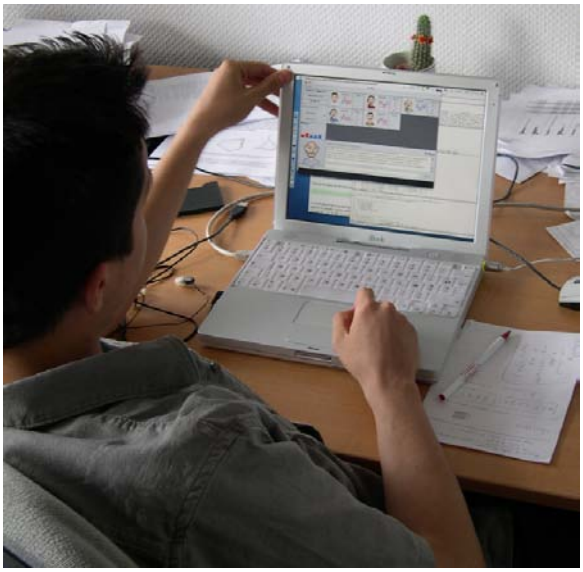


Figure 2: A player in an agent-based participatory simulation

The design of agent-based participatory simulations follows standard multi-agent simulations design methodologies. In particular, the process is iterative and consists in transforming the domain model into an operation model. While the operational model of multi-agent simulations is the actual code, in agent-based participatory simulations it is threefold: the experiment protocol, what players can do (agent effectors) and what information they have access to (agent sensors). When testing interaction patterns, the most important step in this design process is to break the exchanges into smaller primitives so that players can have unexpected behaviors and are not forced to comply to the original domain model [14].

Interactions are limited to messages exchanged between agents and are therefore easily recorded with one or more log agents on the network. The logs are then used for further analysis of the interaction patterns. Moreover, the environment which is the subject of the negotiation is yet another network entity. In the SimCommod experiments, the environment was simulated by a multi-agent model in Cormas [6]. Cormas could also simulate some agents such as the government.

3.2 The e-ComMod model

The e-ComMod model has been designed to epitomize the companion modeling approach [3]. It derives from classical models of participatory simulations belonging to the common resource pool management modeling tradition.

The environment is simulated by a cellular automata implemented in Cormas. It is represented by a grid of 16 cells. A resource is distributed on the cells. The amount of resource on each cell is a discrete property with four different levels, from none to plenty, represented by shades of green (Fig. 3).

Agents in the model belong to three different classes: the harvesters, the environmentalists (also called the conservationists) and the government (the policy maker). The harvesters are the agents really acting on the environment. The government can only prevent them from acting on some cells and the environmentalist has strictly no direct influence on the environment.

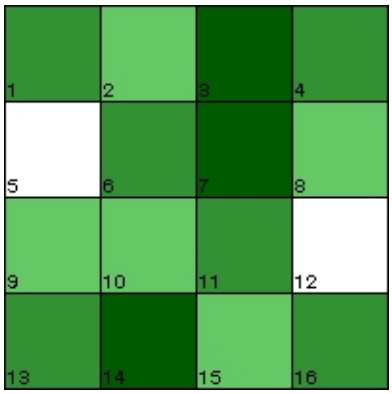


Figure 3: Environment: a grid with shades of green

The experiment is divided in turns. Each turn is divided in 4 phases: harvest, proposition, negotiation and validation.

- During the first phase, the harvesters can work on a maximum of three cells. The yield of a cell depends on the amount of resource and on the number of harvesters. The resource available on the cell is distributed to the harvesters.
- During the proposition phase, the harvesters and the environmentalists put positive and negative tokens on the map. The positive tokens represent the cells that they want to harvest next turn and the negative tokens the cells that they want to protect from being harvested. The harvesters have three tokens each, which they can cast as positive or negative tokens, and the environmentalists have ten tokens each.
- During the negotiation phase, the propositions are revealed synchronously to all players (Fig. 4). They can submit a new distribution of their tokens with an optional comment in natural language. The sum of the propositions may or may not converge during this phase.
- During the validation phase, the government decides to ban three cells. These three cells will not be available for harvesting during next turn. The decision can be taken from the sum of the tokens or the government can take actions to favor the emergence of a negotiation by penalizing the players who could not reach a consensus. The interface for the player who assumes the role of the government does not impose any strategy. The way the environmentalist and the harvesters affect this role's behavior is entirely up to the player. Besides, the user interface displays the discussion during the negotiation but the government cannot interfere with the negotiation process. It can only comment its decision.

The harvester's goal is to harvest as much resource as possible. There is competition among harvesters. Their best strategy is to harvest on cells with the maximum amount of resource and no other harvesters, since the resource is distributed among harvesters on a given cell.

Oppositely, the environmentalist's goal is to preserve the resource. The environmentalist does not act except by putting tokens on the map to influence the decision of the government. e-ComMod is similar to ButorStar in this respect where what some players could do was only to influence decisions of others. In the Sylvopast model, on the contrary, the shepherds can give money to the foresters and sheep actually clean the forest and decrease the risk of fire.

The actual evolution of the cellular automata is not revealed to the players, since they have to discover the dynamics of the game by themselves. The evolution rules of the level of resource on a cell do not depend on any other cell. In other words, the representation of the environment as a grid is strictly unrelated to the actual dynamics of the environment. The rules are the following:

- if more than one harvester is present, the level is decreased by the number of harvesters
- if one harvester is present, the level does not change.
- if no harvester is present for one turn, the level is incremented.

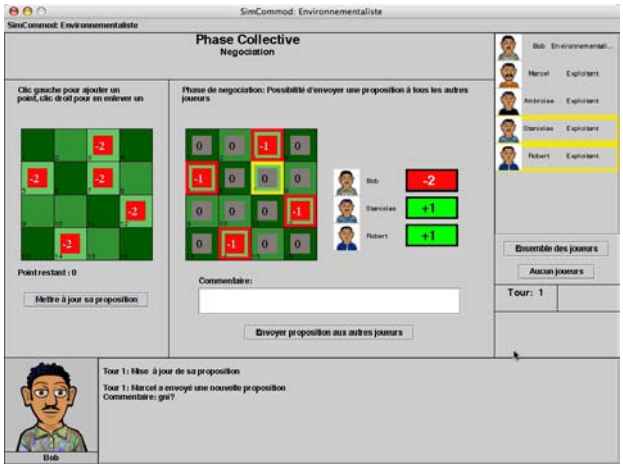


Figure 4: The user interface during negotiation

- if no harvester was present for the last two turns, the level does not change.
- if no harvester was present for more than two turns, the level is decremented.

These rules reflect actual common resource pool management situations where resource should not be over-harvested (more than one harvester) but a regular harvest maintains the resource.

3.3 Experiments to analyze power relations

A participatory experiment is typically conducted in three phases: a presentation of the model, the experiment and a debriefing session. The experiment conducted in the Laboratoire d'Informatique de Paris VI consisted in a single run. The model was calibrated during this first experiment. With five harvesters and one environmentalist, in addition to the competition between harvesters, the model featured a conflict situation: the environmentalist's goal is to preserve the resource and the pressure from the harvesters will make the amount of the resource decrease.

The experiments conducted at the National Institute of Informatics consisted of three runs. The first two sessions were performed in parallel and the government was played by an agent in Cormas. The players were told that the government would sanction the lack of agreement while in fact it played randomly. No complete agreement emerged, yet a player insisted that he found out that the government was acting randomly. During the third experiment, the government was controlled by a human player who penalized harvesters for their lack of agreement.

In order to study power relations between agents, the experiments had to be disconnected from the players' history. The participants knew each other, but the game was entirely anonymous. Players were represented by an avatar that was randomly chosen by the organizers (Fig. 4). While the simulation took place in an open space, players did not look into their neighbor's cubicle and seemingly did not try to find out who was controlling a particular avatar. Besides, when two experiments were conducted in parallel, players could not determine if their neighbor took part in the same

game. Anonymity seems to have been effective, even if negotiations were done in natural language: a player mistakingly thought that one of the avatar was driven by a particular fellow student while it wasn't.

An additional aspect of the methodology of experimental economics was introduced. Except for the name of the roles, the model used abstract symbols rather than emotional concepts such as an endangered species. The resource was only ever called "a resource" during the presentation of the model.

4. RESULTS

The analysis of the results is based on the logs of all exchanges during the experiment as well as interviews during the debriefing sessions. The most surprising aspect of the experiment is the emergence of a new ontology and its role in the power relations. This contrasts with previous participatory experiments grounded in real-world situations.

4.1 Ontologies and power relations

Although all participants had the same information about the game, during the first two runs conducted in parallel at the NII, one of the two environmentalists obviously negociated better than the other. When asked about their strategy, both mentioned that they tried to preserve some specific areas that needed to be preserved. Their choice seemed to consist in preserving the lighter green squares.

The less efficient environmentalist explained that he tried to convince the harvesters to work on the cells that would provide them with the highest revenue while preserving the selected cells. In other words, it consisted in a rationale negotiation approach, trying to obtain a reasonable agreement that served everyone's interest. The player who chose to always follow the environmentalist's advice is the one who collected the highest amount of resource. Others mentioned they simply ignored his messages.

The more efficient environmentalist's strategy was instead to tell players not to harvest the specific cells she wanted to preserve:

> I tried to protect this area by telling people "that's a very important area, go away", some people believed it very easily but some people opposed it.

The resource was better preserved with this strategy. According to these debriefing interviews, forbiding some cells proved to be more efficient than prescribing. Besides, this example seems to contradict common sense. Finding the agreement that satisfies one's constraints while maximizing other players' interest seems less effective than just trying to achieve one's goal and letting other players decide by themselves. This would contradict the social power theory as it would not be necessary to figure out the best interest of the other players.

In fact, the logs reveal that the main difference between the two environmentalists is not about forbiding and prescribing or advising. The more efficient environmentalist introduced notions such as endangered forest and trees, even if they were not present in the original presentation of the model that carefully used only abstract symbols. Furthermore, during the third experiment, the same environmentalist succeeded in prescribing as harvesters concentrated on some cells, allowing others to regenerate:

> In the final turn, it was in a way good that many harvesters concentrate on one white cell and I saw many white cells empty, in the end it worked well.

The success of this new strategy, based on prescription like the strategy of the environmentalist during the first run, is again related to the introduction of concrete notions. The environmentalist told harvesters that they could find gold in these cells. She was conscious that harvesting white cells was their less efficient behavior and somewhat surprised that this argument worked so well. Reluctantly, some players admitted they believed her during the debriefing session and the logs show that some white cells were harvested indeed, decreasing the global pressure on the resource.

4.2 Comparison with participatory experiments

Participatory simulations based on real common resource pool management problems introduce an a priori ontology. They also use abstract symbols. In Sylvopast, for example, the vegetation type is represented by primary colors. The cells with forest are blue, with shrubs red and with grass yellow. This code allows combination with composed colors. Michel Etienne notes that after one round, players argue about "more yellow" or "more blue" instead of "more grass" or "more forest".

Such a use of abstract symbols is not contradictory with the results of the SimCommod experiments. Players in role playing games need an a priori ontology to negotiate but their discussion can be based on symbolic information. In fact, in newer flavors of Sylvopast, some new roles have their own ontology that they can use to influence other players. In particular, the objective of the newly introduced environmentalist is to preserve an endangered species of turtle living in the forest. The very idea of an endangered species of turtles is absent from the ontologies used in negotiations between foresters and shepherds.

The environmentalists in the SimCommod experiments may have used an ontology to convince and influence the players but nevertheless used the abstract symbols to clearly explain what they wanted. For example, the environmentalist during the third run at the NII told players to go on "white cells" because there was gold there. The action is expressed in the common ontology based on abstract symbols.

On the other hand, experiments in economics that do not allow the emergence of an ontology and that only include emotionless abstract concepts could more difficultly feature power relations. We previously ran an agent-based participatory set of experiments based on a cooperative version of the El Farol bar problem [2]. The concepts were all abstract, participants played anonymously. Lacking the possibility to communicate in natural language and therefore to create an ontology, they could not negotiate and did not exhibit any power relation. During the debriefing sessions, some participants insisted that they needed to be able to tag other players as liars or cooperative in order to negotiate.

5. CONCLUSION

In multi-agent systems, power is the corollary of autonomy and pro-activity. Cooperation between agents that do not have a slave behavior implies that agents must influence each

other, especially when interests diverge. Likewise, in agent-based participatory simulations with built-in situations of conflict, human players who control the agents necessarily introduce power relations.

The power relations in multi-agent systems have been previously studied as dependence relations. From a first look at the SimCommod experiments where there was no direct dependence between agents, we could infer that prescribing is less effective than forbiding as a way to influence others. We could also infer that it is not necessarily a good strategy to try to find an agreement maximizing the interests of the other agents. The real lesson from SimCommod experiments is probably that some actions such as introducing new ontologies can be a surprisingly effective way to influence other participants. Consequently, in the multi-agent negotiation protocols that take the ontology alignment into account, the ontologies should not be merged beforehand but during the negotiation.

More generally, this paper also advocates the use of agent-based participatory simulations to test and validate features of interaction protocols. In the forthcoming months, we plan to run additional agent-based participatory experiments based on a similar model to test overhearing and over-sensing protocols against direct interactions [20].

Acknowledgments

The experiments would not have taken place without the work of Christophe Le Page and Patrick Taillandier. The former originally developed the e-ComMod model in Cormas and the latter adapted it to the Simulación framework. We would like to thank participants from the Laboratoire d'Informatique de Paris VI. We would also like to thank every member of Honiden-Lab at the National Institute of Informatics as well as Haruko Ishikawa for their participation in the simulations and for the extremely valuable feedback they gave us.

6. REFERENCES

[1] J. Andreoni. Warm-glow versus cold-prickle: the effects of positive and negative framing on cooperation in experiments. *Quaterly Journal of Economics*, 110:1–22, 1995.

[2] B. Arthur. Inductive reasoning and bounded rationality. *American Economic Review*, 84:406–411, 1994.

[3] O. Barreteau. Our companion modelling approach. *Journal of Artificial Societies and Social Simulation*, 6(2), 2003.

[4] O. Barreteau and F. Bousquet. Jeux de rôles et validation de systèmes multi-agents. In G. M-P and M. P, editors, *Ingénierie des systèmes multi-agents, actes des 7èmes JFIADSMA*, pages 67–80. Hermès, 1999.

[5] O. Barreteau, F. Bousquet, and J.-M. Attonaty. Role-playing games for opening the black box of multi-agent systems: method and lessons of its application to senegal river valley irrigated systems. *Journal of Artificial Societies and Social Simulation*, 4(2), 2001.

[6] F. Bousquet, I. Bakam, H. Proton, and C. Le Page. Cormas: Common-pool resources and multi-agent systems. In A. P. Del Pobil, J. Mira, and M. Ali, editors, *Tasks and Methods in Applied Artificial Intelligence, 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-98, Castellón, Spain, June 1-4, 1998, Proceedings, Volume II*, volume 1416 of *Lecture Notes in Computer Science*, pages 826–837. Springer-Verlag, 1998.

[7] S. Brainov and T. Sandholm. Power, dependence and stability in multiagent plans. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'99), Orlando*, pages 11–16. AAAI Press, 1999.

[8] C. F. Camerer and E. Fehr. Measuring social norms and preferences using experimental games: A guide for social scientists. In J. Henrich, R. Boyd, S. Bowles, C. Camerer, E. Fehr, and H. Gintis, editors, *Foundations of Human Sociality - Experimental and Ethnographic Evidence from 15 Small-Scale Societies*, page 320. Oxford University Press, 2004.

[9] C. Castelfranchi. Social power: A point missed in multi-agent, dai, and hci. In Y. Demazeau and J.-P. Müller, editors, *Decentralized Artificial Intelligence: Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-89)*, pages 49–62, Amsterdam, 1990. Elsevier.

[10] R. Dahl. The concept of power. *Behavioral Science*, 2(3):201–215, July 1957.

[11] M. Etienne. Sylvopast: a multiple target role-playing game to assess negotiation processes in sylvopastoral management planning. *Journal of Artificial Societies and Social Simulation*, 6(2), 2003.

[12] D. Friedman and S. Sunder. *Experimental Methods - A Primer For Economists*. Cambridge University Press, Cambridge, 1994.

[13] J. Grossklags and C. Schmidt. Artificial software agents on thin double auction markets: A human trader experiment. In *Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology (IAT-2003), Halifax, Canada, October 13-17, 2003*, 2003.

[14] P. Guyot. Simulations multi-agents participatives: Vers une méthode de validation et de consolidation des modèles de pratiques collectives. Master's thesis, Université Pierre et Marie Curie, 2003.

[15] P. Guyot and A. Drogoul. Multi-agent based participatory simulations on various scales. In T. Ishida, L. Gasser, and H. Nakashima, editors, *Proceedings of International Workshop on Massively Multi-agent Systems*, volume 3446 of *Lecture Notes in Computer Science*, pages 149–160, Berlin Heidelberg, 2005. Springer-Verlag.

[16] P. Guyot and A. Drogoul. Using emergence in participatory simulations to design multi-agent systems. In F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. P. Singh, and M. Wooldridge, editors, *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-05)*, 2005.

[17] R. Mathevet, C. Le Page, M. Etienne, G. Gigot, G. Lefebvre, B. Poulin, and A. Mauchamp. Butorstar: a role-playing game for collective awareness of reedbed wise use. Technical report, In prep., 2005.

[18] K. Nakano, S. Matsuyama, and T. Terano. Research

on a learning system toward integration of case method and business gaming. In H. Deguchi, K. Kijima, T. Terano, and H. Kita, editors, *Proceedings of The Fourth International Workshop on Agent-based Approaches in Economic and Social Complex Systems (AESCS) 2005*, 2005.

[19] H. V. D. Parunak. Characterizing multi-agent negotiation, 1998.

[20] E. Platon, N. Sabouret, and S. Honiden. Overhearing and direct interactions: Point of view of an active environment, a preliminary study. In D. Weyns, H. V. D. Parunak, and F. Michel, editors, *Proceedings of AAMAS workshop on Environment for Multi-Agent Systems (E4MAS 2005)*, 2005.

[21] J. S. a. Sichman, D. Yves, R. Conte, and C. Castelfranchi. A social reasoning mechanism based on dependence networks. In *Proceedings of the European Conference on Artificial Intelligence*, pages 274–278. Wiley & Sons, 1994.

[22] J. Wang and L. Gasser. Mutual online ontology alignment. In *OAS'02 Ontologies in Agent Systems, Proceedings of the AAMAS 2002 Workshop*, volume 66 of *CEUR Workshop Proceedings*, 2002.

RPD-Enabled Agents Teaming with Humans for Multi-Context Decision Making

Xiaocong Fan
College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802
zfan@ist.psu.edu

Bingjun Sun
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
bsun@cse.psu.edu

Shuang Sun, Michael McNeese, John Yen
College of Information Sciences and Technology
The Penn State University
University Park, PA 16802
{ssun,mmcneese,jyen}@ist.psu.edu

ABSTRACT

Team decision making under stress involving multiple contexts is an extremely challenging issue faced by various real world application domains. This research is targeted at coupling cognitive agent technology and human-centered teamwork to address the informational challenges associated with Command and Control (C2) teams in contemporary military operations. Two sets of experiments, each with various settings of context switching frequencies and tasking complexities, were conducted. To ensure that the human subjects were familiar with the C2 context, they were selected from US Army ROTC (Reserve Officer Training Corps) students. Experiments on C2 teams that involve human subjects only were conducted first. We observed the decision making behavior of human subjects and incorporated expert behaviors into R-CAST—an agent architecture built upon a naturalistic decision making model that captures how domain experts make decisions based on experiences and situational similarity recognition. We then conducted another set of experiments with R-CAST agents as teammates and decision aids for human subjects. The results show that RPD-enabled agents can significantly improve the tasking capacity of C2 teams in multi-context decision making under stress. It also suggests that higher demand situations require more competent teammates.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*

General Terms

Design, Experimentation, Human Factors

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Keywords

Cognitive agent, Human-centered teamwork, Naturalistic decision making, Multiple contexts, Attention switching

1. INTRODUCTION

Team decision making involving multiple contexts is an extremely challenging issue faced by various real world application domains. Command and Control (C2) in complex urban terrain is one of such domains, where C2 teams have to frequently confront the so called three-block challenge [5]—conducting humanitarian, peacemaking, and combat missions in close proximity (i.e., involving three contexts that overlap in time). Multi-context team decision making is challenging because it requires effective team collaborations in rapidly gathering dynamic information from multiple sources (collateral space), in proactively sharing relevant information for establishing global situation awareness, in managing and reasoning across multiple decision spaces for different contexts (areas of interest), and in choosing optimal corridors for movement.

In the Multi-Agent Systems field (MAS), there has been an increasing interest in empowering agents with naturalistic decision making models to better support *human-agent collaboration* in making decisions under time stress [4, 12, 13, 14, 16]. One of such models is Klein's Recognition-Primed Decision framework (RPD) [8]. The RPD model claims that in complex situations human experts usually make decisions based on the recognition of similarities between the current decision situation and previous decision experiences [9]. Cognitive studies have shown that over 95% of human decisions conform to the RPD model in various time-stressed situations [9]. Norling, Sonenberg, and Ronnquist [12] have examined the integration of RPD model into BDI agent framework. Fan, Sun, McNeese, and Yen [4] have implemented an RPD-enabled cognitive agent architecture (R-CAST) and evaluated the performance gains when human decision makers were assisted by R-CAST agents, operating under stress in a real-time simulation environment.

However, multi-context decision making, especially from the human-centered teamwork perspective, has attracted little attention from the MAS field. Norling et al.'s work [12] only explored ways of using reinforcement learning to en-

hance situational recognition. While Fan et al.'s attempt [4] centered on human-agent collaboration in the decision making process, they only investigated the impact of an adaptive decision making mechanism, with the issue of multi-context decision making left open. Although RPD-enabled agents were employed to start to address the issue of multiple contexts [5], we still need experimental studies for fully understanding the nature of multi-context decision making, which is critical to further developing agent technologies for enhancing the performance of human-centered teamwork.

In this paper, we investigate how and to what extent R-CAST agents, as teammates and decision aids, might help in human decision making under stress and multiple contexts. This research is conducted within the C2 domain and is targeted at coupling cognitive agent technology and human-centered teamwork to address the informational challenges associated with multi-context team decision makings. The remainder of this paper is organized as follows. Background on human-centered teamwork and the R-CAST agent architecture are given in Section 2. The problem domain, simulation environment, and scenario designs are described in Section 3. Experiments on human-only C2 teams and human-agent C2 teams are presented in detail in Section 4 and Section 5, respectively. Section 6 presents conclusions.

2. BACKGROUND

2.1 Human-centered teamwork

Multi-agent teamwork [3] is concerned with joint commitments and joint responsibility. In particular, human-centered teamwork, where human and software agents interact as peers, is about establishing situation awareness collaboratively, developing shared mental models as situation evolves, adapting to mixed-initiative activities, etc. It is claimed that agents, if they could collaborate with human peers effectively, can allow humans to pay attention to more important activities [1], and make better decisions using information at a greater accuracy and finer granularity[11].

Bradshaw et al. developed KAoS, a collection of agent services in the form of adjustable policies, which allow the renegotiation of roles and tasks among humans and agents when new opportunities arise or when breakdowns occur [1]. MokSAF is a computer-based simulation system developed to evaluate how humans can interact and obtain assistance from agents within a team environment [11].

Our research reported here differs from the above efforts in that we attempt to deepen the understanding of human-centered teamwork by examining multi-context decision making in time-stressed situations. In particular, we try to uncover the potential impacts of RPD-enabled (or experience-guided) cognitive agents, when acting as human's teammates and decision aids, on the performance of C2 teams.

2.2 The R-CAST Agent Architecture

The R-CAST agent architecture [4] is built on top of the concept of shared mental models [2], the theory of proactive information delivery [6], and Klein's Recognition-Primed Decision (RPD) Model [8]. The R-CAST agent architecture has implemented a "collaborative-RPD" decision process, which supports close human-agent collaborations in relevant information sharing, decision progress monitoring, and expectancy-based decision adaptation.

The major components of R-CAST are shown in Figure 1, where the "RPD-based Decision Making" module uses domain knowledge, past experiences, and the current situation awareness to produce a new or adapt an existing decision. The fulfillment of a decision can involve inter-agent and intra-agent activities, which are coordinated by the "Teamwork manager" and "Taskwork manager", respectively. Any unexpected conditions resulting either from situational changes or from action failures will trigger another round of decision making. Also, any behavior (plan) adaptations that have been tested as workable will be used to adapt the relevant experiences.

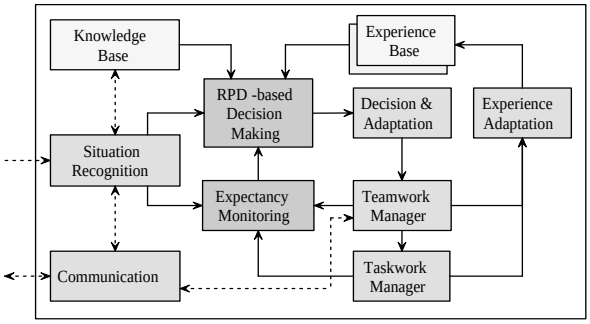


Figure 1: Major components of R-CAST.

R-CAST is a configurable architecture. Many of the components can be replaced (e.g., use JESS as the knowledge base) to serve the purposes of various studies. If the default R-CAST configuration is kept, using R-CAST to develop a multi-agent system involves two major steps: (1) elicit domain knowledge and implement a domain adapter for the problem to be studied, and (2) elicit and represent domain expertise about decision making in terms of experiences (cf. Section 5.1.1 for an example and refer to [4] for detail).

Two important features that are relevant to this study are the "expectancy-based adaptive decision making" and "situation-guided switching of decision spaces."

First, R-CAST agents adapt their decisions by detecting and responding to situations that are anomalies to the original expectancy (i.e., what is observed/informed conflicts with what was expected). Anomalous situations can happen frequently in a dynamic environment. For example, suppose under the current situation it is expected that two crowds, G_1 and G_2, should be active in two isolated regions. Anomaly occurs if G_1 and G_2 start to move closer and closer.

Experiences in an experience base are organized hierarchically: experiences at a higher level are refined by those at a lower level (with more relevant cues and expectancies considered). The RPD-based decision making is a process of finding an experience that worked in a previous situation that is closest to the current situation according to certain similarity metrics. This is an iterative and adaptive process in R-CAST. Start with the most abstract experience in the current decision space, as more and more situational information becomes available, agent's recognition of the past experience closest to the current situation becomes more and more concrete (at the lowest possible level of the hierarchy). During the process, the agent simultaneously monitors the expectancies associated with the recognized experience. The recognition is enforced as new events are emerging as expected. It is challenged when some expectancy becomes

false as the situation evolves; in such a case, the agent has to backtrack along the experience hierarchy to seek a better recognition. In the study reported here, this feature is used such that an R-CAST agent (S2) can determine what information the other teammates (S3/S4 agents) need, and then proactively provide the information.

Second, R-CAST allows the specification of multiple experience bases, and the management of multiple decision spaces at run time. The nature of a situation determines what the current working-context is, and situational changes can cause the switch of one context to another. Consequently, an agent working in one decision space can switch into another decision space, and return to a previous space when the current context switched back. This feature allows us to implement agents that can dynamically switch between humanitarian, peacemaking, and combat contexts.

3. SETTING STAGES FOR EXPERIMENTS

In this section we describe the problem domain, scenario design, and the simulation engine used in the experiments.

3.1 Problem Domain

The problem domain involves C2 teams reacting to potential threats that emerge unexpectedly in a metropolis. It imposes challenging information demands associated with the command and control of urban operations, including humanitarian, peacekeeping, and combat operations.

In the environment, of special interest are main supply routes (MSRs) and three kinds of key buildings (religious buildings, schools, and hospitals). A C2 team consists of S2 (intelligence cell), S3 (operations cell), and S4 (logistics cell); they, as a team, need to work collaboratively to handle incoming threats. Their roles have been simplified and defined as follows: S2 is responsible for processing incoming reports, called Spot reports, collecting relevant information from other sources, and alerting S3 and/or S4 of threats appropriately. S3 needs to process alerts from S2 and decide on the courses of action (COA) to handle threats, given varying timing constraints and limited resources. Two types of friendly units are under S3's charge: squad units and EOD (Explosive Ordnance Disposal) teams. S4 needs to process alerts from S2 and minimize damages to MSRs.

Potential threats come from three types of targets: Crowds, Insurgents, and IEDs (Improvised Explosive Device). A *crowd* represents a group of people that may contain key friends or foes. A crowd can be of medium (M) or large (L) size, and the group size of a crowd can change over time. A crowd can move around and it may get merged with another crowd. Table 1 shows under various situations the credit value (the points a C2 team can get if a target is handled successfully) of a crowd, the resources required to handle a crowd, and what action should be taken on a crowd, where 'U' refers to "squad unit", and the value of a crowd has additional 10 points when the crowd is near an MSR. For example, the second entry says that to disperse a medium size crowd with a foe needs two squad units, and 40 points can be credited if the crowd is dispersed successfully.

Another type of movable targets is *insurgents*, each is associated with a threat level that can be L(low), M(medium), or H(high). As shown in Table 1, how many squad units are required to capture an insurgent depends on its threat level. *IEDs* are motionless targets. An IED, if exploded, can cause damage to the nearby buildings and MSRs. One squad unit and one EOD team (E) are required to remove an IED. If successful, 60 points can be credited if the IED is close to buildings only or MSRs only, 80 points if it is close to both.

Table 1: Requirements on handling targets

Targets			*Value*	*Res. req.*	*Action*
Crowd	M	w/o foe	20	1U	monitor
	M	w/ foe	40 (+10)*	2U	disperse
	L	w/o foe	40 (+10)*	2U	disperse
	L	w/ foe	50 (+10)*	3U	disperse
Insurgent (3 threat levels: L, M, H)			50+50n	(n+1)U	capture
			n=1,2,3 for L,M,H		
IED			60 (80)*	1U + 1E	remove

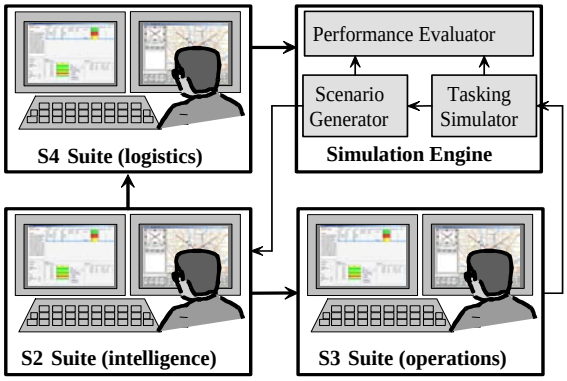


Figure 2: Environment settings of the experiments.

3.2 Simulation Engine

Figure 2 depicts the general settings of the experimental deployment, where each C2 cell (S2, S3, S4) has equipment with a map display and a human user interface (HUI). Human operators can monitor the current situation through the map displays, and interact with other peers through their HUIs.

The development of situations is controlled by the Simulation Engine module. Spot reports about situational changes are dynamically generated and fed to the S2 suite following certain controllable patterns (e.g., every n seconds; interleaving reports of different target types in fixed order or randomly). The S2 suite recognizes potential threats and decides whether and when to alert the S3 and S4 suites. Thus, although each suite has a map display, the S3 and S4 suites may only have partial views of the current situation.

The Simulation Engine module has three components: scenario generator, tasking simulator, and performance evaluator. At every cycle, the *scenario generator* creates a Spot report for each active target based on the script of its dynamics. Here is an example script of a crowd:

```
(  (crowd C121 40.04 -82.96 0.8 10)
   (report_key A12 friend 7) (successor C112)
   (m 40.04 -84.995 1.0) (m 42.05 -82.975 0.5)
   (s 27 20) (s 39 90))
```

This defines a crowd named C121, that appears at location (40.04, -82.96) 10 seconds after the system starts, and can move at full speed of 0.8 u/s. A friend named A12 shows up 7 seconds later. The crowd moves to (40.04, -84.995) at

Table 2: Scenario settings

Active targets	*Macro context switching frequency*		
(3 types in total)	*L: 15 sec.*	*M: 10 sec.*	*H: 5 sec.*
TC = 2 (× 3)	Scenario 1	Scenario 4	Scenario 7
TC = 3 (× 3)	Scenario 2	Scenario 5	Scenario 8
TC = 4 (× 3)	Scenario 3	Scenario 6	Scenario 9

its full speed, then moves to (42.05, -82.975) at 0.5 times of its full speed. The crowd size changes to 27 after 20 seconds and to 39 after 90 seconds. The disappearance of this crowd will trigger another target named C112 to show up.

Whether a generated Spot report is sent to S2 suite depends on the communication pattern set before each run, which is flexible, so that it allows, for example, the setting where reports about crowds being sent every other cycle.

The *tasking simulator* manages the tasks issued by the S3 suite by monitoring the engagement (e.g., taskload) of each squad/EOD unit, tracking the progress of each ongoing task, and reporting to scenario generator about any situational changes effected by the completion of a task.

The *performance evaluator* records the number of targets successfully handled; whether the S3 suite paid attention to the nearby buildings when issuing tasks; how many times all the resources are engaged after issuing a task; how many times S3 canceled a task that was no longer correct relative to the current situation; how many times resources were wasted on non-threatening targets (i.e., small crowds, IEDs far away from key buildings or MSRs); and how many times resources were re-allocated to another target that could offer a better chance to succeed.

3.3 Scenario Design

To study multi-context decision making, we first clarify what "context-switching" means in the problem domain. Humanitarian context refers to activities related to IEDs; peacekeeping context refers to activities related to crowds; and combat context refers to activities related to insurgents. It is a *macro* context-switching when Spot reports from two consecutive cycles are about different types of targets.

When the S2 (or S3) suite gets Spot reports (or alerts) about multiple targets simultaneously, even though they are of the same type, human operators have to switch their attention back and forth to handle the targets. We refer to this as *micro* context-switching. Because task difficulty correlates with time demands [17], varying the number of active targets (i.e., controlling *micro* context-switching) changes how much time a human operator can take on each target, thus varying task complexity.

We thus designed 9 scenarios as listed in Table 2, each with a different combination of context-switching frequencies and task complexities. For example, Scenario 5 switches context every 10 seconds, and has 3 active targets for each type (9 active targets in total). The scenarios also differ in settings of initial locations (targets, MSRs, key buildings, IEDs), itineraries and velocities of movable targets, sizing of crowds, threat levels of insurgents, targets' appearance time, etc. These are specified as scripts to the Scenario Engine.

Each target has a lifespan, which depends on whether a movable target moves to the end of its pre-specified itinerary, whether it comes to the expiration time of a motionless target, or whether the target has been successfully handled by the S3 suite, whichever comes first.

Other settings common to the 9 scenarios include: (a) each scenario lasts 10 minutes; (b) under the control of the S3 suite are 10 friendly units (1 EOD team and 9 squad units); they all move at the same speeds; (c) crowds with size between 10 and 35 are of M size, crowds with more than 35 people are of L size; (d) the Scenario Engine is set such that Spot reports of different target types are sent interleavingly in a random pattern.

4. EXPERIMENTS ON HUMAN C2 TEAMS

The aim of the experiments on Human C2 teams is twofold. First, we want to gain a better understanding of how cognitive limitations affect C2 team performance and how to minimize such impact. Second, we need to gather experiences for RPD-agents to use in experiments on human-agent C2 teams. We thus chose US Army ROTC (Reserve Officer Training Corps) students as the human subjects. At the current stage, we only used a dummy agent simulating the S4 suite, focusing mainly on S2 and S3 suites.

4.1 Team Organization

4.1.1 S2 Human

The left screenshot in Figure 3 is the S2 human user interface, the design of which was largely influenced by the domain experts we consulted. Displayed on the left are incoming Spot reports and the details. The upper-left area shows the current *macro* context. In the experiments, S2 human subjects were asked to do communication actions by following the current macro context as far as they could.

Shown in the list on the upper-right pane are fused information about targets, which are categorized into target type, name, status, crowd size, persons associated with crowds, nearby buildings, and whether S3/S4 suites have been alerted once. Except for a target's location information, which will be sent automatically to S3/S4 suites once they are aware of the target, S2 human has to update S3/S4 suites' knowledge about the situational changes in a timely manner. S2 humans can perform four kinds of communication actions. After selecting an active target from the target list, an S2 human can (a) press the button "MIDB Query" (MIDB stands for Military Integrated DataBase). Depending on the type of the selected target, from MIDB S2 humans can gather threat level information for an insurgent, recognize whether a key person is a friend or foe, and update information about buildings nearby an active target; (b) press the button "Alert to S3" to refresh S3 suite's view of the selected target; (c) press the button "Alert to S4" to alert S4 when the selected target is moving close to a MSR; and (c) press the button "Cancel Alert to S4", if the selected target is moving away from MSRs. The bottom portion of the interface shows S2's communication behaviors.

4.1.2 S3 Human

Figure 4 shows a screenshot of the S3 user interface used in *human-agent* C2 experiments. The interface used in human-only experiments is the same except that (1) there is no cue for resource requirements (i.e., the "Expected Units" column is blank), and (2) there is no color coding for target priorities (cf. Section 5.1.2 for improvements and the reasons).

Shown on the left are context indicator (S3's context switching depends on S2's situated decision making on information

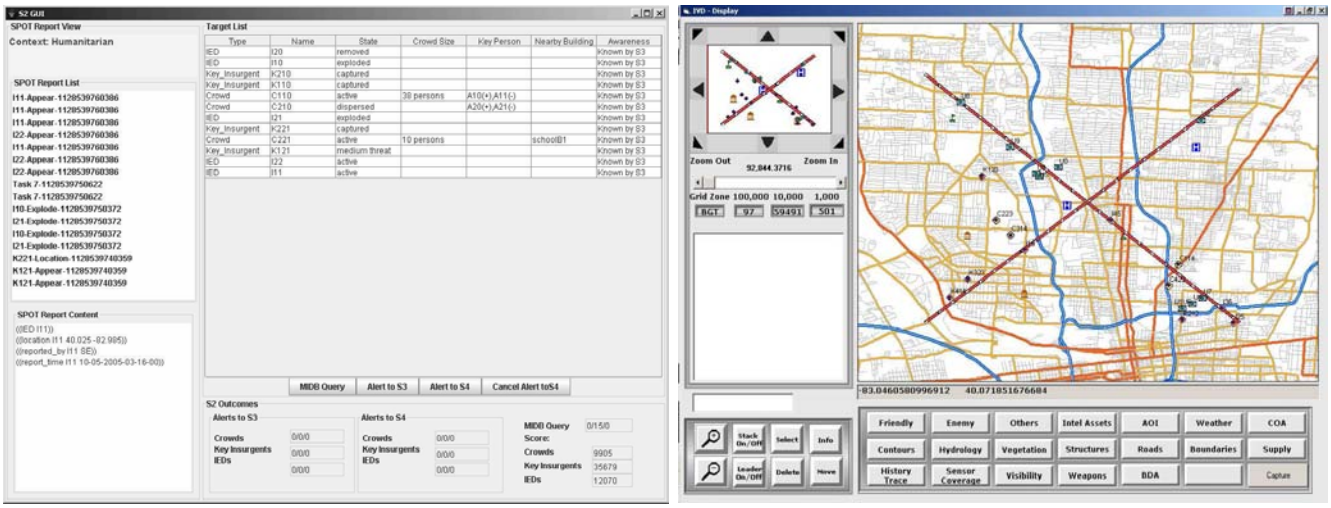


Figure 3: Left: S2 human user interface; Right: Map Display (one for S2, one for S3).

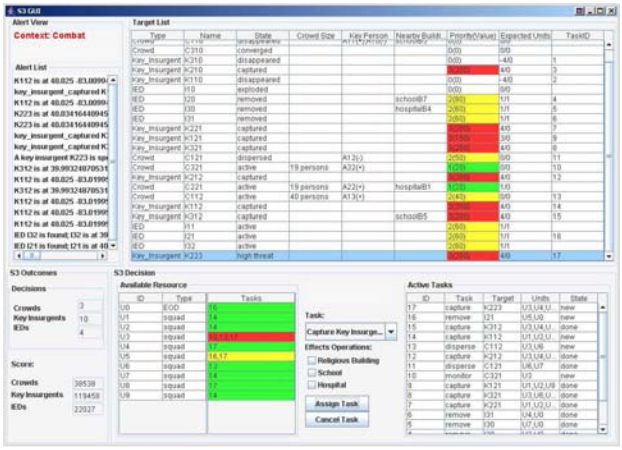


Figure 4: S3 human user interface.

sharing), a list of alerts from the S2 suite, and the S3 suite's current performance. The upper-right list shows target information, similar to S2 human interface. Allocatable resources and their taskloads, command area, and task list are displayed on the bottom-right. S3 subjects can issue 4 types of operations: monitor/disperse a crowd, capture an insurgent, and remove an IED. S3 subjects also need to pay special attention to the effects of their operations; here we simply require S3 subjects to check the type of buildings nearby the selected target, if applicable, when they issue a task. S3 subjects were asked to follow the target values and resource requirements listed in Table 1 when issuing tasks.

The S3 suite's main task is to decide which target to handle first, and how to allocate resources appropriately to multiple active targets, which is a constraint-satisfaction problem. In the multi-tasking environment, S3 subjects need to consider (a) resource constraints–balancing requirements (resource type, amount) among multiple targets; (b) utility constraints–maximizing the total values of targets successfully handled; (c) spacial constraints–others being equal, considered first resources closest to a target; and (d) timing constraints–earlier tasking has a better chance to succeed, due to the uncertain nature of a target's lifespan.

4.2 Result Analysis

We recruited 10 C2 teams (20 human subjects): 8 teams composed of ROTC students, and 2 teams composed of ROTC officers. Each team was tested using the 9 scenarios described in Table 2. The average performance in terms of the total value of targets handled successfully is plotted using a dotted line in Fig. 5(a). It clearly indicates that there is a tasking capacity (limit); the human teams approached the limit roughly in the medium context-switching settings.

This conforms to cognitive studies [17, 10] in general, because both S2 and S3 subjects are limited by their cognitive capacities. While monitoring the situation development from the Map Display, at the same time S2 subjects need to process Spot reports under time stress, to recognize/ gather missing information, and to constantly make decisions on *when* to share information with *whom* about *what*. Similarly, S3 subjects, while being limited by resources and multi-tasking capacity, need to constantly monitor the status of active targets, to decide *when* and *how* to handle *which* target (prioritization, resource (re-)allocation), and to judge whether to cancel an ongoing task if situation changes.

In addition to gaining expert experiences that can be applied to the human-agent studies, critical issues were also revealed from the experiments. For instance, in low-stressed situations some S2 subjects tended to frequently query MIDB much more than necessary, while in high-stressed situations, some S2 subjects simply ignored the spacial information from the Map Display when they should. It was also observed that some S2 subjects had difficulties in switching to the new context, tending to process targets belonging to the last context until completion. Similarly, in high-stressed situations, having no time to check the Map Display, some S3 subjects tended to allocate resources in bundles (e.g., U1-U4), rather than those closest to the target.

Such human "errors" rest on their fundamental cognitive capacities. The performance cannot be improved by simply providing a better human user interface without employing

appropriate technologies to appropriately support or match human cognitive capacities and capabilities.

5. HUMAN-AGENT C2 TEAMS

In this set of experiments, we (1) used an R-CAST agent (S2 agent) to play the role of S2 suite, and (2) used another R-CAST (S3 agent) to act as S3 human operators' decision aid. The purpose is to investigate whether and to what extent cognitive agents, being empowered with expert experiences, could help in improving C2 teams' performance.

5.1 Team Organization

5.1.1 S2 Agent

An S2 agent, playing the role of the S2, acted as the S3's teammate. The S2 agent uses a knowledge base to track the current world state (including targets, S3/S4's information needs, status of their information awareness). Whenever receiving a Spot report, the S2 agent will update the current state of the target under concern, query relevant information from the MIDB, then check whether the target's state has changed significantly (i.e., the change can affect the S3/S4's decisions and decision adaptations). If so, send alerts to whoever affected.

Most importantly, expertise about the S2's operations has to be encoded as experiences in the form acceptable to R-CAST agents. Since the problem domain involves three contexts, the S2 agent also needs to maintain three experience bases. The following is an example experience format used by an R-CAST agent:

```
(Experience      e26
    (Cue         (Close_enough ?crowd1 ?crowd2))
    (Expectancy  (Move_closer ?crowd1 ?crowd2))
    (Anomaly     (Move_away ?crowd1 ?crowd2))
    (Goal        (peacekeeping yes))
    (Action      (Alert_converged_threat)))
```

It says that experience "e26" applies to the cases where two crowds are coming close enough, and it is expected that they are moving closer; if this is the case, the action "Alert_converged _threat" is triggered; if it turned out that the crowds moved away, the action would be terminated.

5.1.2 S3 Agent and S3 Human

For the S3 suite, an S3 agent acted as the S3 human's decision aid. To offer highly acceptable aids for decisions and decision adaptations, we incorporated a recommendation model into the S3 agent based on the observations during and confirmations after the experiments on human-only C2 teams. Such an approach of employing heuristics from human experts to further help them is significant in practice.

The recommendation model has four components. First, the S3 agent can recommend *target priorities* to the S3 human. Based on the advice collected from the S3 subjects in the human-only C2 experiments, the S3 agent was implemented such that it can use colors to show target priorities: red for targets with a high priority, yellow for targets with a medium priority, and green for targets with a low priority. However, there may exist multiple targets with a high priority at the same time. Also, due to the uncertainty of targets' lifespans and the current locations of targets and friendly units, the best target selection may not be the one with the highest priority. Thus, which target to handle first still depends on the S3 human's decision, which can largely impact how the situation evolves and the team performance in the long run. Second, the S3 agent can remind the S3 human about the current resource requirements of each active target (i.e., how many more units are needed). Third, the S3 agent can automate the selection of operation types and the "effects of operation". This releases the S3 human's attention for other activities.

Lastly, based on the following heuristic model, the S3 agent can recommend resources to the currently selected target to support S3 human's *resource allocation behavior*.

We use $1, 2, \cdots, n$ to denote resource types; R_i^o to denote the set of resources of type i; R_i to denote the set of *free* resources of type i; and R_i' to denote the set of *re-assignable* resources of type i. *Re-assignable* resources are those that have already engaged in a task but can be preempted by another task. The current domain has 2 resource types: squads and EOD.

Each target places a resource requirement (cf. Table 1). The resource requirement of target t is denoted by $\overline{r_t} = (r_1(t), r_2(t), \cdots, r_n(t))$, where $r_i(t)$ is the number of needs on resources of type i. For example, $\overline{r_{c121}} = (2, 0)$.

Given a target t, if $r_i(t) > |R_i|$, we need to find re-assignable resources engaged in the on-going tasks. The on-going tasks can be ordered ascendingly by *priorities* up to tasks with priorities no more than the task being considered. Let $target(T)$ be the target of task T, $R(T)$ be the set of resources engaged in T. The priority of a task T with respect to a target t, is determined by its $TValue$:

$$TValue(t,T) = \frac{TargetValue(t)}{\sum_{r_i \in \overline{r_t}} w_i \cdot r_i} \times \frac{TargetRemainTime(t)}{max\{distance(t,\gamma) | \gamma \in R(T)\}}$$

where w_i is the weight of resources of type i, representing the cost of using a resource with type i, and the second factor on the right represents the chance of success when using resources engaged in T to handle target t. Here, $TargetRemainTime(t)$ represents the *estimated* value of how long t will remain before being removed from the simulation.

Resources engaged in task T will be considered re-assignable for t if $TValue(t,T) > \delta * TValue(target(T), T)$,
where a relatively good value for the adjustable variable δ can be obtained by running pre-experiments.

Let $(T_1, T_2, \cdots, T_k)$ be a list of tasks determined in the above way relative to target t, and $(y_i^{T_1}, y_i^{T_2}, \cdots, y_i^{T_k})$ be a list, where $y_i^{T_j}$ is the set of resources of type i from task T_j. Then, resources of type i re-assignable for t is $R_i'(t) =$

$$\begin{cases} y_i^{T_1} & (r_i(t) - |R_i|) \leq |y_i^{T_1}| \\ \bigcup_{j:1..m} y_i^{T_j} & |\bigcup_{j:1..m-1} y_i^{T_j}| < (r_i(t) - |R_i|) \leq |\bigcup_{j:1..m} y_i^{T_j}| \end{cases}$$

Next, given the set $R_i(t)$ of resources of type i available for target t, resources closest to the target t ought to be selected first. Let $D = (D_1, D_2, \cdots, D_x)$, where $\{D_i : (1 \leq i \leq x)\}$ is the partition of $R_i(t)$ by the distance of resources to target t, and D_i in D are ordered by distance ascendingly. That is, resources in D_i have the same distance to t and they are closer to t than those in D_{i+1}. Then, $closest(R_i(t), x) =$

$$\begin{cases} select(D_1, x) & x \leq |D_1| \\ select(\bigcup_{j:1..m} D_j, x) & |\bigcup_{j:1..m-1} D_j| < x \leq |\bigcup_{j:1..m} D_j| \end{cases}$$

which returns x number of resources closest to t, where $select(A, z)$ returns any one of A's subset of size z. Here, it

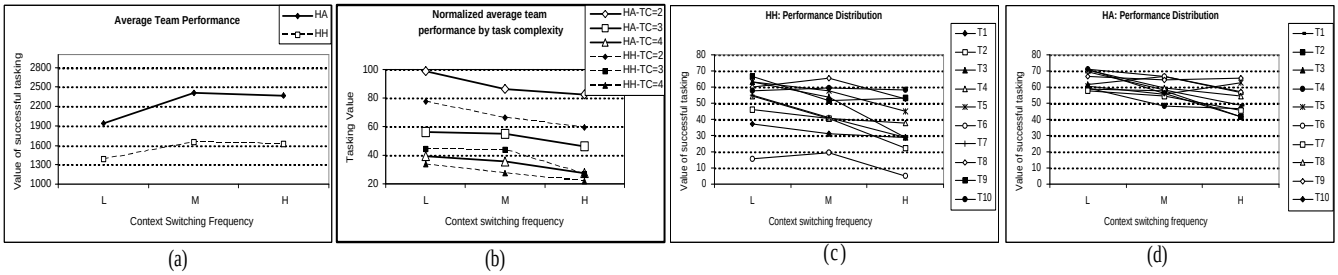


Figure 5: (a) Average performance (in net value); (b) Normalized performance by task complexity, (c) Performance distribution of human C2 teams, (d) Performance distribution of human-agent C2 teams.

doesn't matter whether to consider resources in $\bigcup_{j:1..m-1} D_j$ first, because at least one will come from D_m, which dominates the time it takes for the resources to reach t's location (we assume all the resources move at the same speed).

Now, given a target t and its resource requirement $\overline{r_t} = (r_1(t), r_2(t), \cdots, r_n(t))$, S3 agent follows the criteria below to allocate resources for target t. For each resource type i, (1) if $r_i(t) \leq |R_i|$, return $closest(R_i, r_i(t))$; (2) if $|R_i| < r_i(t) \leq |R_i \cup R'_i(t)|$, return $R_i \cup closest(R'_i(t), r_i(t) - |R_i|)$.

Actually, in cases where $r_i(t) > |R_i \cup R'_i(t)|$, two tasks can be issued separately to meet the resource requirement of t: one task employs resources in $R_i \cup R'_i(t)$, then issue another task later when more resources become available. However, this might mess things up and human subjects may easily forget to return to finish those partially committed operations. We thus left this open: human subjects could choose to split tasks if they could manage their attention.

In experiments, S3 humans could, if they want, override the above four kinds of recommendations from the S3 agent.

5.2 Result Analysis

We recruited ROTC students to form 10 human-agent C2 teams, and each team was tested using the same 9 scenarios. The average performance in terms of total value of targets handled successfully is plotted in Figure 5(a). It suggests that the C2 teams, even assisted by cognitive agents, still have a tasking capacity (limit). However, it also clearly indicates that, compared with human-only C2 teams, the tasking capacity of human-agent C2 teams have been significantly improved (all results discussed were 'significant' at the $p < .05$ level). This suggests that cognitive agents can play a critical role in alleviating the impact of human's cognitive capacity on the performance of decision makings involving multiple contexts.

5.3 Performance Comparisons & Discussion

In order to gain a better understanding of the performance difference, we plotted the *normalized performance* categorized by task complexities (amount of concurrent tasks, cf. Table 2) in Figure 5(b). It shows that the performance of both human C2 teams (dotted lines with filled markers) and human-agent C2 teams (lines with hollow markers) were affected considerably by task complexity. In addition, human-agent C2 teams performed much better when the performance at the same level of task complexity is compared.

Going deeper, we plotted the performance of individual teams in Figure 5(c)-(d). It shows that human-agent C2 teams' performances are relatively stable, while human C2 teams differed enormously. This may demonstrate that cognitive agents could augment human cognitive capacity and improve the overall C2 performance. Additionally, in both cases, the performance deviation became larger as context switching frequency increased. This may suggest that more stressed situations require more competent C2 operators.

Table 3: Decision making behaviors in detail

	Tasks	*FullUses*	*Reassigns*	*W. use*	*W. att.*
HA	32.52	3.84	11.65	0.207	0.336
HH	23.73	3.07	4.97	0.443	0.556

W. use: incorrect resource use; W. att.: incorrect attention

The experiment logs also allowed us to further identify what had contributed to the performance difference. Table 3 gives some descriptive statistics.

With the help of R-CAST agents, S3 subjects could issue more tasks (32.52 vs. 23.73), could use the limited resources more effectively (the number of times where all the resources were engaged: 3.84 vs. 3.07), and were apt to reallocate resources (11.65 vs. 4.97). Such improvement can be attributed to the balance of workload between the humans and agents such that the limits of human cognitive capacity were accommodated. In particular, the impact of S2 subjects' cognitive capacity was eliminated (S2 agent took the role); the impact of S3 subjects' multi-tasking capacity and mental efforts in resource allocation was alleviated.

With respect to the metric of "closer-resource-used-first", S3 subjects in human C2 teams (HH teams) made more incorrect judgments than S3 subjects in human-agent teams (HA teams): 0.443 vs. 0.207. S3 agent's recommendation could be wrong because its world model can be outdated: it only gets updated information periodically from the Simulation Engine. Also, S3 subjects missed most of the attention requirements on the selection of "Effects of Operations", while it was a big change when S3 agent helped S3 subjects: 0.556 vs. 0.336. Again, S3 agents might judge the situation incorrectly due to its outdated world state. Using agents, other kinds of human errors (e.g., wasting resources on non-threatening targets, issuing wrong operations to targets) were significantly diminished or avoided under high-stressed situations.

The results are encouraging. Although the sample space is restricted, the selected human subjects can be taken as representatives of domain experts—they have been trained to do similar operations. It may be argued that ROTC students typically are near the bottom of the expertise hier-

archy. Referring to the subjects as experts might be problematic, because the working memory load for a task differs greatly for experts vs. non-experts. However, as far as this particular experiment is concerned, ROTC students can be taken as domain experts. This was confirmed by our experiment which showed that the performance of the 2 ROTC officer teams were not the best among the 10 teams.

6. CONCLUSION

Multi-context real-time decision making is an extremely challenging problem faced by various real-world application domains. We have started to address this challenge associated with C2 teams operating in complex urban terrain, using the R-CAST cognitive agent architecture as human operators' teammates and decision aids. This paper, focused on the coupling of cognitive agent technology and human-centered teamwork, reported our experimental studies about the impact of RPD-enabled agents on C2 teams' performance in multi-context decision making under stress.

The experiments represent an important step forward in uncovering the nature of real-world problems, because the environment simulated the real domain and the human subjects were recruited from Army ROTC students. The results demonstrated that C2 team performance, while still limited by the human cognitive capacity, could be largely improved when they were assisted by cognitive agents capable of proactive information gathering/sharing and experience-based decision making.

Intelligence analysts need tools and techniques to help protect themselves from avoidable errors [15, 7]. Our experiments demonstrated that RPD-enabled agents can serve as one such tool to achieve reduced cognitive load, enhanced situation awareness, and positive human-agent collaboration. However, in real-world applications where information availability and credibility is concerned, it is expected that C2 teams with both the S2 and the S3 assisted by an R-CAST agent could outperform either of the two team structures considered in this paper, because S2 human's ability to reason about imperfect information can be fully exploited. This is worthwhile to consider in future studies.

7. ACKNOWLEDGMENTS

This work is supported as an FY06 Research Task under the Army Research Laboratorys Advanced Decision Architectures Collaborative Technology Alliance (ARL ADA CTA). Thanks to John Dumer, Holly Ingham and Mark Mittrick from US Army Research Lab at Aberdeen Proving Ground for their supports, and Dr. Frank Ritter at the Penn State University for the valuable comments.

8. ADDITIONAL AUTHORS

Rashaad Jones (College of Information Sciences and Technology, Penn State University, email: `rjones@ist.psu.edu`); Timothy Hanratty (US Army Research Lab, Aberdeen Proving Ground, email: `hanratty@arl.army.mil`); and Laurel Allender (US Army Research Lab, Aberdeen Proving Ground, email: `lallende@arl.army.mil`).

9. REFERENCES

[1] J. Bradshaw, M. Sierhuis, A. Acquisti, Y. Gawdiak, D. Prescott, R. Jeffers, N. Suri, and R. van Hoof. What we can learn about human-agent teamwork from practice. In *Workshop on Teamwork and Coalition Formation at AAMAS'02*, Bologna, Italy, 2002.

[2] J. A. Cannon-Bowers, E. Salas, and S. Converse. Cognitive psychology and team training: Training shared mental models and complex systems. *Human Factors Society Bulletin*, 33:1–4, 1990.

[3] P. R. Cohen and H. J. Levesque. Teamwork. *Nous*, 25(4):487–512, 1991.

[4] X. Fan, S. Sun, M. McNeese, and J. Yen. Extending the recognition-primed decision model to support human-agent collaboration. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 945–952. ACM Press, 2005.

[5] X. Fan, S. Sun, B. Sun, G. Airy, M. McNeese, and J. Yen. Collaborative RPD-enabled agents assisting the three-block challenge in C2CUT. In *Proceedings of BRIMS'05*, pages 113–123, 2005.

[6] X. Fan, J. Yen, and R. A. Volz. A theoretical framework on proactive information exchange in agent teamwork. *Artificial Intelligence*, 169:23–97, 2005.

[7] R. J. Heuer. *Psychology of Intelligence Analysis*. Center for the Study of Intelligence, 1999.

[8] G. A. Klein. Recognition-primed decisions. In W. B. Rouse, editor, *Advances in man-machine systems research*, volume 5, pages 47–92. JAI Press, 1989.

[9] G. A. Klein. Recognition-primed decision making. In *Sources of power: How people make decisions*, pages 15–30. MIT Press, 1998.

[10] G. Kuk, M. Arnold, and F. Ritter. Effects of light and heavy workload on air traffic tactical operations. *Ergonomics*, 42(9):1133–1148, 1999.

[11] T. L. Lennox, T. Payne, S. K. Hahn, M. Lewis, and K. Sycara. MokSAF: How should we support teamwork in human-agent teams? Technical Report CMU-RI-TR-99-31, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, September 1999.

[12] E. Norling. Folk psychology for human modelling: Extending the BDI paradigm. In *AAMAS '04: International Conference on Autonomous Agents and Multi Agent Systems*, pages 202–209, 2004.

[13] E. Norling, L. Sonenberg, and R. Ronnquist. Enhancing multi-agent based simulation with human-like decision making strategies. In S. Moss and P. Davidsson, editors, *Proceedings of the Second International Workshop on Multi-Agent Based Simulation*, pages 214–228, 2000.

[14] J. Sokolowski. Enhanced military decision modeling using a multiagent system approach. In *Proceedings of BRIMS'03*, pages 179–186, 2003.

[15] A. Tversky and D. Kahneman. Judgement under uncertainty: Heuristics and biases. *Science*, 185:1124–1131, September 1974.

[16] W. Warwick, S. McIlwaine, R. Hutton, and P. McDermott. Developing computational models of recognition-primed decision making. In *Proc. of the tenth conference on Computer Generated Forces*, 2001.

[17] C. D. Wickens. Multiple resources and performance prediction. *Theoretical Issues in Ergonomics Science*, 3(2):159–177, 2002.

Camera-based Observation of Football Games for Analyzing Multi-agent Activities

Michael Beetz
beetz@in.tum.de

Jan Bandouch
bandouch@in.tum.de

Suat Gedikli
gedikli@in.tum.de

Nico v. Hoyningen-Huene
hoyninge@in.tum.de

Bernhard Kirchlechner
kirchlec@in.tum.de

Alexis Maldonado
maldonad@in.tum.de

Intelligent Autonomous Systems Group
Technische Universität München, Munich, Germany

ABSTRACT

This paper describes a camera-based observation system for football games that is used for the automatic analysis of football games and reasoning about multi-agent activity. The observation system runs on video streams produced by cameras set up for TV broadcasting. The observation system achieves reliability and accuracy through various mechanisms for adaptation, probabilistic estimation, and exploiting domain constraints. It represents motions compactly and segments them into classified ball actions.

Categories and Subject Descriptors

I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding—*Video analysis*

General Terms

Motion tracking, Analysis of intentional activity

Keywords

video analysis, state estimation, object tracking, motion interpretation

1. INTRODUCTION

In order to realize computer systems that can interpret and analyze cooperative and competitive activity in multi agent systems, we need to realize powerful observation systems for agent behavior. One domain that is intensively studied and has proven to be a very interesting and challenging domain in the multi-agent community is football — in particular in the context of the RoboCup competitions.

In this paper, we investigate the observation of real football games based on video streams provided by a set of TV cameras. This computational problem is interesting because it can be solved with an ordinary setup for broadcasting football games without further additions. Based on the camera image streams the software system computes a representation of the game that enables it to answer questions such as: What are the characteristic offensive plays of the two teams? What are the strengths/weaknesses of a particular team/player? What roles do the players have? Do their capabilities match their roles? Do they achieve their tasks? How does a team create scoring opportunities? What are each players' skills? What is the tactical formation of a team?

Here we consider the problem of generating an action model suitable for answering such questions. The knowledge representation mechanisms needed for inferring such answers and the application of these techniques to the analysis of football games are investigated in a companion paper [3].

Building up a game representation for analyzing multi-agent activity requires a software system to exhibit the following capabilities,

- The reliable and accurate estimation of the positions of the players and the ball.
- The compact representation of motions and their segmentation into actions

The realization of these capabilities is also very difficult for various reasons. The positions, direction, and zooming factor of the camera are not known and must therefore, be estimated from the image stream. Processing the image streams is made difficult by the change in lighting conditions. These conditions change substantially when taking a camera sweep from one part of the field to another one or from one moment to the next when clouds are passing. Other complications include the inaccuracy in depth estimations caused by the low height of the camera positions and the fact that TV cameras are typically placed only on one side of the field resulting in players at the other side being very small in the image. There are also frequent occlusions of players by other ones.

The contributions of this paper are twofold. First, we describe the design, implementation, and empirical analysis of a camera-based observation system for football games. Second, we demonstrate how abstract representations of the game actions can be inferred that enable automatic game analysis systems to represent and reason about the actions in multi agent systems in sophisticated ways. These techniques are described in a companion paper [3].

In the remainder of the paper we proceed as follows. The next section presents an overview of the observation and interpretation mechanisms of the system. Section 3 details the components and methods used for the visual perception of the players and the ball. Section 4 then describes the probabilistic estimation methods that

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

enable the system to keep track of players on the field and estimate their postions more accurately. The interpretation of position data in terms of actions is detailed in section 4. We conclude with a discussion of related work and our conclusions.

2. SYSTEM OVERVIEW

Before we dive into the technicalities of the vision-based game analysis system let us first describe the sensing apparatus of the system and the overall software architecture.

2.1 Physical Setup

The physical setup of the system consists of a set of cameras used to broadcast football games on television. The cameras are pointed and zoomed by a camera man and provide complete image streams. The game observation system passively receives the camera stream without having control over it, and without receiving information about the pan and tilt angles of the camera nor its zoom parameters.

The cameras do only partly cover the playing field and not all players are visible all the time. In addition the cameras usually focus on the surroundings of the ball that is information about other parts of the field are only available at a substantially lower resolution. Another complication is the lack of height of the camera positions. Because the cameras are fixed in a maximal height of about 18 meters the depth resolution of a pixel imaging the opposite side of the field is about 1,5 meters. The sizes of players are only about 35 pixels where the course grained resolution of pixels blurrs the colors. Therefore the appearance of players close to the camera differs substantially from those of distant players. Finally, drastical and abrupt changes of lighting conditions complicate the reliable interpretation of color information.

2.2 Software Architecture

The software architecture of the game analysis system is depicted in figure 1. The system is decomposed into two main components: the model acquisition and the reasoning and model mining component.

The model acquisition component consists of the *visual perception module*, the *blob tracker*, and the *motion and action interpreter*. The *visual perception module* receives multiple streams of camera images and computes for each image the blobs that correspond to players, the referee, and the ball. A blob description contains an $\langle x, y \rangle$ coordinate on the field, a class label that specifies the team the player belongs to or whether it is a referee, and the covariance as a measure of the expected inaccuracy of the estimated position.

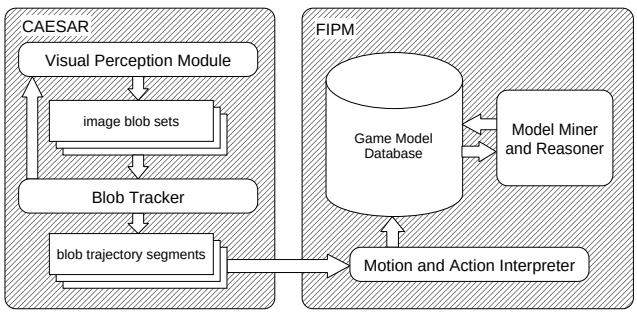


Figure 1: Software architecture of the FIPM analysis system.

The *blob tracker* integrates the observations generated by the *visual perception module* and produces trajectories of the observed object motions and associates these trajectories with the respective players. Using temporal redundancy and coherence the *blob tracker* estimates the player motions much more reliably and accurately. Additional computational tasks that the blob tracker performs are the association of player track identities with blobs and the completion of partly unobserved object tracks through probabilistic guessing.

The final component of the model acquisition component is the *motion and action interpreter*. The interpreter stores the motion data very compactly as a concise motion model and structures the continuous motions into application specific ball actions such as passes, dribblings, and shots. The interpreted motions and ball actions are then stored into the game model database that is used by the *reasoner and model miner* for game analysis purposes.

The *reasoner and model miner* learns situation specific models of the skills and action selection criteria of the football players using the game model database as the relevant set of experiences. It then uses the models in order to predict, analyze, and diagnose the teams' playing behavior.

The FIPM system is implemented as a distributed system running on multiple computers. In the current version the visual perception module for each camera is running on its own computer, so is the tracking, and the motion interpretation module. The modules communicate via Corba.

3. VISUAL PERCEPTION

As we have sketched before the computational task of visual perception is to estimate the 3D position of each player, the referees, and the ball in the camera view. A camera image and the recognized objects and their positions are shown in figure 6. Since the estimated positions are inaccurate the expected accuracy is computed and returned with the estimates.



Figure 2: Output of the visual perception. The players and the ball in the camera view are detected. The players are labelled with respect to the team they belong to. For each player, the referee, and the ball the $\langle x, y \rangle$ position is estimated.

The visual perception of the players and the ball based on a stream of camera images entails a number of difficult computational problems. The first problem is the camera position that gives the system a very flat viewing angle, which causes high inaccuracies in depth estimates. For example, if a player is at the opposite side of the field one pixel in the image corresponds to more than 1,50m depth. Also, if the camera does not zoom in on a player who

stays at the other side of the field the size of the player in the image is about 35 pixels. This reduction in size also results in a considerable change of the visual appearance of players causing the color models of football dresses to exhibit much higher entropies. There is also frequent occlusion especially caused by one player covering another one. Luckily the space occluded by players is quite small because of the distance and position of the camera. Another issue is the rapid change of lighting conditions either due to change of cloudiness or due to the sharp shadow caused by the stadium roof in intense sun shine.

There are, on the other hand, also a number of assumptions that we can make in order to simplify the computational task of visual perception drastically [7]. The first assumption is the *ground plane assumption*. We can assume that all objects that interest us are standing on the field and have roughly the same size. Another important assumption is the *distinctive color* assumption: we assume that objects and their classes can be recognized by their colors. The football field is green and the lines white and the two teams are required to wear dresses that are visually easy to distinguish. We further know the static environment: the form of the field lines and all but a few parameters such as the length and the width of the field. Also since the frame rate is about 25 frames per second and the motions of the observing cameras and the players are not too fast, there is a high temporal coherence between subsequent camera images.

The visual perception of football games is decomposed into three closely interrelated subproblems: (1) the identification of the relevant color regions, (2) the estimation of the pan- and tilt angle of the camera and the zooming parameter, and (3) the identification and positioning of the players and the ball that are in the camera view. In the remainder of this section we will describe our approaches to the solution of these computational problems.

3.1 Color Segmentation

Since football is a spectator sport the playing field, the lines, and the dresses of the players are designed to be visually distinctive. From the visual features that allow for these distinctions color is the most informative one: the field is green, the lines are white, and the players of the teams are required to dress so as to achieve the highest possible contrast.



Figure 3: Computing the regions that are colored football field green.

Thus our first step in processing the camera images is to apply color classification and segmentation to the images. Given a previously learned set of color classes including ones for *field green*, ones for the dresses of the teams, and others, the visual perception module finds the regions of interest by first mapping the image pixels into the respective color class and then grouping the pixels that belong to the same color class into regions using morphological operations for noise elimination. Figure 3 shows the mapping of an image into the color class field green. The white pixels belong to this class the black ones not.

In order to find meaningful regions or blobs, in particular those that correspond to the objects of interest, the visual perception module applies morphological operations to the images resulting from color classifications.This step eliminates noise and makes regions more homogeneous.

In addition, we can characterize the objects of interest through properties of the image blobs they generate. In particular, using the assumptions that players are in upright positions and having contact with the field plane (at least, if we consider image sequences of particular length) we obtain very accurate estimates of the blob sizes based on the $\langle x, y \rangle$-coordinates of blobs in the image. Furthermore, the objects of interest, the players and the ball, have to satisfy certain compactness (ratio between area and perimeter). We can apply these assumptions to filter the data, and to more reliably extract the relevant objects.

Having color regions and color blobs as building blocks we can define complex regions of interest to which we might apply certain image processing operations. For example, in order to look for field lines in a focussed way we consider the image region that is field green, in the area where we expect the field but disregard those regions that are occluded by a player or the referee. This region can be expressed by: *(¬ green ∩ field region) - Team1Regions - Team2Regions - RefereeRegion*.

Robustness of Color Classification. Lighting conditions change when the camera sweeps from one side to the other, when the clouds change, when it begins to rain, etc. For these reasons reliable color segmentation cannot be achieved by learning color classes in advance and then hold them fix during the game. Rather they have to be adapted on the fly, in particular the color class *field green*. Thus we adapt the color class *field green* in an expectation maximization manner. In one step we use the estimates of the camera parameters to identify those regions that must be *field green* given the field model and the camera parameters. The relevant region is: *FieldRegion - Team1Regions - Team2Regions - RefereeRegion - neighborhoods of fieldlines ...* processed by some morphological operators to eliminate holes in the regions. We then take the pixels in these regions to estimate the color class *field green*. This color model is then used for the estimation of the camera parameters. In praxis, the class model can be estimated at a much lower rate than the camera parameters.

3.2 Estimation of Camera Parameters

In order to estimate the 3D position of the players and the ball, we need estimates of the camera parameters, that is its position, the direction it is pointed to, and the zoom parameter. Given the camera parameters and an assumption about the plane in which pixels lie the 3D coordinates of each pixel can be determined. In this section we will investigate the estimation of the camera parameters for a given image sequence.

We will do so by first describing our model of football fields. Then we look at the three-step iterative estimation process, detailed in section 3.2.2.

3.2.1 The Model of the Football Field

The purpose of the field model is the estimation of the exact position of the camera, of the direction it is pointed to, and the zoom of the camera. Based on these camera parameters and the knowledge

of the exact position and orientation of the field plane the visual perception module can accurately determine the $\langle x, y \rangle$ positions of the players on the field (see section 3.3).

The model of the football field is a set of 3D curves where each curve can either be a line or an edge. Edges are curves that separate two neighboring surfaces. Lines are combinations of two parallel curves where the area between the curves is visually distinct from the areas left and right of the line. The field lines are perfect examples of this line notion.

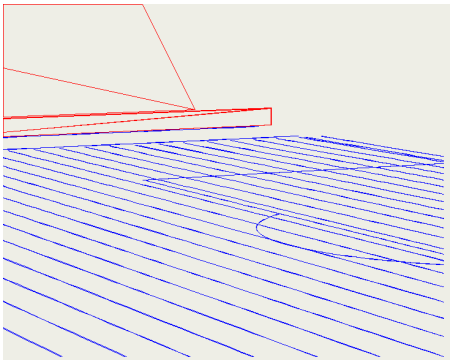


Figure 4: Field model used for estimating the camera parameters. Curve segments are depicted in blue and triangles in red.

The FIPM system uses the field model depicted in figure 4. The model contains the lines on the field. In addition, it contains the boards surrounding the football field and the stands. Both the boards and the stands are modeled as triangles with given position and orientation in space. Using these triangles the visual perception system can accurately determine the 3D position of each pixel that lies in the projection of such a triangle onto the image.

The boards and the stands are included in the field model because for many camera views the field lines alone do not suffice to unambiguously estimate the camera parameters. Using the boards and stands submodels the visual perception module can on the fly determine visual landmarks on the boards and in the stands and use them for estimating the camera parameters even if no field lines are in the camera view.

3.2.2 Estimation Process

We formulate the estimation of the camera parameters as an iterative optimization problem. The opimization process performs in each iteration three steps. In the first step the model is projected onto the image using the predicted parameter values. In the second step the optimizer searches for the image points that correspond to given model points. In the third step, the predicted parameter settings are modified such that we obtain the best match between the model and the image data. The search window for finding the best match is determined based on the expected innaccuracy of the predicted parameters. These three steps are depicted in figure 5.

Since the frame rate of the camera is about 25Hz (50 Hz interlaced) the change from one image to the following one is typically very small, the exceptions being camera sweeps following very long and hard passes. We will return to this issue later in section 3.2.3. So in the typical case, where the changes are small the predictions are also very accurate allowing for the use of small search windows.

The search methods for correspondences are specific to the different types of curves which allows for the realization of a high performance hybrid correspondence search. This means that we have curve segment specific "curve line" detectors. For example, the lines on the field are detected by looking for edges in the *field green* region, the edges resulting from lawn mowing look for points along the search perpendicular such that brightness variation left and right of the point becomes mininmal, that is the points to the left have a very similar brightness and so do the pints on the right. Other mechanisms for finding correspondences are taken for the edges between the field and the surrounding boards.

Given the correspondences we perform an optimization of the camera parameters such that the distance between corresponding points becomes minimal. This is done by a Newton iteration step. The system also estimates the covariance of the parameter values which captures the expected accuracy of the estimation outcome. The camera parameter values resulting from this optimization step is returned as the estimate of the parameters.

3.2.3 Increasing Robustness

In order to estimate the camera parameters reliably and accurately even if the field lines in the camera view do not suffice for an unambiguous estimation we generate visually distinctive landmarks on the fly and track them over sequences of camera images.

For this purpose, we first generate regions of interest by projecting the triangles of the field model onto the image. These are the regions where we can determine the 3D coordinates of the surface that generated the pixel. To these regions of interest we apply the feature detection and tracking method of Shi and Tomasi [14].

Even with these extensions the estimation of the camera parameters will sometimes fail, in particular when the camera man makes fast sweeps. To deal with failures in parameter estimation we run a monitoring process in parallel with the estimation process. This monitoring process signals a failure whenever the quality of the match between the predicted parameters and the parameters that result in the best local fit fall below a specified threshold. In this case a reinitialization of the camera parameters is triggered.

3.2.4 Empirical Results

At the current state of implementation the system can keep track of the camera orientation and the zoom parameter for up to 2,5 minutes, depending on the speed of camera motion and zooming and the visible components in the camera view. As far as we can tell the cases where the system looses track are caused by our model of the field beeing too inaccurate. In general, the length and the width of the field can vary substantially from stadium to stadium. An accurate and reliable estimate of these parameters from an image is not possible because if the camera view covers enough of the field then the field lines of the opposite side are blurred too much. Thus, so far we work with field models that are locally consistent with respect to the assumed camera position but not globally accurate. We currently, work on mechanisms that are less sensitive to the lack of globally accurate field models.

When successfully tracking the camera position the average accuracy of the field model being back projected onto the image is typically within a pixel occasionally 3-4 pixels. The system can automatically recover from lost tracks if the estimate is still within the convergence area of the model fitting procedure. For the remaining situations we currently develop a monitoring mechansm that detects lost tracks and reinitializes the system as soon as the camera view allows for an unambiguous match of the field lines.

At the moment processing an image can still take up to more than a second. We expect that by applying faster methods, and exploiting more prior information, we can reduce the processing time and bring the system close to frame rate.

3.3 Blob Recognition and Localization

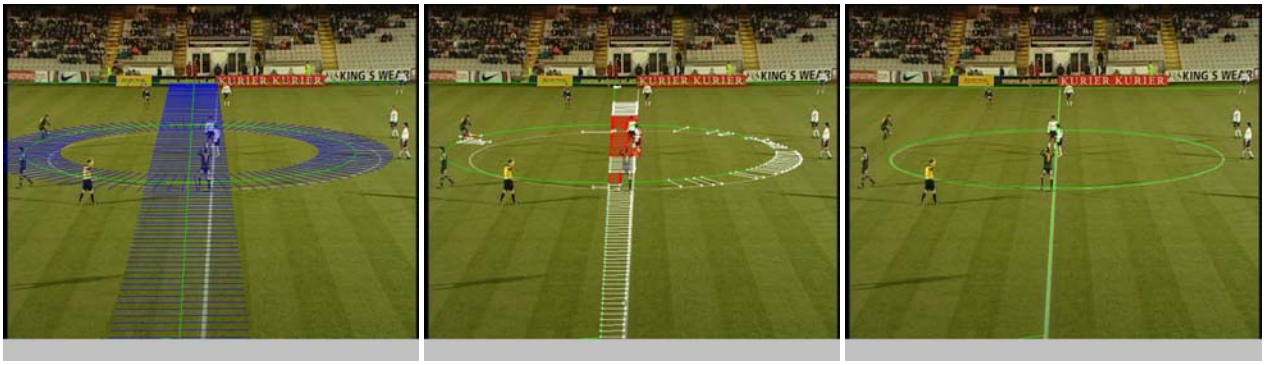


Figure 5: Estimating the camera parameters by taking the current parameters, projecting the field model using the current parameters onto the image (left), finding the correspondences between model points and the imagepoints (middle), and adjusting the camera parameters to achieve minimal errors (right).

The visual perception module gets all blobs in the field green region as its input. These are obtained through the regional expression: InRegion(field) $\cap$ $\neg$FieldGreen $\cap$ $min \leq Compactness \leq max \cap min \leq Size(blob, distance) \leq max$

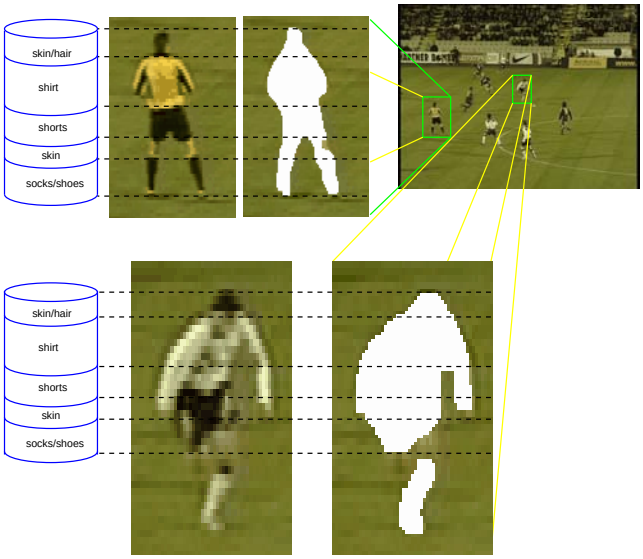


Figure 6: Player blobs segmentation. The player blob detection algorithm finds compact regions that are not field-green. The segmentation of players close to the camera is fairly accurate, but the situation is harder for the distant players because of the course grainedness, which causes some of their pixels to look field-green. Then they are segmented into two regions.

The next step is the blob classification. Blobs are classified into: the ball, the players of the one and the players of the other team and the referees. All other blobs on the field are ignored. The classification of the blobs into the different categories is straight forward. We first apply size constraints to the blobs and then identify the dress colors of the players. These color models are learned and updated during the game. The biggest complication are the partly occluded players that result in bigger blobs with possibly not identifiable color classes. These complications are resolved using the information provided by the object tracking system (see next section).

Another module monitors the blob sizes in the camera image. If the blob size is big enough, the blob identifier attempts to read the player number — if possible. If the player number is recognized the information is passed to the object tracker as evidence about the players' identity.

Given a blob that is inferred to be a player or ball the FIPM system estimates the $\langle x, y \rangle$ of the corresponding object on the field. To do so, we estimate the 3D position of the center of gravity of a player which can be much more reliably and accurately estimated than the position where the player stands on the field. The gain in accuracy and reliability has been empirically studied in several experiments.

Surprisingly, the player blob detection and localization was substantially less robust and accurate than we initially expected. This is explained by a number of reasons including the motions of the legs and the typical colors of skin and player dresses varying substantially. In particular as the players in the image are smaller and more distant a pixel in the image covers a much larger area in the world. Therefore many of the pixels depicting the player have a high fraction of field green and therefor cannot be segmented easily from the background.

To deal with these issues we have then implemented blob recognition using predictions made on the basis of previous observations (see next section). In this method the perception module receives hypotheses about players from the state estimation system and then validates and rejects these hypotheses based on the image. This way we can concentrate the player recognition mechanisms on selected regions of interest and use much more informative expectations about the blob size and the color distribution within the blob. Using this prior knowledge the accuracy and reliability of player recognition is increased substantially while at the same time reducing the computational cost for the player recognition.

4. PLAYER TRACKING

The detection of players and their classification based on single images is unreliable. Players might violate size restrictions when bending down or when colors get blurred due to fast camera motion. Sometimes players are hallucinated due to specular reflections in the field that make regions appear to be not field green.

4.1 Multi-Object Tracking

To reliably estimate the positions and motions of the players we apply probabilistic multi object tracking algorithms to keep track of player positions.

We use an extension of Reid's Multiple Hypothesis Tracking (MHT) algorithm [11, 4] that has been further developed by Schmitt and his colleagues [12, 13]. Using probabilistic motion and camera models the MHT maintains probabilistic estimates of the players positions and update these estimates with each new observation. The computational structure of the algorithm is shown in Fig. 7. An iteration begins with the set of hypotheses of player states $H^k = \{h_1^k, \ldots, h_m^k\}$ from the previous iteration k. Each h_i^k is a random variable ranging over the possible positions of a player on the field and represents a different assignment of measurements to players, which was performed in the past. The algorithm maintains a Kalman filter for each hypothesis.

```
algorithm MULTIPLEHYPOTHESISTRACKING()
 1  let Ĥ^k = {ĥ_1^k, ..., ĥ_{m_k}^k}  % pred.hypos.
 2    Z(k) = {z_1(k), ..., z_{n_k}(k)}   % ob.feat.
 3    H^k = {h_1^k, ..., h_{o_k}^k}  % new hypos.
 4    X^{k-N}  % world state at time k-N.
 5  do for k ← 1 to ∞
 6    do Z(k) ← INTERPRETSENSORDATA();
 7       Ĥ^k ← APPLYMOTIONMODEL(H^{k-1}, M);
 8       for i ← 1 to n_k
 9       do for j ← 1 to m_k
10             do h_ij^k ← ASSOCIATE(ĥ_j^k, z_i(k));
11                COMPUTE(P(h_ij^k|Z(k)))
12       for j ← 1 to n_k
13       do H^k ← H^k ∪ {GENERNEWHYP(z_j(k))};
14       PRUNEHYPOTHESIS(H^k);
15       X^{k-N} ← {x_1^{k-N}, ..., x_{o_{k-N}}^{k-N}}
```

Figure 7: The multiple hypothesis tracking algorithm.

With the arrival of new sensor data (6), $Z(k+1) = \{z_1(k+1), \ldots, z_{n_{k+1}}(k+1)\}$, the motion model (7) is applied to each hypothesis and intermediate hypotheses $\hat{h}_i^{k+1}$ are predicted. Assignments of measurements to players (10) are accomplished on the basis of a statistical distance measurement, such as the Mahalanobis distance. Each subsequent child hypothesis represents one possible interpretation of the set of observed players and, together with its parent hypothesis, represents one possible interpretation of all past observations. With every iteration of the MHT probabilities (11) describing the validity of an hypothesis are calculated. Furthermore for every observed player a new hypothesis with associated probability is created (13).

Obviously, the heart of the MHT algorithm is the computation of the likelihood of the different hypothesis-observation associations, $P(h_{ij}^{k+1}|Z(k))$, in line 12 of the algorithm in Algorithm 7. Let Z^k be the sequence of all measurements up to time k. A new hypothesis of a player at time k is made up of the current set of assignments (also called an event), $\theta(k)$, and a previous state of this hypothesis, h_j^{k-1}, based on observed features up to time step $k-1$ inclusively. We can transform the probability of a player's hypothesis $P(h_i^t|Z^k)$ using Bayes' rule and the Markov assumption in an expression that can be obtained more easily:

$$P(h_i^k|Z^k) = P(\theta(k), h_j^{k-1}|Z(k), Z^{k-1}) \quad (1)$$

$$= P(\theta(k), h_j^{k-1}|Z(k), H^k) \quad (2)$$

$$= \alpha * P(Z(k)|\theta(k), h_j^{k-1}, Z^{k-1}) \quad (3)$$
$$P(\theta(k)|h_j^{k-1}, Z^{k-1}) P(h_j^{k-1}|Z^{k-1})$$

According to Bar-Shalom and Fordham [1], this equation can be transformed into an expression over a set of parameters, and they allow us to fine tune the calculation of probabilities to exploit the domain constraints. These parameters include ones for prior probability mass functions of the number of spurious measurements and of new detection of players, probabilities of detection and termination of a track originating from a particular hypothesis, model the declination of an unobserved hypothesis probability over time, and total numbers of spurious measurements and new detections of players. The exact form of the transformed data association probability and a more thorough explanation of its terms and parameters can be found in the papers by Cox and Hingorani [4] and Schmitt et al. [13].

In our tracking system we dynamically modify these parameters for each track and measurement according to the relevant situation. For example, it is highly unlikely that a player disappears while being well observed by the camera and in the middle of the field. The likely explanation is that the player is being obstructed by another one and the detection module is having problems detecting him/her. In this case, we set the parameters accordingly to favour the continuation of the track without measurements, hoping to detect the player in the next images. See section 4.2 for further discussion of the used constraints.

4.2 Exploiting domain constraints

The nature of our tracking task allows us to make assumptions that do not hold in the general case of multi object tracking. Under these assumptions we are able to substantially increase the robustness, accuracy, as well as reduce the required computational resources by specializing the estimation and inference techniques to the characteristics of the application. We will list some of these assumption that result in substantial improvements of the algorithm below.

The first and obvious way of increasing the performance of the MHT is to set the parameter of the formula in situation specific ways. For example, players cannot magically appear or disappear in the middle of the camera frame. This constraint is implemented by adjusting the average time of new track appearance and track termination. New tracks are to be primarily created at the boarders of the camera frame. The same holds for the termination of tracks. Making these assumptions enables the system to substantially prune the hypotheses tree and therefore allows for more thorough analysis of the individual hypotheses. This is done by maintaining hypotheses trees of greater depth.

Another improvement is the handling of occlusions of one player by another one. This could be handled by increasing the time that a track survives without being supported by additional observation. A more controlled way of "remembering" occluded players is to generate virtual observations from hypotheses that the system knows to be occluded. A similar mechanism was used by Fox and his colleagues [6] to avoid collisions with known obstacles that were invisible for the robot sensors.

Another twist of the tracking problem is that we can assume that the players all move in the same 3D plane. Under these conditions a player is closer to the camera if and only if his position in the image is lower than that of the other one. This qualitative relationship holds independently of the inaccuracy of our estimate of the camera position and can be exploited to reject many data associations that are impossible with respect to this constraint.

These and other modifications to the MHT algorithm allow for reliable and accurate player tracking in situations where the more general version of the algorithm is doomed to fail [7].

At the moment our algorithm generates maximally extended trajectories for which the system is sure that they are generated by the

same player. The player identity itself is specified manually.

In particular the semi-automatic inference of the player identity will be a focus of our future research. The tracking system will use prior distributions of player positions, knowledge about the player roles, occasionally read player numbers, the identity of players that are known in order to estimate a probability distribution over the identity of a given but unknown player.

5. MOTION AND ACTION RECOGNITION

Motion and action recognition abstracts away from the details of the estimated position data and the track segments generated by the Multi Hypothesis tracking algorithm. The raw data are transformed into compact motion models and then classified into ball actions [2]. These computational tasks are detailed in this section.

5.1 Motion Interpretation

The motion interpreter computes motion models for the players and the ball *o* that are sequences of motion segments $m = m_1, m_2, ..., m_n$. The trajectory of an object is represented as a piecewise defined function *f* that maps time indices into the respective *x* and *y* coordinates of the object. The function f is defined for all t with $t_1 \leq t_2$ and $f(t)$ returns the position of o at t. The individual tuples m_i have the form $\langle o, t_1, t_2, p_1, p_2, f : \mathbb{T} \rightarrow \mathbb{P} \rangle$ where o denotes the object (ball or player number), t_1, t_2 the starting and end time point of the motion segment, and p_1, p_2 the respective start- and end position.

The motion model also contains the set of all ball contacts, ball out of bounds, and referee whistles (game interruptions and continuations) as instantaneous events. These events are asserted manually. Ball contact events are those where the ball is accelerated due to the action of a player. Ball out of bounds events occur if the ball leaves the football field.

As the motion model we have chosen piecewise linear models. This model is very efficient to compute and the motion tuples are very compact. The linear modeling of motions also simplifies the computation of distances, intersections, and other derived information. The accuracy can, as in the other models be adjusted through the segmentation thresholds.

The motion interpreter maps the position data into a sequence of motion tuples, where each motion tuple describes a uniform motion segment. The interpreter works iteratively. If it has a partial segment m_i it represents this partial segment as a motion function. It then uses this motion function to predict the next position p_j. If the difference between the predicted and the observed next position is smaller than a threshold ϵ_1 then m_i is extended to cover p_j and the motion function is updated accordingly. Otherwise m_i is asserted to the motion model as a complete motion segment and a new motion segment m_{i+1} is started.

Instantaneous motion events are currently recognized mainly manually. Referee whistles are explicitly asserted. Out of bounds events are detected by intersecting the linear motion segments with the field lines. Finally, the ball contacts are asserted by hand.

5.2 The Episode Model

The motion model is further abstracted into the episode model.

In the episode model we consider the episodes of ball movement to be primary. This is natural because people watching football and summarize plays in football games primarily as a sequence of ball actions. Actions away from the ball are included only if they become relevant for the ball actions later.

We distinguish three different classes of ball actions: keeping control over the ball, passing, and shooting. If ball actions had very high success rates we could recognize them using simple rules. A *ball possession* is a sequence of ball contacts of the same player. A *pass* is a ball contact of one player followed by a ball contact of another player (hopefully a team mate). A *shot* is a ball contact of a player followed by an out of bounds event, where the out of bounds event occurs in or close to the goal.

Unfortunately, the high technical difficulty of playing football makes such simple classification rules very inaccurate. Control over the ball, in particular with opponents around, is very difficult and therefore actions do not only have nondeterministic effects but often fail completely: the player looses possession of the ball. It is often impossible to distinguish whether a lost ball is caused by a bad pass, shot, or dribbling. In our approach we deal with these ambiguities with probabilistic reasoning techniques.

We consider an *episode* e to be a triple $\langle \langle m_i, ..., m_j \rangle, se, fe \rangle$ where $m_i, ..., m_j$ is a sequence of motion segments, se is the starting and *fe* the finishing event. To be an episode the triple must satisfy the following requirements: (1) m_i is started by ball contact se of player p. (2) m_j is ended by *fe*, which is of the type ball contact, referee whistle, or out of bounds. (3) All events e that occur are ball contacts of player p. (4) $m_i, ..., m_j$ is the maximum sequence of segments that satisfies (1) to (3).

For the recognition of episodes we use a small finite automaton that starts a motion segment sequence if it receives a motion segment starting with a ball contact of player p. The automaton stays in its intermediate state as long as the motion events are ball contacts of player p. It terminates for all other motion events. All accepted motion sequences are episode candidates.

The episode model differentiates between passes, shots, and extended ball possessions of a single player. The main problem is the classification of football actions that have failed. To deal with ambiguities in the classification of failed actions we apply classification rules that assign classes to episodes with a subjective probability. To get a principled way of stating such rules we apply decision tree learning . Given a set of classified examples a decision tree learning algorithm learns a set of classification rules with the objective of making the rules as general and accurate as possible. Decision tree learning algorithms also estimate the expected accuracy of the classification rules.

When we apply decision tree learning to the acquisition of classification rules for football actions two problems must be solved. First, we need a suitable feature language for representing episodes such that their representation correlates well with their classification. Second, we must provide the examples that we need for learning. The main problem here is that we need to know the intentions of the players to provide examples of ball actions and their classification. For the feature language we have used the following features: the duration, the number of ball contacts within the episodes, whether ball possession changed from one player to another one, from one team to the other one, the dominant motion direction, and the average velocity of the ball.

We obtained the best results by classifying at the toplevel into the action classes pass, dribbling, and shot and adding one additional class "lost ball" that comprises the episodes where ball possession goes from a player to the other team, from a player to the out, and episodes interrupted by referee whistles. In this approach, we have the advantage that we can specify all toplevel classes crisply. The top-level classification rules are complete and classify uniquely. That implies that every episode candidate is classified as an instance of exactly one top-level class. The biggest assumption is that the classification assumes that if a football action was successful the player has intended it. We believe that by making this assumption the resulting accuracy is higher than it would be when learning the classes from examples. We will reevaluate this assumption when

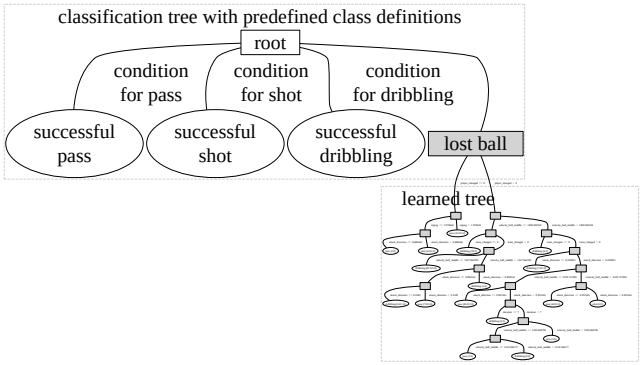


Figure 8: Classification tree. To classify the ball actions into different action classes we use a classification tree consisting of defined conditional leafs for successful actions and a learned decision subtree for lost balls.

we have comprehensive data from real games.

In this approach we apply decision tree learning only to the class of lost ball possession. A sample classification rule that we obtained is: A failed pass is an episode in which the ball is played to the front but not in the accurate direction of the opponent goal, the ball velocity is smaller than that of shots, and the ball possession changes from one team to the other.

6. RELATED WORK

A number of observation systems for football games and sport games have been developed, most of them in the sport sciences and some of them even commercially. Those systems are typically characterized by their requirements for extensive manual data entry by operators. Intille and Bobick [9, 8] have developed a seminal visual observation system for American Football and this is the closest approach to ours. Their system include mechanisms for action, formation, and play recognition. Differences are caused by the different natures of the game. American football is structured into modular, preplanned plays with failed actions (interceptions and turnovers) being exceptions, players having very specific roles in plays, and the ball being held most of the time. In the real football these characteristics are not met and complicate the visual observation drastically. Another difference is our emphasis on accurate estimation of player positions which is the key for the recognition of game situations such as scoring opportunities, players being under pressure, passing opportunities, etc.

The research work in computer vision that is applicable to game observation is too big to be discussed in detail. Perhaps most relevant is the work of Malik's and Forsyth's vision group. In particular ideas of their work on recognizing actions at a distance [5] and the learning and application of appearance models for tracking human action [10] have been incorporated into our system.

7. CONCLUSIONS

This paper has described the nuts and bolts of observing multi agent systems for analyzing agent behavior. We have considered a particular application domain: the visual observation of football games. Football, in particular in the disguise of RoboCup has become a challenging testbed for agent technology.

The observation and interpretation system described in this paper is a part of a larger system. The analysis of Football games based on position data is discussed by Beetz, Kirchlechner and Lames [3]. Kinds of action models that are proposed in this paper are also used for making autonomous robots "action-aware" [15].

We plan to showcase the vision-based game observation system for games of the Football World Championship 2006 in Germany.[1]

8. REFERENCES

[1] Y. Bar-Shalom and T. Fortmann. Tracking and data association. Academic Press., 1988.

[2] M. Beetz, S. Flossmann, and T. Stammeier. Motion and episode models for (simulated) football games: Acquisition, representation, and use. In *3rd International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS)*, 2004.

[3] M. Beetz, B. Kirchlechner, and M. Lames. Computerized real-time analysis of football games. *IEEE Pervasive Computing*, 4(3):33–39, 2005.

[4] I. J. Cox and S. L. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(2):138–150, 1996.

[5] A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *IEEE International Conference on Computer Vision*, pages 726–733, Nice, France, 2003.

[6] D. Fox, W. Burgard, S. Thrun, and A. Cremers. A hybrid collision avoidance method for mobile robots. In *Proc. of the IEEE International Conference on Robotics and Automation*, Leuven, Belgium, 1998.

[7] I. Horswill. Analysis of adaptation and environment. *Artificial Intelligence*, 73(1-2):1–30, 1995.

[8] S. Intille. *Visual Recognition of Multi-Agent Action*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1999.

[9] S. Intille and A. Bobick. Recognizing planned, multi-person action. *Computer Vision and Image Understanding*, 81:414–445, 2001.

[10] D. Ramanan and D. Forsyth. Finding and tracking people from the bottom up. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '03)*, 2003.

[11] D. Reid. An algorithm for tracking multiple targets. IEEE Transactions on Automatic Control, 24(6):843–854, 1979.

[12] T. Schmitt, M. Beetz, R. Hanek, and S. Buck. Watch their moves: Applying probabilistic multiple object tracking to autonomous robot soccer. In *The Eighteenth National Conference on Artificial Intelligence*, Edmonton, Canada, 2002.

[13] T. Schmitt, R. Hanek, M. Beetz, S. Buck, and B. Radig. Cooperative probabilistic state estimation for vision-based autonomous mobile robots. *IEEE Transactions on Robotics and Automation*, 18(5), October 2002.

[14] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[15] F. Stulp and M. Beetz. Action awareness – enabling agents to optimize, transform, and coordinate plans. In *Accepted for the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2006.

[1]The research reported in this paper is partly funded by the Deutsche Forschungs Gemeinschaft.

Modeling one Human Decision Maker with a Multi-Agent System: the CODAGE Approach

Jean-Daniel Kant
Université Paris 6
Laboratoire d'Informatique (LIP6)
8 rue du Capitaine Scott
75015 Paris, France
Jean-Daniel.Kant@lip6.fr

Samuel Thiriot
Université Paris 6 and France Télécom R&D
Laboratoire d'Informatique (LIP6)
8 rue du Capitaine Scott
75015 Paris, France
Samuel.Thiriot@francetelecom.com

ABSTRACT

In this paper, we propose an agent-based model of human decision making: CODAGE (Cognitive Decision AGEnt). Classical Decision Theories have been widely used in multi-agent systems, but imply a too rational behavior when faced with real-world human data. Moreover, classical model usually exceeds human capabilities. Therefore, we derived our decision model from several cognitive psychological theories (e.g. Simon's decision theory, Montgomery's search of dominance structure, etc.) to take human bounded rationality into account. While most of existing cognitive agents use the BDI framework, we propose a new kind of architecture. In the CODAGE model, the decision maker is modeled by an entire multi-agent system, where each agent is in charge a particular sub-process of the whole decision. The architecture is intended to be as generic as possible. It could be viewed as an agent-based decision framework, in which different decision heuristics and biases could be implemented. We illustrate this approach with a simulation of a small experimental financial market, for which our model was able to replicate some human decision behaviors.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Human Factors

Keywords

Cognitive Modeling, Decision Making, Bounded Rationality

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

Multi-agent systems (MAS) have been already used to model human decision making processes (e.g [19, 17]), and many successful applications have been derived from these works in various fields (e.g. economics [14], electronic commerce [7]). MAS seem appealing for human decision making because they capture the two levels of decision processes: the *individual level* (i.e. agent level), where each decision process could be modeled, from the reactive to the more cognitive one ; and the *collective level* (i.e. system level), where one has to model interactions among agents, communication between them, coordination, etc. For instance, in the field of Computational Economics, most works used rather reactive (zero-intelligence or zero-intelligence plus) agents and large-scale interactions in order to exhibit interesting market properties [14]. Other works would use more cognitive agents, mostly based on the BDI (Belief Desire Intentions) architecture [17].

When reactive (or zero-intelligence) agents are used, the emphasis is made on *emergence*: from a simple individual computation process, a global collective behavior emerges. This is consistent with the Artificial Life principles, stating that most of the complex phenomenon we observe in our world should be built from the bottom-up. This approach offers a variety of behaviors and phenomena at the macroscopic level. For instance, in the case of Financial Market Simulations, with few equations, reactive models could lead to equilibrium, oscillations, non equilibrium states, bubbles, that is to many phenomena observed in the real world at the macroscopic level. However, when one needs to explain these phenomena at the agent-miscroscopic level, the reactive approach suffers from its limitations to model the information processes the human agent may have used in the real world. In order to understand and explain complex social human functions (real markets, organization dynamics, coalition formation), more complex (i.e. more cognitive) models of human mind and actions are needed [4].

In the case of decision making, where can this more complex (more cognitive) theory be found ? In the last years, Classical Decision Theory and Game Theory have been used to design multi-agent systems. This includes classical (e.g. utility-based) theory models, extension of utilities, and Markovian Decision Processes (see [19] for a review), and yields several interesting applications. However, most of them are not compatible with the limitations of human capabilities, as stated by Simon with his concept of bounded rationality. Decision and Game Theories usually imply op-

timization processes, while agents are to explore a quasi-infinite (or exponential-growth) search space. And even if one relaxes this request of optimality, cognitive psychology has shown that humans violate most of the principles underlying Decision Theory Models. These works in psychology have been initiated by the seminal theories of Kahneman and Tversky, where the focus is made on the *heuristics* a decision maker used and the *biases* he/she had during a decision process [10]. Many recent models of decision making are based on these psychological facts, and focus on information processing [13].

This is the path we want to take. To do so, we follow a methodology based on what we called earlier *psychomimetism* [11]: (i) we need a cognitive model of decision based on psychological facts that have shown to be consistent with experimental observations, and (ii) we need to implement this cognitive model within an artificial system as faithfully as possible, so that the behavior of the artificial system gives some insights on the real-human cognitive process.

The remainder of this paper is organized as follows. In the next section, we present the cognitive model used in CODAGE, and then we give its implementation as a MAS architecture in section 3. To illustrate how CODAGE works, we present some results from an experimental financial market simulation in section 4. Finally, we conclude this paper with a discussion on our results, limitations and future work.

2. COGNITIVE MODEL

2.1 Simon's Descriptive Approach

There are three basic types of models: descriptive model, prescriptive and normative. Descriptive models tell us what people do or have done, prescriptive models suggest what people should and can do, and normative models tell us what people should ideally do (in theory). A typical example of normative theory is the Classical Decision Theory we mentioned above. In this paper we focus on *descriptive* theory, that is we try to model real-world human decision making. Broadly speaking, decision making could be defined as the cognitive process of selecting a course of action from among multiple alternatives. However, it should not be reduced to a simple selection of prototyped behavior or a case-based reasoning process, that is to a "déjà vu" kind of strategy. Decision Making is a *construction process*: the decision is built based on complex information processing mechanisms, and deals with various forms of knowledge representations (e.g. features, criteria, rules, etc.). Moreover, this process implies a commitment to the selected action.

To clarify what a decision making process is, we adopt the framework introduced by Simon in 1960 [22], as depicted in Figure 1. The decision process includes three steps. *Intelligence* means here the understanding of the decision situation, that it to identify what the decision is about (the need for a decision), the problem we have to solve and the context. Once this environment has been searched, the *design* phase begins, where a deep analysis of the problem domain and context leads to the identification of all possible options and alternatives. The final *choice* phase is the selection of the most appropriate course of action from the set of possible alternatives produced by the design phase.

The Choice phase may include an optimization process (choosing the best alternative) or not. According to the Satisficing model proposed by Simon[21], the decision maker will chose the alternative that exceeds a particular criterion. Not surprisingly, the Satisficing model is fully consistent with Simon's idea of human Bounded Rationality[20]: humans have limited cognitive capacities, they usually are not able to optimize in most situations. A descriptive model of decision-making should take these limitations into account.

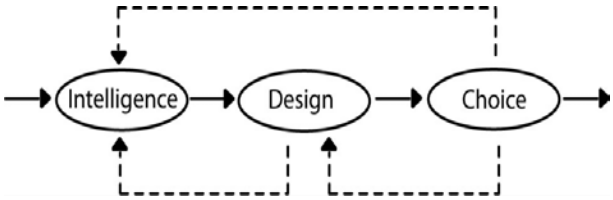


Figure 1: Decision Making as a three-step process (from Simon, 1960)

Finally, one should note that the decision process depicted in Figure 1 is not linear, as it includes feedback loops at each phase. For instance, the alternatives construction at the design phase might need further intelligence. Moreover, each phase incorporate a decision-making process itself, making Simon's decision model a quite complex system. From this framework, many models have been proposed (e.g. see [6, 13] for some reviews). Let us now present our decision model we implemented in CODAGE, which brings a synthesis of several cognitive theories.

2.2 Our decision model

The decision model implemented in CODAGE starts from the three phases of Simon's descriptive theory we described above:

Intelligence: due to limited perception abilities and thanks to its experience[1], the decision maker has the ability to quickly select the important informations. For instance, he/she might be more attentive to an information concerning himself or inducing some emotional reaction. The perception of the decision environment is based on a highly *selective attentional process*. The importance of attentional process in our model partly derives from the framing effects, as first observed by Kahneman and Tversky, where the formulation of a problem drastically influenced decision strategy [23]. This is because subjects focus primarily on a limited number of aspects (e.g. gain or loss in Kahneman and Tversky's experiments) and give the highest importance at these aspects to design alternatives and choose one of them. The concept of *salience* of information implements this principle (Cf. section 3.1.2). Moreover, we incorporate an *anchoring* mechanism, where some information is not continuously encoded : there is a set of relevant discrete value around which the decision maker anchors his/her judgment. This is consistent with the anchoring and adjustment heuristic observed in psychology of decision making [10].

Design: due to its bounded rationality, the decision maker cannot represent the whole world, he/she cannot produce all consequences and all possible actions. However, he/she needs an ability to anticipate short-term events, that is not

[1]As in many modeling works on decision making, we assume our decision maker to be what psychologist called an *experienced* subject. This subject is familiar with the decision task and has developed particular skills to solve similar problems. An experienced subject is not necessarily an expert, but never a novice.

only design a set of alternatives, but also produce some consequences of these alternatives. This will be implemented in CODAGE with a partial *tree of alternatives* where only some branches will be computed and explored (Cf. section 3.1.3).

Choice: in order to chose a satisficing alternative without requiring an optimization process, we adopt a heuristic proposed by Montgomery, the Search for Dominance Structure (SDS) [16]. Broadly speaking, the dominance structure is made of selected attributes that represent the good arguments by which one alternative dominates the others. The decision maker processes the information (the set of attributes) in order to find a dominant alternative. If one is found, it is chosen. Otherwise, he/she will look at a different set of attributes and alternatives. We present in section 3.2 our agent-based implementation of SDS.

Finally we slightly depart from the information flow in Simon's phase model. Instead of a sequential process combined with feedback loops between each phase, we view the three phases as a fully *concurrent and parallel* process. As soon as some information is perceived, the building of the alternatives tree starts and will be updated continuously as new information are coming. In parallel, dominance structures are built and as soon as one satisfying decision is found, the whole process is stopped to make the final decision. Hence this concurrency mechanism enables to take bounded capacities into account, by preventing the decision maker from exploring useless alternatives.

To sum up, our decision model incorporates the Simon's phase model augmented by five mechanisms:

- selective attention (salience) ;
- anchoring ;
- partial tree of alternatives ;
- search for dominance ;
- concurrency.

3. THE CODAGE MACRO-AGENT

In this section, we propose an agent architecture to implement our cognitive decision model. Broadly speaking, the CODAGE agent is a *macro-agent* managed by a cognitive multi-agent architecture. Let us first give some intuitions that support this idea. In section 2.2, we described our decision model with five mechanisms we incorporate into Simon's model. These mechanisms could be viewed as modules of decision subprocesses. As suggested by the concurrency mechanism, we do not believe in a sequential organization of theses modules. On the contrary, we claim that most of decision subprocesses are autonomous entities that interact at different phases of the decision process. Let us take the example of anchoring. In trading, round values (e.g. 50) are easier to be memorized and might be favored as decision thresholds. This number anchoring effect will not only bias the perception process, but also the alternative choice and probably the dominance search (round numbers will be favored to be the attributes of the dominance structure). Given our concurrency mechanism, we adopt the "Minsky's Society of Mind" point of view [15] and propose to model all these subprocess modules as autonomous agents in interactions.

The MAS comprises a set of specialized agents, we call *micro-agents* in order to distinguish them from the CODAGE macro-agent they belong to, and a tree of alternatives implemented as a blackboard system to facilitate information sharing, as depicted in Figure 3 below.

3.1 Knowledge representation

3.1.1 Facts

In CODAGE, we represent facts with a set of attributes, values, and predicates. For instance in a trading game, `$capital[capital_euros]=2501.2` means that the attribute "capital" has a value of 2501.2 and this numerical value is typed as "capital_euros";
`buy_proposition(alice,3,14.5)` encodes the fact that Alice proposed to buy 3 stocks at 14.5 euros each. We add two important mechanisms to encode the information processing prescribed by our cognitive model : salience and tree of alternatives.

3.1.2 Salience

The *salience*[2] of a fact represents its importance within the selective attention process. Each micro-agent ma of global agent pool $\mathcal{P}$ can vote to set the salience of a knowledge K within an alternative C (context, i.e. a possible state of the world). We denotes $v_{ma,K,C} \in [0,1]$ the resulting value of such a vote. If the value is strictly positive, K is added to C with the corresponding salience value $v_{ma,K,C}$ if K is new to C ; if K is already instantiated in C, then its value is simply updated in the equation 1 that gives the final value $S_{K,C}$ of the salience of a given knowledge K within the context of an alternative C as the mean of the micro-agents ' votes:

$$S_{K,C} = \frac{\sum_{ma \in \mathcal{P}} v_{ma,K,C}}{card(\mathcal{P})} \quad (1)$$

Neurobiology supposes that a salient fact is processed more quickly than an non-salient one [8]. In our model, knowledge-source agents will focus their attention on salient facts. This is implemented with two kinds of delays: an *event propagation delay* $d_{K,C}$, which causes agents to be warned later for non-salient facts, and a *reaction delay* $d_{R,C}$ for each rule R activable in a knowledge-source agent.

The propagation delay is 0 if the salience is 1, and rises to a maximum level (γ) if the salience is 0. We use the following function:

$$d_{K,C} = -\gamma . \left(\frac{S_{K,C} - 1}{S_{K,C} + 1} \right) \quad (2)$$

Figure 2 shows how this delay evolves in function with salience.

Let $A_{R,C}$ be the activation of a rule R in alternative context C. It equals the mean of premise 's saliences:

$$A_{R,C} = \frac{\sum_{\pi \in Premises(R)} s_{\pi,C}}{card(Premises(R))} \quad (3)$$

The agent reaction delay is inspired by the ACT-R theory [1, p.1043]:

$$d_{R,C} = I + Fe^{-A_{R,C}} \quad (4)$$

where $I = 597ms$ and $F = 890ms$.

Finally, the total information propagation delay of knowledge K in context C is given by:

$$td_{K,C} = d_{K,C} + d_{R,C} = -\gamma . \left(\frac{S_{K,C} - 1}{S_{K,C} + 1} \right) + I + Fe^{-A_{R,C}} \quad (5)$$

[2]Several psychological studies support the concept of salience. Due to lack of space, we suggest this review of salience effects [9]

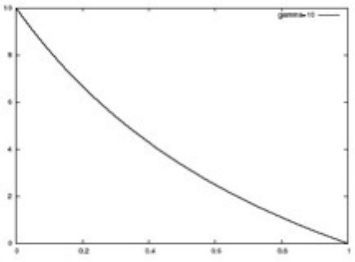


Figure 2: Propagation delay as a function of salience ($\gamma = 10$)

Our mechanism of propagation delay has two major benefits:

- the system is more robust to information permutation: even if a low-salient information is added prior to a high-salient one, the latter will be considered first.
- it enables the partial exploration of the tree of alternatives, since the micro-agents will act based upon the most salient facts. Alternatives based on low-salient facts will not lead to further consequences exploration.

3.1.3 Tree of Alternatives (TA)

In CODAGE, the knowledge that the decision-maker has about the world is encoded into a decision tree[3], as the one depicted in Figure 4.

Each node is an alternative that represents a possible state of the world (past, current or future). TA is a decision tree, as in decision theory, but it will be only partially built and explored to be consistent with bounded rationality. TA works at a symbolic level: each alternative represents an instantiation context in which each micro-agent may add a fact and/or an action into the tree: this is a way to share information between micro-agents. Each fact in the tree has a salience that measure its degree of importance.

Arcs between alternatives nodes represent *transitions* in time, that what produce the transition from one alternative (parent) to another one (child). We implemented two types of transitions that triggers the change to a new state of world:

- *action* transition: a possible action, performed by the macro-agent (myself)
- *fact* transition: the probability that some attribute will have a certain value (e.g. the final stock value will be 56.2 Euros at the closing of the market) or that an other agent perform some action (e.g. bob has sold 5 stocks to alice at 14.6 Euros).

TA could be viewed as a *blackboard* system. As one knows, the opportunistic control of knowledge sources (the micro-agents in our case), that is running the right agent on the right data at the right time, is a tricky issue in blackboards [5]. In our model, there is no fixed agenda to select one agent at a time: each agent is autonomous, and is able to modify data on the decision tree whenever it needs to. From a computer implementation point of view, it is a full multi-threading process. To preserve data coherence and integrity inside the tree, we implemented a mechanism to *solve eventual contradictions*. Any agent can signal a contradiction inside a given context C. In this case, the blackboard removes the two contradictory facts from C, creates two children of C (C_1 and C_2) and instantiates the two incompatible facts in two separated contexts. This method preserves the existence of the two solutions while avoiding the contradictions.

[3]We do not assume that a human decision-maker actually has such a decision tree inside his/her head. This is just a convenient modeling tool to tackle alternatives management in our model.

3.2 Agents

Each agent encodes a subprocess of the decision system, like an heuristic, an inference mechanism, perception, etc.

The *perception agent* (abbreviated as **PER** in the remaining part of this paper) imports informations from environment: e.g. buy and sell orders, accepted transactions and so. This knowledge is introduced at the root of the TA as symbols, predicates and variables. Initial salience values are set, depending on decision maker's habits and experience (what he/she is used to consider as important information)[4].

The *egocentric agent* (**EGO**) helps the macro-agent to selectively enhance the salience on every facts and actions he/she is involved in (e.g. the orders he gave, the proposals he made).

The *world rules agent* (**WRU**) contains the knowledge about the world rules. It encodes the main rules and constraints within the environment like the possible actions (e.g. in our simulated game, a trader can emit buy or sell order, or cancel a previous order), the forbidden actions (e.g. to buy with a null capital), and some anticipated consequences of actions (e.g. if an order is accepted, capital and bids count are updated according to a particular formula).

The *expertise agent* (**EXP**) contains a set of domain-specific heuristics and strategies the decision maker may use to perform his/her actions. In our example of a trading game, these strategies will increase the salience of critical attributes like total capital, gain and loss. They will give the relevant hypothesis to explore, like buying or selling a share. They also value the different facts (e.g. in term of expected outcomes).

The *anchoring agent* (**ANC**) gives the set of anchoring values, that will be used as reference points. In a predicate where some attribute value is unknown, the anchoring agent enumerates all possible values, and will propose to anchor to an already perceived value or to a given reference-point value, e.g. a value linked to the personal situation of the decision maker, or a constant specific to the problem domain (a national interest rate for instance).

The *uncertainty agent* (**UNC**) encodes the uncertainty of informations in the TA. It (i) sets probability p_K for a fact K to occur, and (ii) sets the probability $Pr(C)$) of alternative context C to occur in the real world.

The *decision agent* (**DEC**) monitors the decision tree and implements the search for dominance. When an alternative is added into the tree, it evaluates it. If this is a *satisficing solution*, the tree building process is stopped, and the action that created this branch is selected. If the alternative is too low (the aggregated utility of this alternative is lower than an elimination threshold), it is ignored. In other cases, the alternative is considered to be studied later, and added to

[4]In real-world applications, we could ask some experienced subjects to give their rankings importance for a set of domain facts, and derive the initial salience from this. However, when we will design a learning mechanism for the salience, the importance of these initial values will be much lowered (Cf. section 5.2 in the discussion)

an internal list. When this list is full, the alternative having the highest aggregated utility is selected. We compute the utility of an alternative A as follows :

$$AU(A) = f(\sum_{C \in Child(A)} Pr(C).u(C)) \quad (6)$$

where $Child(A)$ is a the set of immediate children of A in the tree. $Pr(C)$ the probability given by UNC agent (see above) and f is an utility normalization function, a numerical function valued in $[0, 1]$. For instance, we could adopt CARA (Constant Absolute Risk Aversion) function for risk-averse subjects $f(x) = -(1/\rho).e^{-\rho.x}$, where $\rho \in [0, 1]$ is a risking factor.

The utility $u(C)$ of an alternative C is given using a classical multi-attribute utility model, where we use the salience to weight each fact :

$$u(C) = \sum_{K \in C} p_K.S_{K,C}.v(K) \quad (7)$$

where K is a knowledge fact in C, p_K his probability, and $v(K)$ its associated value (e.g. expected outcome) as given by EXP agent. It is worth noticing this influence of salience (computed by the other agents) into the decision process : the most salient knowledge will have the highest weight in the utility function.

3.3 Decision process overview

We summarize the decision process in CODAGE with the flow charts depicted in Figure 3.

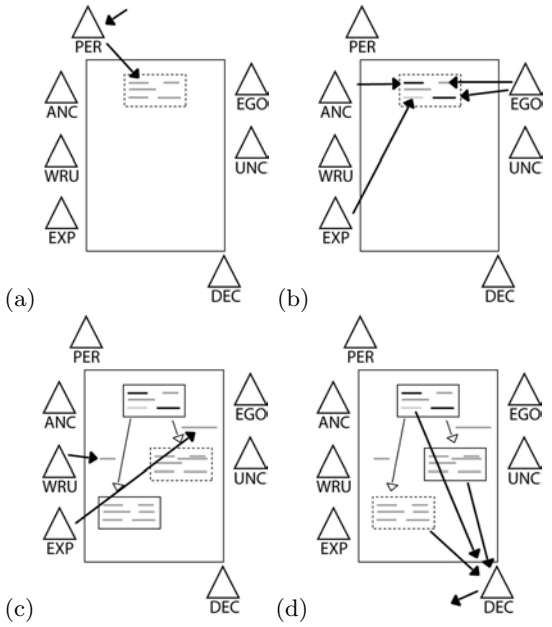


Figure 3: CODAGE decision process overview

Intelligence (a) The perception agent represents the current world in the root of the tree TA. (b) As soon as information appears, the EGO agent look for personal concerns and increases the corresponding saliences. Expertise agent may also update salience based on new information and its heuristics, while ANC agent increase the saliences of anchored values.

Design (c) Based on the most salient facts, agents use the TA to simulate actions, and to anticipate events and other decision maker's actions in a short or medium term. New alternatives are added to the TA, from EXP, WRU and DEC among others.

Decision (d) In parallel with (b) and (c), the decision agent assesses alternatives (utility computation), apply dominance search that leads either to the choice of an action or a selection of alternatives to be further explored.

4. SIMULATION RESULTS

4.1 Simulating an experimental market

This architecture has been instantiated in the economy field. We did not reuse the classical benchmarks used by the *Agent-based Computationnal Economics* community, like the well-known Santa-Fe Artificial Stock Market (SF-ASM), since we want to focus on cognitive aspects of decision-making within a simulated market, while SF-ASM focus on conditions of equilibrium and market behaviors using reactive agents.

We have selected an experimental financial market conducted by Biais, Hilton, Mazurier and Pouget [2]. This experimental market is aimed to study the effects of cognitive biases on the decision of traders placed on a market under asymmetric information. On this market, traders can publish at any time buy or sell orders (fixing the count and the limit price), accept an offer or cancel a previous order. There is a single risky asset, which pays a liquidating dividend at the end of the game which can be A, B or C with equal probability (in the experiment, 50, 240 and 490). Before trading starts the players receive heterogeneous private signals. For instance, if the final dividend is B, half the participants are privately informed it is not A, while the others know it is not C. There exists no communication between participants.

Hilton et al. suggest than the participants try to analyze the actions of the others to find the final asset price. Traders are reasoning in a high-uncertainty context, making them more influenced by **cognitive biases**. The authors study two biases : *overconfidence* and *self-monitoring*. Overconfidence makes the decision-maker to overestimate the representativity of his current informations. Traders suffering of self-monitoring are more attentive to the image they present to others, making them more manipulative.

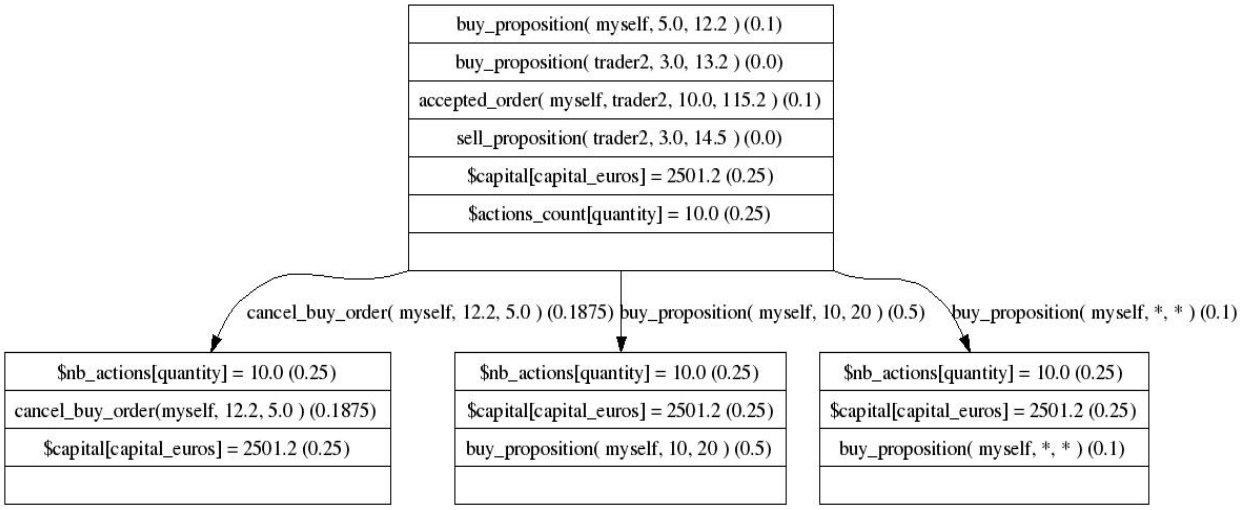


Figure 4: A Tree of Alternatives

We implement the **overconfidence bias** in CODAGE. To do so, we decrease the importance of initial probabilities (to favor current informations). Giving p_i the initial probability, $nb\_observations_K$ the number of times K is observed by the macro-agent, and $total_nb\_observations$ the total number of observations, the probability of a fact K is given by :

$$p_K = \frac{\beta.p_i + nb\_observations_K}{\beta + total_nb\_observations} \quad (8)$$

The modification of the β parameter of UNC agent modifies the sensitivity of personal experience. The self-monitoring bias seemed to be too general to be implemented yet.

The other experimental settings are as follows. We use the decision equations (7)-(8) described in section 3.2, with f set to a simple mean function, and $v(K)$ set to fixed randomly chosen values (no prior knowledge). Finally, we use here two instantiations of ANC agent : one for quantity values anchoring, and the second for prices anchoring. Each ANC agent only favor a set of discrete value (e.g. price value or quantity value), according to a salience anchoring curves like the one depicted in Figure 5. In this Figure, the quantity values are discretized using a step equals to 5, other values will a null salience value :

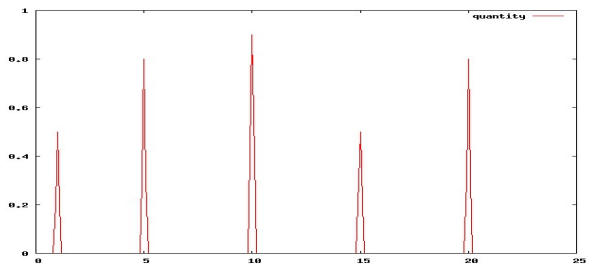


Figure 5: salience anchoring curve for quantity

4.2 Example of simulation

To see how CODAGE implements the experience described above, let us examine a tree generated by our program depicted in Figure 4 above. The process that generated this tree is the following:

- the *perception agent* added facts in the root alternative. At this time, no fact is salient, it is the raw perception.
- when a fact is added, an event "NewNonSalientFactEvent" is sent to all agents. Agents dealing with salience vote for facts : EGO agent votes for all facts concerning this trader, the ANC agents vote for salience according to their own salience curve, and the EXP agent highlights facts useful for trading (salience is displayed between brackets on the figure).
- each time a salience is modified, an event "NewSalientFactEvent" is sent with a latency, computed using Equation 5 . Each agent can react. Here, the WRU agent has proposed to cancel a previous offer or to emit a new buy order.
- when an agent proposes a new action, the TA copies salient facts (over an given recopy threshold) in the new alternative.
- WRU has added an incomplete predicate $buy\_proposition(myself, *, *)$, which contains two undefined variables : count and price. The ANC agents propose first the most salient values, here 10 unities and 20.0.
- at each alternative modification, the DEC agent evaluates - if possible - the alternative, and selects it if it is a satisficing one.

4.3 Overview of market simulation

An overview of the market simulation is displayed in Figure 6, which shows offers (plain) and demand (dotted) curves, and the trades (squares). During a primarily period, the agents put orders that are too riskless for being accepted (low offers, high demands). Then the EXP agent modifies salience of facts leading to a compromise (we supposed it was one of the trader's general heuristics). Traders will trade on this basis. Since traders use values generated by ANC agents, only anchored values will be used in the market.

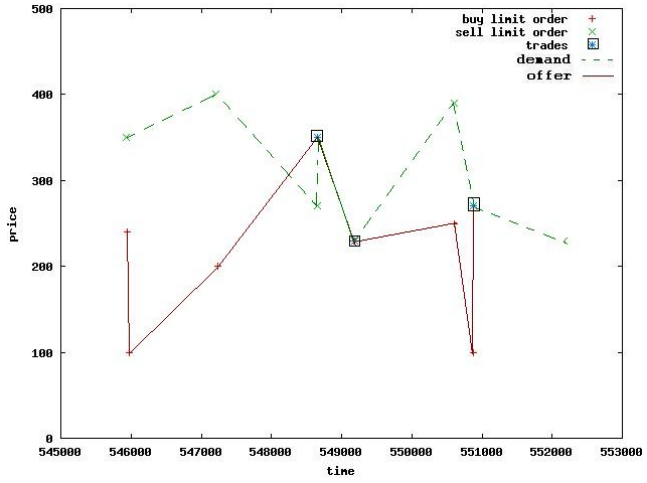


Figure 6: Biais et al. 's market simulation (extract)

5. DISCUSSION

In this paper, we propose the CODAGE approach to model human decision-making, where the decision-maker is modeled with a multi-agent system. Following psychomimetism, we designed a cognitive decision model based on psychological theories. We implemented this model with a specific multi-agent system architecture, made of a set of specialized cognitive agents that share information using a tree of alternatives TA. We gave some simulation results, and have shown how a cognitive bias could be implemented into the system. Now that we have fully described our approach, it is easier to compare to related works. Due to lack of space, we limit this comparison to the most used cognitive architecture in MAS : BDI.

5.1 CODAGE and BDI

Belief-Desire-Intention (BDI) agents have been very popular to implement cognitive processes, including decision making [18, 17]. Based on the philosophical concepts of intentions, plans and practical reason as developed by Bratman [3], it involves two important processes: deliberation where the agent commits to a particular plan, and means-end reasoning where the agent builds the possible options (plans). The agent *intends* to do what will achieve its *desires* (goals) given its *beliefs* about the world. An intention means an action the agent commits to. The question arises whether a BDI architecture could implement our cognitive decision model, and to what extent the CODAGE agent differs from the BDI framework. BDI and CODAGE do have things in common. First, they both try to account for bounded rationality: they incorporate partial exploration of states (alternatives in CODAGE, plans in BDI). Second, they both implement Simon's three stages decision model: in BDI, intelligence includes beliefs building and beliefs revision, design includes means-end reasoner to generate options and filtering, and choice includes deliberation.

However, they differ from several points. At first, it should be noted that BDI theory intends to capture rational behavior as a combination of deliberative planning and alternatives selection. It is strongly goal-oriented, and is aimed to produce plans, i.e. organizing a sequence of actions through time. In CODAGE, we adopt a broader - more abstract view of decision-making, where actions could be selective (e.g. buying a particular stock at a given price) or more complex (e.g. executing a plan). We focus on how possible actions (alternatives) are built, explored, biased and interact with other subprocess like perception and choice. We propose a descriptive theory on how a decision-maker may process information to do so. Second, CODAGE includes several mechanisms derived from psychological theories of decision-making (selective information, anchoring, search for dominance, miscalibration bias) which is not the case in original BDI architecture[5]. Last but not least, as we mentioned above, BDI is a sequential process. In CODAGE, we believe that most of the subprocesses involved in decision-making are independent and concurrent, they influence each-other at several phases within the Simon's model and therefore should be implemented as autonomous agents in interaction within a MAS. At this early stage of our work, we are aware that this strong concurrency assumption needs further validation (we discuss the validation issue in 5.2 below).

To sum up, BDI and CODAGE do not tackle the same issues, and are not exactly concerned with the same kind of cognition process: CODAGE focuses on alternatives and decision-making while BDI focuses on plans and reasoning. Therefore, when the emphasis is made on planning, nothing prevent from using a CODAGE agent to select a plan within a BDI architecture. Similarly, one could add a BDI agent within the MAS architecture in CODAGE in order to implement some practical reasoning and planning process within a complex decision making. Again, this is yet a theoretical view, and further work is to be done in that direction to study the mutual benefits of BDI and CODAGE.

5.2 Limitations and future work

At first, we need a more complete experimental validation of our model. In section 4, we presented some experimental results of the Biais et al 's experimental market that are encouraging as the global results look like the one observed in the experiment. CODAGE is a "white box" which produces a decision process to explain human data in a human-readable form. However, we lack many informations and expertise about this experiment, since we did not have access to it. As many symbolic systems aimed to model mental behavior, we need many informations (heuristics, anchors, initial probabilities, game rules...) to be inserted into the agents. We are now looking for an experimental market to set up, with an associated methodology to define how the necessary knowledge will be obtained. Concerning the validation process, we could use external experts (e.g. economists, traders) to assess to model and/or we could put real decision makers "into the loop", that is we have real decision makers that play against our artificial macro-agents.

Second, concerning the current model itself, one shall note that we introduced saliences only for facts and not for alternatives. But, when the tree of alternatives is enough developed, too many branches might be explored simultaneously. This could affects the stability of the system, and is not "cognitive-like" as it violates bounded rationality. There-

[5]Recently, Norling proposed to incorporate some elements of Folk Psychology into BDI, by adding an ability to learn to recognize situations and select appropriate plan based upon this [17]. This is done by adding a metal-level plan with a Q-learning reinforcement algorithm to select plans. However, the rest of BDI architecture remains unchanged

fore, we need to implement the salience notion at alternative level, allowing only one alternative to be salient at the same time. We need to determine the rules to manage this salience of an alternative, that is to define what makes an alternative salient.

Third, we need to incorporate *learning* into the system, so the CODAGE agent becomes more adaptive. In order to overcome the influence of prior knowledge, and to enhance the reliability of the decision system, we plan to introduce reinforcement learning, where the critical parameters will be affected by the results of actions. The first step would be to have salience values learned. To do so, we could use the reinforcement learning of the RALF neural network, a cognitive architecture designed to set the parameter values in order to get a positive reward as much a possible (RALF does not optimize reward, but find a satisficing solution) [12].

Finally, the CODAGE architecture is intended to be as much generic as possible. It could be viewed as an agent-based decision framework where different decision heuristics and biases could be implemented. We should move further into that direction, and try to incorporate new types of agent (e.g. emotional), new kinds of biases and heuristics.

Acknowledgments

We woud like to thank Nicolas Sabouret for helpful discussions and comments on early versions of our model.

6. REFERENCES

[1] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin. An integrated theory of the mind. *Psychological Review*, 111(4):1036–1060., 2004.

[2] B. Biais, D. Hilton, K. Mazurier, and S. Pouget. Judgmental overconfidence, self-monitoring and trading performance in an experimental financial market. *Review of Economic Studies*, 72(2), April 2005.

[3] M. Bratman. *Intention, Plans and Practical Reason*. Harvard University Press, Cambridge, MA, 1987.

[4] C. Castelfranchi. The theory of social functions. challenges for multi-agent-based social simlation and multi-agent learning. *Cognitive Systems Research*, 2(1):5–38, 2001.

[5] D. D. Corkill. Collaborating Software: Blackboard and Multi-Agent Systems & the Future. In *Proceedings of the International Lisp Conference*, New York, New York, October 2003.

[6] S. Dillon. Descriptive decision making: Comparing theory with practice. In 33rd *Annual Conference of the Operational Research Society of New Zealand*, Auckland, 1998.

[7] F. Guttman, A. G. Moukas, and P. Maes. Agent-mediated electronic commerce. *The Knowledge Engineering Review*, 13(2):147–159, 2002.

[8] N. H.-C. Salience from feature contrast : temporal properties of saliency mechanisms. *Vision Research*, 40(18):2421–2435, 2000.

[9] C. Haynes and S. Kachelmeier. The effects of accounting contexts on accounting decisions: A synthesis of cognitive and economic perspectives in accounting experimentation. *Journal of Accounting Literature*, 1998.

[10] K. Kahneman, P. Slovic, and A. Tversky, editors. *Judgement under uncertainty: Heuristics and Biases*. Cambridge University Press, New-York, 1982.

[11] J.-D. Kant. Modeling human cognition with artificial systems : some methodological considerations. In *Proceedings of HCP 99 (Human Centered Processes)*, pages 501–508, Brest, September 1999.

[12] J.-D. Kant and D. Levine. Ralf : A simplified neural network model of rule formation in the prefrontal cortex. In *Proceedings of JCIS98,*, volume II, pages 8–14, Research Triangle Park, USA, october 1998.

[13] D. Koehler and N. Harvey, editors. *Blackwell Handbook of Judgment and Decision Making*. Blackwell, 2004.

[14] B. LeBaron. Agent-based computational finance: suggested readings and early research. *Journal of Economic Dynamics & Control*, 24:679–702, 2002.

[15] M. Minsky. *The society of mind*. Simon & Schuster, Inc., New York, NY, USA, 1986.

[16] H. Montgomery. Decision rules and the search for a dominance structure: toward a process model of decision-making. In P. Humphrey, O. Svenson, and A. Vari, editors, *Analyzing and Aiding Decision Process*. North-Holland, Amsterdam, 1983.

[17] E. Norling. Folk Psychology for Human Modeling: Extending the BDI Paradigm. In *AAMAS 04*, New York, July 2004.

[18] S. Parsons and N. R. Jennings. Argumentation and multi-agent decision making. In *Proceedings of the AAAI Spring Symposium on Mixed-Initiative Decision Theoretic Systems*, Stanford, 1998.

[19] S. Parsons and M. Wooldridge. Game theory and decision theory. In S. Parsons, P. Gymtrasiewicz, and M. Wooldridge, editors, *Game Theory and Decision Theory in Agent-Based Systems*. Springer Verlag, Berlin, Heidelberg, New-York, 2002.

[20] H. Simon. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69:99–118, 1955.

[21] H. Simon. Rational choice and the structure of the environment. In H. Simon, editor, *Models of Man*. John Wiley, New-York, 1957.

[22] H. Simon. *The New Science of Management Decision*. Harper and Brothers, New York, 1960.

[23] A. Tversky and D. Kahneman. The framing of decisions and the psychology of choice. *Science*, 211:453–458, 1981.

Efficient Agent-Based Models for Non-Genomic Evolution

Nachi Gupta
Oxford Univ., Computing Lab
Numerical Analysis Group
Wolfson Building, Parks Road
Oxford, OX1 3QD, UK
nachi@comlab.ox.ac.uk

Adrian Agogino
UC Santa Cruz
NASA Ames Research Center
Mailstop 269-3
Moffett Field, CA 94035, USA
adrian@email.arc.nasa.gov

Kagan Tumer
NASA Ames Research Center
Mailstop 269-4
Moffett Field, CA 94035, USA
ktumer@mail.arc.nasa.gov

ABSTRACT

Modeling dynamical systems composed of aggregations of primitive proteins is critical to the field of astrobiological science, which studies early evolutionary structures dealing with the origins of life. Current theories on the emergence of early life focus either on RNA-world models [2, 3] or protein world models [4, 5]. Traditional modeling based on either model are generally either too slow to converge or too simplified to provide good tools for exploring the trade-offs in the early stages of the emergence of life. This paper focuses on protein-world models and discusses how to model protein aggregations through a utility based multi-agent system. We define agents to control specific properties of a given set of proteins. These properties determine the dynamics of the system, such as the ability for proteins to join or split apart, while additional properties determine the aggregation's fitness as a viable primitive cell. We show that over a wide range of starting conditions, there are mechanisms that allow protein aggregations to achieve high values of overall fitness. In addition through the use of agent-specific utilities that remain aligned with the overall global utility, we are able to reach these conclusions with 50 times fewer learning steps.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Performance

Keywords

Multiagent Systems, Non-genomic Evolution, Reinforcement Learning

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

In this paper we show how multi-agent systems can make an important contribution to the field of astrobiology - the study of primitive ecosystems that occurred early in Earth's history and may be present of other planets. We focus on the subfield of "non-genomic evolution," involving the evolution of protein ecologies that do not contain nucleic acids (DNA and RNA) which form the basis of modern evolution. Protein ecologies are important since simple proteins are easier to make through natural processes than nucleic acids and it is thought that life on Earth might have started with these ecologies. The significant feature of protein ecologies is that unlike nucleic acids, proteins cannot directly replicate themselves. As a result, non-genomic evolution is concerned with the global behavior of the entire protein ecology and its ability as a whole to form new protein ecologies. It is therefore a good domain to use multi-agent systems to model the interactions between proteins and the emergence of specific protein ecologies.

The study of non-genomic evolution seeks to find the initial conditions and mechanisms that must be present to form self-sustaining protein ecologies [6]. Unfortunately with current knowledge it is impossible to directly simulate the evolution of a protein ecology. Direct evolution of low-level chemical processes has limited promise, because of the computational difficulties in creating a simulation that would encompass a full planetary ecology. In principle, feasible simulations can be obtained by modeling the higher level dynamics of protein interactions that map a protein's chemical structure (amino-acid sequences) to probability distributions over possible functions. However, because at present few of these mappings are known, a meaningful simulation at this level is not possible. As a consequence non-genomic evolution is currently analyzed using "plausible" models for the dynamics of the protein ecology [4, 5]. These simplified dynamical models are then run and are evaluated based on a global utility function. This global utility abstractly measures the value of the resulting protein ecology. Unfortunately creating a model that leads to a good final ecology is laborious and extremely time comsuming.

This paper shows how learning agents can be used to improve this modeling process by having agents automatically learn parameters of protein ecology models that lead to high global fitness, allowing more complex and meaningful models than the ones currently used in the astrobiological community. In this approach agents choose properties of individual proteins that affect the dynamics of the system (such as a protein's ability to split other proteins) as well as

protein properties that are important in producing a viable ecology, such as the ability of an ecology to produce lipids needed for a cell membrane. In a continuous process, agents choose protein properties while the distribution and number of proteins is being updated through the dynamics of the ecology. Using reinforcement learning, the agents attempt to make choices that lead to high global fitness, determined by a given utility function evaluating the ecology's viability. In our experiments the mapping between the proteins' properties and the ecology's global utility is more complex, indirect and realistic than in the models previously used in the astrobiological community. As a consequence ecologies that are able to achieve high fitness provide a better example of mechanisms that can lead to self-sustaining protein ecologies.

Section 2 describes the underlying principles and previous work associated with non-genomic evolution. Section 3 mathematically describes the model used in this paper simulating the dynamics and evaluating ecologies of proteins. Section 4 shows how learning agents are used to optimize the properties of the proteins so that the entire system achieves high global fitness. Section 5.1 presents results from simulations of the system showing that ecologies with high global fitness can be found quickly. Section 6 then discusses the significance of these results to the field of astrobioligy, and provides future directions in which multi-agent systems can be used to benefit this study.

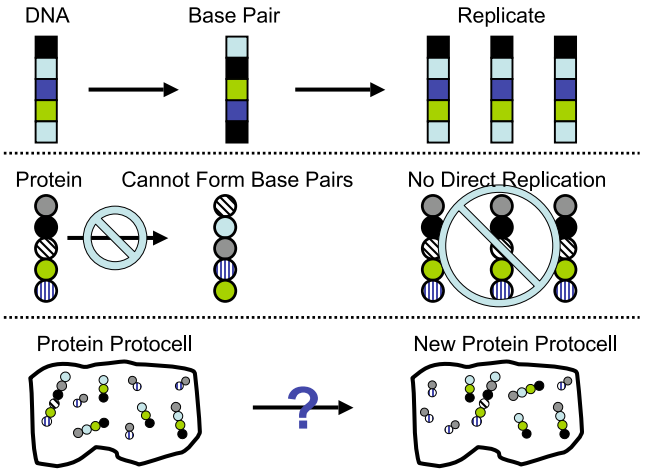


Figure 1: Replication of Proteins. DNA and RNA can form base pairs and directly replicate their sequences. Proteins cannot form base pairs and therefore cannot directly replicate themselves. However as a system, a group of proteins forming a protocell may be able to replicate by creating other protocells with approximately the same characteristics.

2. BACKGROUND AND PREVIOUS WORK

All current life is based on organisms replicating through information coded in their DNA (Deoxyribonucleic acid) or RNA (Ribonucleic acid). DNA and RNA are composed of sequences of nucleic acids, which can form base-pairs allowing replication of the sequence. Due to their ability to replicate, models of the origins of life based on DNA or RNA are the most popular models [3]. However, DNA and/or RNA based models have many difficulties [6]. Current organisms require a complex interaction between DNA and RNA to survive. Even hypothetical organisms based only on RNA ("RNA worlds") require complicated interactions between different types of RNA [2, 3]. In addition, the nucleic acid sequences which form DNA and RNA are difficult to make under the pre-biotic conditions that existed early in Earth's history.

Therefore, there is a growing interest in non-genomic evolution - the study of organisms that do not use nucleic acids. Instead of using nucleic acids, these organisms use only proteins. The non-genomic evolutionary model of primitive organisms has the advantage that the proteins are made of sequences of amino-acids, which have been shown to be made under conditions similar to those that existed in Earth's early history. However, since amino-acids cannot form base-pairs, unlike nucleic acids, proteins cannot be directly replicated (Figure 1). Therefore the field of non-genomic evolution focuses on global properties of protein ecologies to see if protocells (precursors to cells which have some properties of cells such as having a membrane, but do not have all cell functions or cell components) formed from these ecologies can produce new protocells [2, 6]. Understanding the combined behavior of the ecology is paramount in order to ascertain the potential place of nongenetic evolution in the history of life on Earth.

While recent laboratory and computation progress has been made in understanding self-replicating protein ecologies, we are still far from being able to precisely model the evolution of a protocell based on its constituent proteins [4, 5, 7, 8, 11]. As a result current models are greatly simplified, using high-level assumption about the dynamics of the system [6]. One of the few models for non-genomic evolution is described in New and Pohorille [6] and is based on the relationship between the length of the protein and its efficiency - the percentage of time it succeeds in performing operations on other proteins. In this model, proteins perform two operations: hydrolysis and ligation. Hydrolysis is the process of decomposing a protein, whereas ligation is the process of building a new protein by joining two proteins. In this model the efficiency of ligation and hydrolysis are based on Gaussians, with longer proteins generally being more efficient. The fitness evaluation of the protocell in this model is based on protein length with longer proteins being evaluated higher than proteins with shorter length. The computational experiments performed with this model show that protocells become more fit with time. However, in this model the fitness evaluation is only concerned with the efficiency of two functions while numerous other properties are required for a viable protocell, such as a protein's ability to produce cell-membrane building lipids.

In this paper we will expand on these results and develop a more complex model of non-genomic evolution with the help of a multi-agent system. This setup differs from previous ones in that efficiencies that the proteins attain at certain tasks is determined by the actions of learning agents, allowing many more degrees of freedom in the dynamics of the evolution. In addition the proteins have more functions besides their ability to break and build other proteins. Finally the global utility used in evaluating the fitness of the protocell takes into account the distribution of all the protein functions in the protocell.

3. THE PROTOCELL MODEL

The model of the primitive protocell used in this paper is based on different protein types performing different tasks within the protocell and contributing differently to the fitness of the protocell. In this model a protocell has a set of proteins of various types. The two most important types of proteins are combiner and breaker proteins. Combiner proteins take two other proteins and combine them into a single protein (ligation). Breaker proteins take a single other protein and break them into two separate proteins (hydrolysis). Other protein types have important functions such as lipid production for cell walls, energy production and energy transport. However in our model, these other proteins do not actively contribute to the dynamics of the system and are only important in evaluating the final viability of the protocell as defined by the system's global utility.

3.1 Global Utility

For a protocell to survive and thrive, it needs different types of proteins performing different tasks. Certain types of proteins are needed for building cell membranes, while others are needed for tasks such as energy production, energy transportation and chemical exchange. If a cell has a poor distribution of proteins, such as having no proteins for building cell membranes, it will be unlikely to survive and should receive a poor utility. Our model makes an abstraction over the different types of possible proteins, ignoring their low-level functions. We only assume that there are optimal numbers of each type of protein and we use an exponential decay to assign utility to off-optimal distributions.

To formally define the global utility of a protocell, let there be k possible types of proteins and let y_i be the desired amount of proteins of type i in the system, while x_i represents the current amount of proteins of type i present in the system. We choose the functional form $x_i e^{-\frac{x_i}{y_i}}$, which takes its optimal value of 1 when $x_i = y_i$, to describe how close we are to the desired number of proteins performing function k. The intuition behind this choice is that there is an optimal amount of a particular protein needed in the protocell. If there is less than that amount, the protocell would not function properly, and if there is more than that amount, the protein uses more resources than needed to perform its task.

The global utility for the protocell is simply the sum of these functions:

$$G(x, y) = \sum_{i=1}^{k} x_i e^{-\frac{x_i}{y_i}} , \quad (1)$$

where x is the vector of the current amounts of each protein type and y is the vector of desired amounts of each protein type.

3.2 Protocell Dynamics

The dynamics of the protocell are determined by hydrolysis and ligation. During ligation, a "combiner protein" joins two proteins into a single protein. During hydrolysis, on the other hand, a protein is broken into two proteins. There are two crucial issues in both operations. First, the combiner or breaker protein has to choose the proteins on which it will act. Second the protein(s) resulting from the operation need(s) to inherit certain properties of their predecessors. These issues will be modeled through concepts of *specialization* and *clustering* respectively.

Based on current theory of non-genomic evolution, it is unlikely that combiner proteins in early protocells were specialized, i.e. they would combine two other proteins together regardless of what type they were [6]. However, it is likely that breaker proteins were somewhat specialized: There were some types of proteins which a particular breaker protein would not break. While the actual specialization dynamics are complex, we will model this specialization by assigning each breaker protein a single type of protein which it will never break. All other proteins will be broken by that protein.

When new proteins are created as a result of combining or breaking operations, they will be assigned types based on the proteins they came from. This assignment will be modeled through the principle of "protein type clusters." In this model we assume that proteins of the same type tend to congregate near each other, forming clusters. In our model each protein belongs to exactly one cluster, though multiple clusters may have proteins of similar types (Figure 2). When a new protein is created as a result of a breaking operation it will tend to go to a cluster of the same type as its parent (though not necessarily the same cluster). When a new protein is created based on combing two proteins, it will go to a cluster chosen based on the properties of the parent clusters. Details of these assignments are given in Section 4.

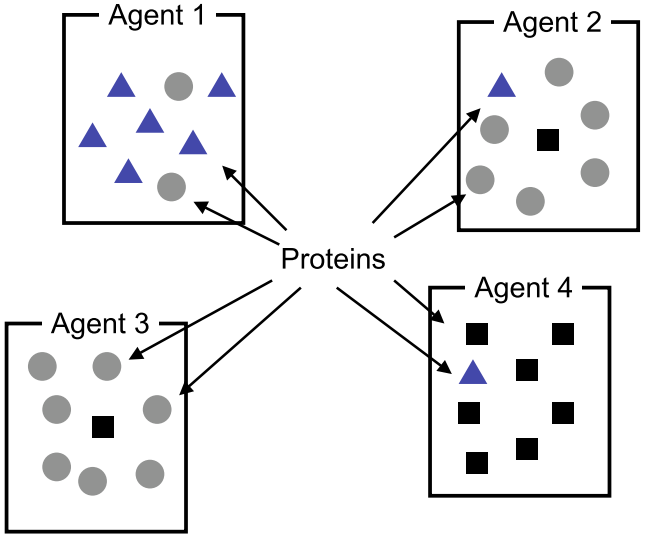


Figure 2: Type Agents. Each type agent determines the protein types in its cluster. Each agent makes decisions about the type of protein in its cluster independently, so proteins in the same cluster can have different types.

4. AGENT FRAMEWORK FOR PROTEIN MODEL

In this paper, agents play an important role in determining the specialization of the breaker proteins and determining the properties of the clusters. Specifically, each cluster will be assigned a "type" agent, whose actions determine the type of each protein in its cluster. Also each specialization

of a breaker protein will be determined by the actions of a "breaker-specialization" agents. This section describes the details of the learning process and the actions taken by the agents.

4.1 Type Agents

The types of the proteins in a cluster are determined by an agent assigned to that cluster. In our model an agent takes a discrete action ranging from 1 to k, where k is the number of possible types. Every time an agent takes an action, its action determines the type of a single protein in its cluster chosen at random. Over time, an agent takes many actions, defining the types of more and more proteins within the cluster. The goal of the agent is to choose types that ultimately lead to a protocell of high global utility. To accomplish this task, our agents learn through a simple reinforcement learning.

The system is modeled using a series of discrete time steps, where each agent takes an action after each time step. At the beginning of a time step each type agent takes an action determining the type of single protein in its cluster. This action is determined by an ϵ-greedy reinforcement learner with a learning table of size k, the number of protein types. Each entry in the table represents the agents expected utility when choosing the action associated with that entry. With probability $1-\epsilon$ an agent takes an action associated with the highest table entry. With probability ϵ it takes a random action. Note that with the ϵ-greedy learner, we would expect a cluster to have proteins of several different types. Most of the time the agent would set proteins to the type associated with highest expected utility, though sometimes it will choose random types with frequency depending on ϵ.

If an agent sets the protein type to something other than a combiner protein or a breaker protein, then nothing happens in the system until the next time step. However, if an agent chooses a protein to be a combiner or a breaker protein, the corresponding combining or breaking operations then take place. If the protein is a combiner, then two other proteins are chosen at random and combined together to form a new protein. If the protein is a breaker protein, then another protein is chosen at random for possible breaking. Whether this protein is actually broken depends on the specialization of the breaker protein, discussed next.

4.2 Breaker-Specialization Agents

Each protocell has several breaker-specialization agents associated with the proteins. A breaker-specialization agent will typically be associated with several proteins, but a protein will only be associated with one breaker-specialization agent. The actions of its breaker-specialization agent determine the specialization of any breaker proteins that are associated with it. At each time step, the breaker-specialization agent takes one of k actions chosen with an ϵ-greedy reinforcement learner. This action then defines the specialization of a breaker protein chosen randomly. This specialization determines the type of protein the breaker protein cannot break: If the protein it operates on during its breaking operation is of the same type as its specialization, then the operation does nothing.

4.3 Cluster Assignment of New Proteins

After a combining or breaking operation, the cluster that the new protein belongs to is determined by the clusters of the original proteins involved in the operation. In our model we want the child protein(s) that result from these operations to end up in cluster(s) that are "similar" to the cluster(s) the parent protein(s) came from. While there are many ways to define the similarity between clusters, our model will use the learning tables for the agents assigned to each cluster to define similarity (Figure 3). For a breaking operation, a protein p in cluster c is broken into two new proteins. To find the cluster c_{new} for one of the new proteins p_{new}, we first compute a vector, v as follows:

$$v = Q_c + e \ , \tag{2}$$

where Q_c is the learning table of the type agent associated with cluster c, and e is Gaussian noise. The cluster for the new protein, c_{new} is then chosen as the cluster associated with the agent that has the closest learning table to v, based on the Manhattan Norm (1-norm):

$$c_{new} = argmin_c \delta(Q_c, v) \ , \tag{3}$$

where $\delta(\cdot)$ is the Manhattan Norm.

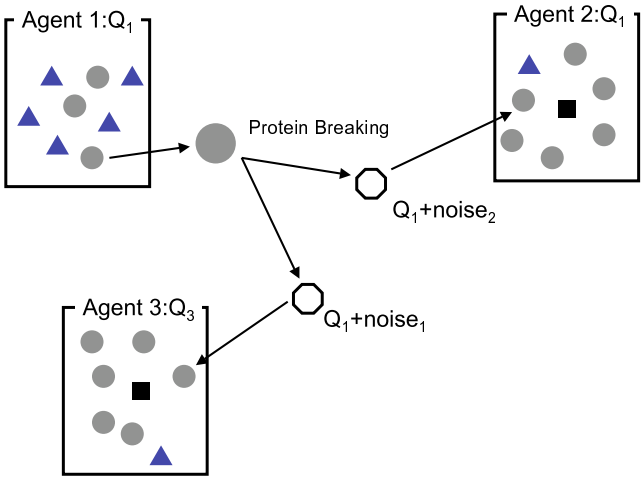


Figure 3: Breaking Proteins Apart. When a protein is broken apart, two child proteins are created. The types of each of the child proteins depend on the learning table of the agent for the cluster the original protein came from. For each child protein a new table is made by adding noise to this learning table. Each child protein is assigned to the the cluster whose agent has the learning table is closest to the proteins table.

For a combining operation a protein p_1 in cluster c_1 is combined with protein p_2 in cluster c_2 to form a single new protein, p_{new}. As with the breaking operation, to find the cluster c_{new} for the new protein we first compute a vector, v. However this time v is a function of two learning tables since p_{new} is created from two proteins. To handle this we combine the learning tables using the element-wise max operator $m(x, y)$, which assigns to each component of m_i the maximum of x_i or y_i: $m_i = \max(x_i, y_i)$. The vector v can then be computed as:

$$v = m(Q_{c_1}, Q_{c_2}) + e \ , \tag{4}$$

where Q_{c_1} and Q_{c_2} are the learning tables of the type agents associated with clusters c_1 and c_2, and e is Gaussian noise.

As before the cluster for the new protein, c_{new} is then chosen as the cluster associated with the agent that has the closest learning table to v, based on the Manhattan Norm (1-norm):

$$c_{new} = argmin_c \delta(Q_c, v) \ , \quad (5)$$

where $\delta(\cdot)$ is the Manhattan Norm. The process for determining the breaker-specialization agent associated with a new protein is parallel, with the learning tables of the breaker-specialization agents being used.

4.4 Agent Utilities

The goal of the agents is to maximize the global utility $G(x, y)$ as defined in Equation 1. However when there are many agents, it is difficult to maximize this utility. This difficulty arises from the fact that a single agent will only have a small impact on the value of G. If an agent takes an action and G increases, that agent does not know if the increases were due to its action or the actions of the other agents. When agents do not have significant influence over their utility as compared to the influence of all the other agents, it usually takes a very long time for them to learn to take actions that maximize it.

To alleviate this problem, this paper has each agent maximize a "difference utility" instead of the global utility. A difference utility is specific to an agents i, and can be computed as follows:

$$D_i(z) = G(z) - G(z_{-i}) \ , \quad (6)$$

where z is a vector of all the actions taken by all the agents and z_{-i} is a vector of all the actions except for the action taken by agent i. The second term of the difference equation is a counterfactual version of the global utility that can be seen as the value of the system without agent i. Therefore the difference utility can be seen as the agent's net contribution to the system. By subtracting this counterfactual second term from first term, the difference equation removes much of the noise caused by the actions of the other agents. The difference utility has been shown to greatly speed up reinforcement learning in many domains, including internet traffic routing, multi-robotic control and satellite communication [9, 10, 1].

The specific form of the difference utility for our global utility, $G(x, y)$, defined in this paper is:

$$D_i(x, y) = G(x, y) - G(x_{-i}, y) \ , \quad (7)$$

where x_{-i} represents the counts of the protein types for a protocell, had agent i had not taken any action. The exact form of x_{-i} depends on the type of action made, especially if the agent made a protein a combiner or a breaker. For all actions let j be the type of protein chosen by the agent. For combiner actions let a and b be the types of proteins that were combined and let c be the type of protein produced. For breaker actions let a be the type of protein broken and let b and c be the types of proteins produced. Then the value of x_{-i} is as follows:

1. $x - e_j$ for non-combiner or breaker proteins ,
2. $x - e_j + e_a + e_b - e_c$ for combiner proteins ,
3. $x - e_j + e_a - e_b - e_c$ for breaker proteins ,

where e_j is a vector where element j has a value of 1 and the other elements are 0. Essentially for combiner and breaker proteins, the proteins created are removed from the counts and the proteins consumed are returned.

5. RESULTS

To test the ability of our agent-based system to produce effective models, we ran a number of experiments where agents had to create models that led to protocells with high utility starting with different protein distributions. In this task as agents had to create a model with the proper distribution of combiner and breaker proteins, and had to ensure that the breaker proteins had the proper selectivity profile. In addition agents had to ensure that the functions of all the proteins were in the proper distribution.

In all the experiments there were thirty agents. In each experiment there were five different types of proteins that were desired in the final outcome besides combiner and breaker proteins (e.g., proteins for energy collection, energy transport, membrane building, and nucleus building). The experiment tested different starting conditions, different desirable combinations of these proteins by selecting an optimal number of each these five proteins, y_1, y_2, y_3, y_4, y_5, along with a corresponding optimal total $y = y_1 + y_2 + y_3 + y_4 + y_5$, and changing optimal conditions.

- The first experiment tested the ability of the agents to create a model that reached a good final equilibrium, starting from y proteins. The intent of this experiment was to determine whether the desired combination of proteins can be reached by rearranging the proteins present at the start.
- The second experiment tested this ability starting from 2 proteins. The intent of this experiment was to determine how few proteins are needed for the process to start, and whether a protocell can develop starting from just two proteins and generate the necessary proteins to perform all the required tasks.
- The third experiment tested this ability starting with $3y$ proteins. The intent of this experiment was to test whether the protocell can cut through clutter and eliminate unnecessary proteins.
- The fourth experiment tested the ability of the protocell to adapt to a sudden change in environment where the optimal number of each type of protein is changed halfway through the simulation. The intent of this experiment was to see the robustness of the protocell to sudden changes in its environment.

5.1 Protocell Evolution from y Proteins

In the first experiment the protocell started with 30 proteins. The optimal number of proteins was 31 (1+2+4+8+16), with $y_1 = 1, y_2 = 2, y_3 = 4, y_4 = 8$, and $y_5 = 16$. Since the starting and optimal number of proteins was almost the same, this problem was relatively simple and the only task of the agents was to create a model that forms the proper distribution and maintains it. However, despite the simplicity of the task, Figure 4 shows that agents directly maximizing the global utility could not produce an adequate model. In fact the initial actions of the agents produced a model that was worse than the starting one, and after 5000 learning trials they still did not manage to reach their initial starting performance. This result is not surprising since many choice

of model parameters can be very destructive, such as creating too many breaker proteins. Since it was very difficult for the agents to learn from the global utility, they had difficulty avoiding these destructive choices. In contrast, agents using the difference utility were able to produce adequate models for this task. By changing the distribution they were able to create a model that achieved a utility of 3.7, up from the initial utility of 2.8. The agents using the difference utility were more effective than the agents using the global utility because the difference utility was much more learnable. The significance of this model to non-genomic evolution, however is minimal. It shows that the proper distribution can be maintained, intuitively meaning that there should be few combiner and breaker proteins.

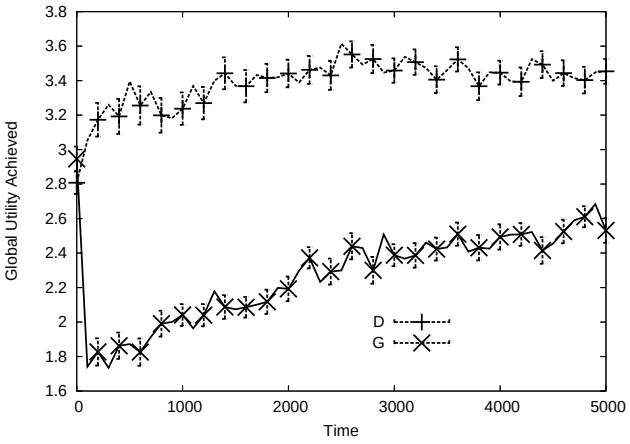


Figure 4: Performance for protocell starting with near optimal number of proteins. Agents using difference utility form adequate model by changing protein distributions to fit targets. Global utility cannot reach original (random) performance.

5.2 Protocell Evolution from 2 Proteins

In the second experiment, the protocell started with 2 proteins. The optimal number of proteins was 31, with $y_1 = 1, y_2 = 2, y_3 = 4, y_4 = 8$, and $y_5 = 16$. This presented a much more difficult setting than the previous one, since starting with 2 proteins, the agents had to create a model that increases the number of proteins by a factor of 15, while providing that the proteins are in the proper distribution. This requires an initial increase in the distribution of breaker proteins to create enough total proteins and a shift in the distribution, as well, to maintain the proper protein levels. Figure 5 shows that agents using the difference utility are able to create a model that leads to highly fit protocells. In this experiment, the performance of the protocell goes from 0.8 to 3.5, a significantly larger increase than in the previous experiment. This larger increase is expected since the initial protocell with two proteins is in a much worse state than the one starting with 30. This experiment shows that a model exists where a protocell with a small number of proteins can grow to one with a large number of proteins and form the proper protein distribution. This has significant insight into the required initial conditions for the formation of protocells. This result demonstrates that with the proper conditions a protein ecology can be formed starting from only 2 proteins.

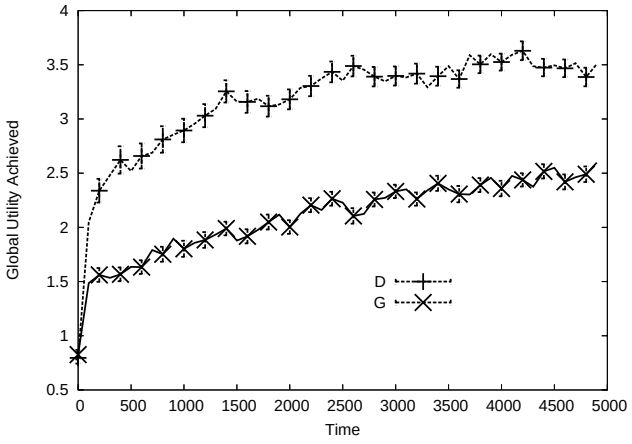


Figure 5: Performance for protocell starting with 2 proteins. Agents using difference utility are able to form model where combiner and breaker proteins are formed, and lead protocell to an equilibrium where there are enough proteins, and the proteins are in the correct distribution.

5.3 Protocell Evolution from 3y Proteins

In the third experiment the protocell started with 95 proteins. The optimal number of proteins was again 31, with $y_1 = 1, y_2 = 2, y_3 = 4, y_4 = 8$, and $y_5 = 16$. This experiment requires an initial increase in the distribution of combiner proteins so that proteins in the initial high population are combined together to form a smaller population. As before results show that agents using the difference utility are able to create a model that leads to highly fit protocells. In this experiment the performance of the protocell quickly shoots from 2.2 to 3.6, but then goes down slightly. This slight reduction in utility is due to the model producing too many combiner proteins that causes the system to overshoot its protein reduction. Figure 6 shows that when there are too many proteins in a protocell, a mechanism can be created to reduce the number and create a proper protein distribution.

5.4 Protocell Evolution in Dynamic Environments

In the final experiment, the protocell started with 35 proteins with an optimal number of proteins of 31. The optimal number of proteins was thirty one, with $y_1 = 1, y_2 = 2, y_3 = 4, y_4 = 8, y_5 = 16$, but at 5000 episodes is changed to $y_1 = 16, y_2 = 8, y_3 = 4, y_4 = 2$, and $y_5 = 1$. This experiment requires the agents to adapt to new model requirements after they have already developed an existing model. The results shown in Figure 7 show that the agents are able to quickly adapt to this change in model requirements.

6. DISCUSSION AND CONCLUSION

In this paper we explore multi-agent models to simulate potential ways in which early non-genomic cells can evolve to perform the tasks required for survival. Through the use of learning agents, we create an abstract dynamical model

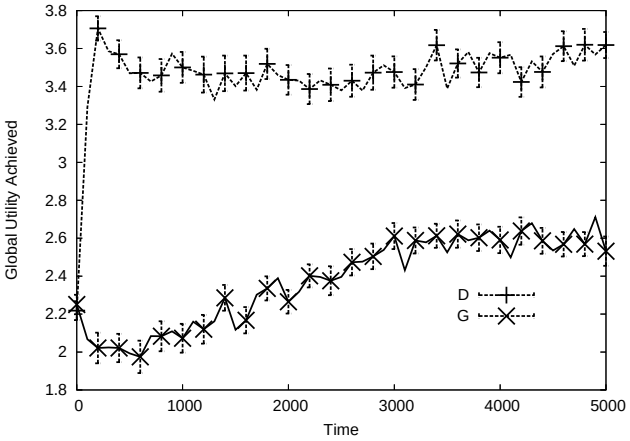


Figure 6: Performance for protocell starting with three times too many proteins. Agents using difference utility are able to form model where combiner and breaker proteins are formed that lead protocell to an equilibrium where there are enough proteins, and the proteins are in the correct distribution.

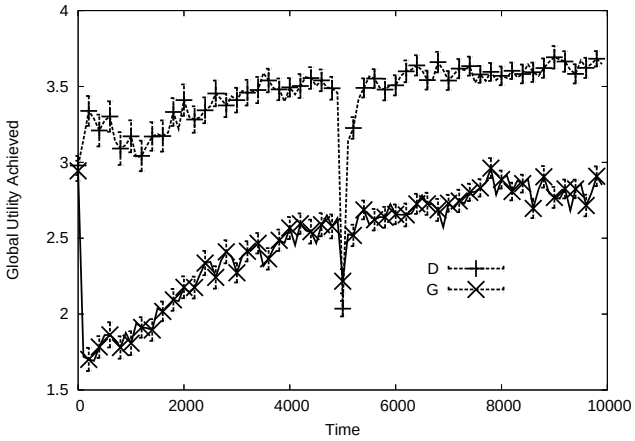


Figure 7: Model Formation for Changing Distribution. When requirements for the distribution of proteins changes at time step five thousand, agents are able to form a model that adjust to change.

for a primitive protocell of considerably greater complexity than previous models. In particular our models include proteins that had many different properties and had more complex dynamics for protein combining and breaking. In addition, while the fitness utility used in other models directly reflects the dynamics of the system, such as efficiency, our fitness utility is only indirectly influenced by the dynamics. Therefore our fitness utility is more realistic and provides many more degrees of freedom for the system to evolve. With our framework, we show that models exists that can produce highly fit protocells from a wide variety of starting conditions.

While our use of multi-agent systems allow us to create models of non-genomic evolution that are more complex than previous models, they are still highly abstract. One issue with the model is that only combiner and breaker proteins affect the dynamics of the system. We are currently working with astrobiologists to help add other dynamics into the framework. Another issue is that the optimal protein distributions used in our global utility of the protocell are somewhat arbitrary. However our framework allows other distributions to be plugged in, and we are also working with astrobiologists to come up with more realistic values.

Even though many details need to be added to our models to increase their biological plausibility, the multi-agent framework provides the power and flexibility to do so and naturally matches the distributed properties of protein-only ecologies. In general more complex evaluation of protocells can simply be implemented through the agent utilities. In addition more complexity can be introduced by adding different types of agents or changing the agents action space. Due to their flexibility we believe that multi-agent systems will become an important tool the study of astrobiology.

Acknowledgments: The authors thank Dr. Andrew Pohorille, Astrobiology branch, NASA Ames Research Center, for invaluable discussions and for patiently describing the intricacies of the "protein world" models of early life.

7. REFERENCES

[1] A. Agogino and K. Tumer. Efficient evaluation functions for multi-rover systems. In *The Genetic and Evolutionary Computation Conference*, pages 1–12, Seatle, WA, June 2004.

[2] A.J. Hager, J.D. Pollard, and J.W. Szostak. Ribozymes: Aiming at RNA replication and peptide synthesis. *Chemistry and Biology*, 3:717–725, 1996.

[3] G.F. Joyce. Ribozymes - building the RNA world. *Current Biology*, 6:965–967, 1996.

[4] D.H. Lee, J.R. Granja, J.A. Martinez, K. Severin, and M.R. Ghadiri. A self-replicating peptide. *Nature*, 382:525–528, 1996.

[5] D.H. Lee, K. Severin Y. Yokobayashi, and M.R. Ghadiri. Emergence of symbiosis in peptide self-replication through a hypercyclic network. *Nature*, 390:591–594, 1997.

[6] M. H. New and Andrew Pohorille. An inherited efficiencies model of non-genomic evolution. *Simulation Theory and Practice*, 8:199–208, 2000.

[7] R.W. Roberts and J.W. Szostak. RNA-peptide fusions for the in vitro selection of peptides and proteins. In *Proceedings of the National Academy of Science USA*, pages 12297–12302, 1997.

[8] K. Severin, D.H. Lee, J.A. Martinez, M. Vieth, and M.R. Ghadiri. Dynamic error correction in autocatalytic peptide networks. *Angewandte Chemie International Edition*, 37:126–128, 1998.

[9] K. Tumer and D. Wolpert, editors. *Collectives and the Design of Complex Systems*. Springer, New York, 2004.

[10] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.

[11] S. Yao, I. Ghosh, R. Zutshi, and J. Chmielewski. Selective amplification by auto and cross-catalysis in a replicating peptide system. *Naturre*, 396:447–450, 1998.

Cognitive Agents Based Simulation for Decisions Regarding Human Team Composition

Bruno Gonçalves
We, the Body and the Mind Research Lab, ADETTI/ISCTE
Av. das Forças Armadas 1649-026 Lisboa, Portugal
+351 21 782 64 80
Bruno.goncalves@we-b-mind.org

Sofia Esteves
We, the Body and the Mind Research Lab, ADETTI/ISCTE
Av. das Forças Armadas 1649-026 Lisboa, Portugal
+351 21 782 64 80
Sofia.esteves@we-b-mind.org

António Lopes
We, the Body and the Mind Research Lab, ADETTI/ISCTE
Av. das Forças Armadas 1649-026 Lisboa, Portugal
+351 21 782 64 80
Antonio.lopes@we-b-mind.org

Luís Botelho
We, the Body and the Mind Research Lab, ADETTI/ISCTE
Av. das Forças Armadas 1649-026 Lisboa, Portugal
+351 21 782 64 80
Luis.botelho@we-b-mind.org

ABSTRACT
This paper proposes to use cognitive multi-agent based simulation to facilitate human team formation decision processes. The models of organizational actors are acquired by a data-gathering monitoring system that stores all the interactions that take place in the selected cooperative task. In this knowledge acquisition process, only a relatively small number of team compositions are monitored. The extracted models are used in the agent-based simulation of all the possible team compositions. Simulated team compositions are evaluated and only a small set of the most promising ones is submitted to the human decision maker. An emergence model overrides the multi-agent based simulation in localized situations for which the involved set of agents should not be trusted. The model of the simulation agents is a motivated case-based reasoning system, in which the regular functioning of the CBR is biased to fulfil the agent's motives. The motivation model includes individual motivation and social influence.

General Terms
Algorithms, Experimentation, Theory.

Keywords
Multi-agent simulation, agent motivational model, brainstorms.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

1. INTRODUCTION
This paper addresses the KnowledgeAndCulture.org project. The main objective of the project is to use multi-agent simulation of teams of organizational human actors to assist the decision process involved in selecting the best human team for a specific cooperative task. The chosen cooperative task to be studied is the brainstorm, which frames a broad set of specific cooperative processes in a large range of problems and domains.

In some resourceful advanced organizations team formation involves highly specialized psychologists that manually create cognitive maps of all the available people [9]. This approach is a long and very expensive process. In order to improve this extremely resource consuming decision process we propose to use multi-agent based simulation.

The project has reached the first year of its duration therefore no simulation results have yet been produced. However, several other intermediate concrete results have been achieved. This paper presents some of those achieved results and contributions.

1.1 General Approach
Our approach (see Figure 1) is divided in two main stages: organizational knowledge acquisition stage; and simulation and evaluation stage.

In the first stage, a reduced set of human teams is observed while performing the cooperative task (brainstorms, in our case study). The interactions among the human team members are extracted through a computational system and stored into a repository of interaction cases. Only a few team compositions are monitored.

The stored interactions combined with a motivation model used to determine the agents courses of action based on their motives, allows us to create an artificial cognitive agent for each available organizational actor.

In the second stage, different teams of agents are simulated and evaluated according to defined evaluation criteria. Since there are agent teams that have not been observed, we've decided to use models of emergent social behaviour as a way of generating plausible emergent interactions among the agents that represent people that have not been observed working together.

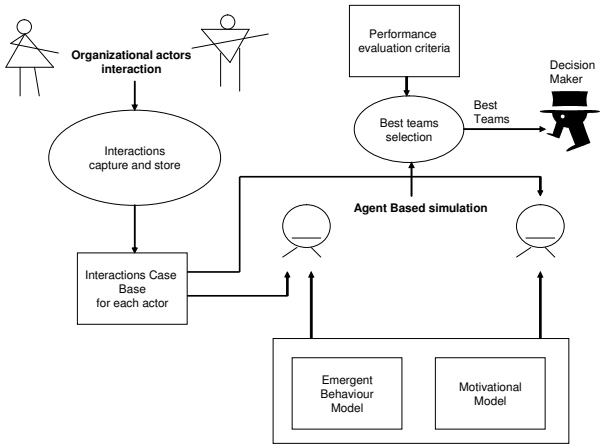


Figure 1. Cooperative work simulation model

The result of the simulation process is the set of a few best teams, which are then considered by the human decision makers. This approach will help focus the human decision process on more promising choices.

1.2 Contributions

The main contributions of the project stem from its decision to use multi-agent based simulation for real concrete organizational problems instead of relying upon toy problems. The requirement to use multi-agent simulation of real organizational problems places constraints on the design of the simulation process and on the computational and conceptual tools to be used. The satisfaction of such constraints continuously produces results for similar future simulation processes.

The first and maybe the most innovative contribution of the project is the designed simulation process. The proposed design has not produced simulation results yet. However the new proposed ideas contribute to and can be essayed in other simulation processes. The proposed simulation design makes two contributions regarding the multi-agent based simulation mainstream: the use of knowledge extracted from actual simulated organizational actors monitored during real interactions in the simulated task, and the integration of cognitive simulation agents with a model of emergent behaviour. In traditional multi-agent based simulation, the agents of the simulation are endowed with pre-defined mental models that are assumed to capture each simulated human actor. In the proposed simulation process design, such kind of pre-defined models are integrated with knowledge extracted from actual organizational actors. Besides, when the behaviour of the simulation agents is not believed to accurately mimic the simulated actors, the model of emergent behaviour takes control over the agents.

The second contribution of the project is the rigorous formalization of brainstorms, which is the cooperative task considered in the project. The formalism models all possible brainstorm interactions between participants.

The third contribution is the definition and implementation of a computational system that supports brainstorm interactions based on the mentioned brainstorm formalization [8], avoiding the complexity of natural language processing. This system monitors the interactions and stores them in a case base for each actor.

The fourth contribution is the definition and partial implementation of a motivated case based reasoning system (CBR). The motivated CBR system integrates a model of motivated behaviour defined in the project with traditional case based reasoning techniques. The flat CBR system (CBR without motivation) has been defined and implemented. Several contributions regarding the use of CBR in large real-life problems have been made. The model of motivated behaviour has been defined and implemented.

The fifth main contribution is the design of two sets of evaluation criteria, domain dependent and domain independent, to be used during the simulation of the cooperative task for ranking the teams according to their performance in the task.

In section 2, we describe the brainstorm formalization. In section 3, we present the model of the simulation agents. Section 4 presents the evaluation criteria used to rank simulated teams. Section 5 presents the chosen simulation scenario. Section 6 provides an overview of related work. Finally, section 7, presents our work's conclusions and plans for future developments.

2. BRAINSTORM FORMAL MODEL

Brainstorms are cooperative tasks in which a set of participants proposes a problem to be solved for which they try to find a satisfactory solution. There are two aspects to be considered regarding brainstorms: the process consisting of proposing a problem and the components of its solution, and the decision process used by the set of participants to accept the proposals being made.

The decision process depends on the structure of the particular group of participants. There are autocratic decision processes, democratic decision processes and hybrid decision processes. The steps of the decision process are not considered in this paper. Final decisions regarding the several alternatives being proposed and discussed during the brainstorm are modelled as if they were the responsibility of a particular virtual actor hereafter referred to as the *Brainstorm Decision Maker*. This option does not commit to a particular decision process. The decisions are made according to the group decision structure but they are represented as if the virtual *Brainstorm Decision Maker* made them.

This section describes the brainstorm processes involved in the proposal of the problem to be solved, the proposal and discussion of the several components of the problem solution. This process will be called hereafter as the *Brainstorm Problem Solving Process*. The formalization of the *Brainstorm Problem Solving Process* encompasses two issues: the formalization of the general problem solving process, and the representation of the individual proposals made by the participants.

2.1 General problem solving process

Traditionally, Artificial Intelligence and Operations Research view problem-solving processes as search problems in the problem space or in the solution space. These search problems are formally represented as graphs in which nodes represent states, and hedges represent actions causing state transformations.

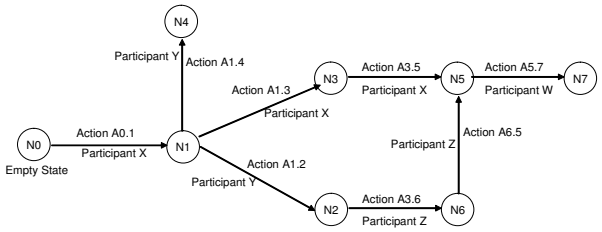


Figure 2. Brainstorm search graph

We propose to represent the general problem solving process used in brainstorms as an acyclic directed graph. In general, nodes represent partial solutions to the proposed problem; and arcs represent actions performed by brainstorm participants that transform the partial solution to which they were applied into a new partial solution.

In Figure 2, the arc from node *N1* to node *N2* is the action *A1.2* performed by participant *Y* in the state contained in node *N1*, which originates the new state contained in node *N2*.

Brainstorm actions always add new elements to brainstorm partial solutions therefore it is impossible to have closed loops in a brainstorm graph. Although it is possible to remove elements from a given partial solution, the removed element is not erased; instead, it is marked as removed. This way, it is possible to capture all the proposals being put forth by all participants during the brainstorm. The result of this policy is that the brainstorm problem solving process is represented by an acyclic graph. Brainstorm graphs are branching structures because it is possible that two or more different actions are performed by possibly different participants in the same partial solution. Different actions performed in the same partial solution give rise to different alternative approaches to solve the same problem.

Brainstorm graphs are not trees because, as shown in Figure 2, it is possible that two alternative courses of action meet in the same brainstorm state (node *N5*, in this case).

By convention, the initial node in a brainstorm graph is the empty state. Usually, the first step in a brainstorm is the definition of the problem to be solved. Problems are introduced in the brainstorm by the addition of a goal to the brainstorm. Among other possibilities, goals may be satisfied by plans of action, by systems design, and by strategy / policy definition, depending on the goal's nature. These elements that directly or indirectly contribute to achieve a brainstorm goal (actions, propositions, objects) are called solution elements. Besides goals and solution elements, brainstorm states may include documents, arguments, different kinds of compositions of those elements, and individual and collective approvals and rejections of proposed elements. As previously explained, collective approvals or rejections are achieved through group decision processes whose model will not be addressed in this paper.

There are brainstorm actions to add and remove (mark as removed) goals, to decompose a goal in sub goals, to add and remove solution elements of any kind (e.g., relationships, classes and objects), to relate existing brainstorm elements (e.g., "this element contributes to this goal", or "from those two elements, it is possible to derive this new element"), to add and remove documentation about any brainstorm element, to add and remove arguments against or in favour of any brainstorm element, to individually and collectively accept or reject any solution element.

2.2 Brainstorm state description

Nodes in the brainstorm graph represent brainstorm states. The formalism used to represent brainstorm states is a propositional version of a SNePS like language [13], which is a specific semantic network representation language. Currently, we use the SNePS graphical notation for being seen by people and an equivalent textual notation to be processed by agents. We are considering using only the graphical notation in future versions of the system if an automated reasoning mechanism can be created that takes advantage of the properties of the graphical notation to improve efficiency.

The brainstorm state representations are used in agent-based simulations. The simulation agents have to mimic the behaviour of the actual organizational human actors therefore they don't need to understand the detailed contents of brainstorm states' propositions. They only need to learn when and how to use the same propositions. Hence propositions are always represented as propositional symbols. However using only propositional symbols is not enough for automated processing therefore the formalism uses classes (i.e., attribute-value descriptions) to better describe propositions.

Since human participants must understand the detailed contents of propositions, the brainstorm state representation language allows adding free text annotations to propositional symbols. Such annotations are often natural language and / or predicate logic based versions of the propositional symbols. Besides being necessary to improve human readability, textual annotations may be used in future applications of the brainstorm representation for instance for the automated generation of meeting minutes or for more demanding automated reasoning systems.

The graphical notation of the brainstorm state representation is a graph in which nodes represent atomic and compound concepts, and arcs represent structural relations between nodes. There are two kinds of nodes: atomic and molecular nodes. Molecular nodes are nodes with arcs to other nodes and represent compound concepts. Atomic nodes do not have arcs to other nodes.

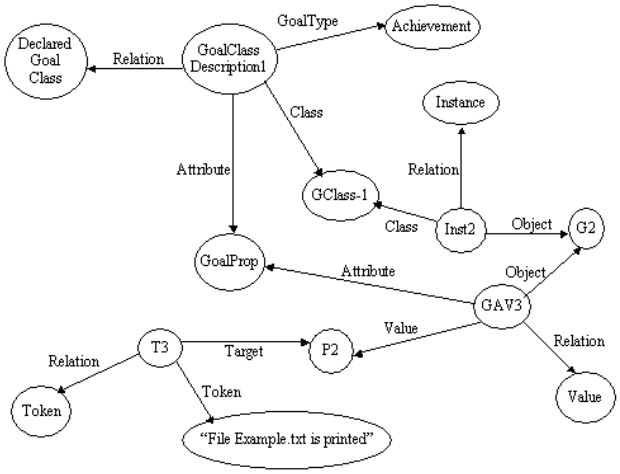


Figure 3. Brainstorm state: Goal definition

In the brainstorm state shown in Figure 3, the molecular node *GoalClassDescription1* describes the goal class *GClass-1* of type *Achievement* with an attribute named *GoalProp* (goal proposition); the molecular node *Inst2* represents the fact that the goal *G2* is an instance of *GClass-1*; molecular node *GAV3* represents the fact that the value of the attribute *GoalProp* of goal *G2* is the proposition *P2*; and finally, *T3* attaches the annotation *"File Example.txt is printed"* to proposition *P2*. All other nodes in Figure 3 are atomic nodes representing atomic concepts. In summary, the brainstorm state of Figure 3 represents the problem to be solved, namely to achieve a state of the world in which *P2* is true (i.e., the file *Example.txt* is printed).

In the brainstorm state of Figure 4, molecular node *Contr23* states that the action of printing file *Example.txt* in printer *LPT1* (molecular node *Act5*) contributes to achieve goal *G2*, which as explained earlier (see Figure 3) is the goal of having the file printed. This statement can now be accepted or rejected by individual participants or collectively by group, according to its decision structure. Acceptance or rejection of this statement would be represented by a propositional molecular node taking another proposition as an argument (*Contr23*). That is, the acceptance / rejection of this proposal would be a proposition taking another proposition as an argument. Since first order logic cannot represent such kinds of knowledge, we have decided to use the SNePS representation language.

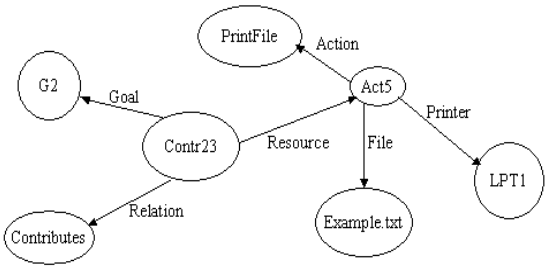


Figure 4. Brainstorm state: solution element

The graph of Figure 4 is textually represented by the following expression:

```
Contr23 # Contributes(resource : Act5 #
PrintFile(printer : LPT1, file : Example.txt),
goal : G2)
```

In the textual representation, the nodes representing relationships (*Contr23*) and actions (*Act5*) are attached to the expressions representing those relationships and action.

2.3 Brainstorm assistance tool (BAT)

The presented formal model allowed us to define and implement a computational system supporting distributed brainstorms with a user interface (BAT – Brainstorm Assistance Tool) [8]. Using the BAT tool, organizational actors participating in the distributed brainstorm only have to choose menu options. Those options are general brainstorm actions like: "propose a discussion problem", "argue", "propose a (partial) solution for a chosen problem", remove previously proposed solutions, and approve and reject previously proposed solutions.

The computational system supporting distributed brainstorms has a distributed peer to peer architecture through which each peer can be used by a brainstorm participant to interact with the others.

This system, besides supporting distributed brainstorms, records the interactions of each participant in a case base. The generated case bases are used in the simulation stage, in which artificial agents representing the real organizational actors interact with each other using the BAT system.

For brainstorms related with particular domains it is possible to create plug-ins capable of converting domain specific actions into domain independent actions used by the computational system (e.g. cell phone selection plug-in).

3. BRAINSTORM SIMULATION AGENT MODEL

The agent model (see Figure 5) comprises two sub models: domain specific knowledge used to control the agent's actions; and the motivational model used to select the agent current motives, to compute their strengths, and to influence the operation of the domain specific action control knowledge. Despite being developed for the brainstorm specific application, the agent model is absolutely independent of the domain.

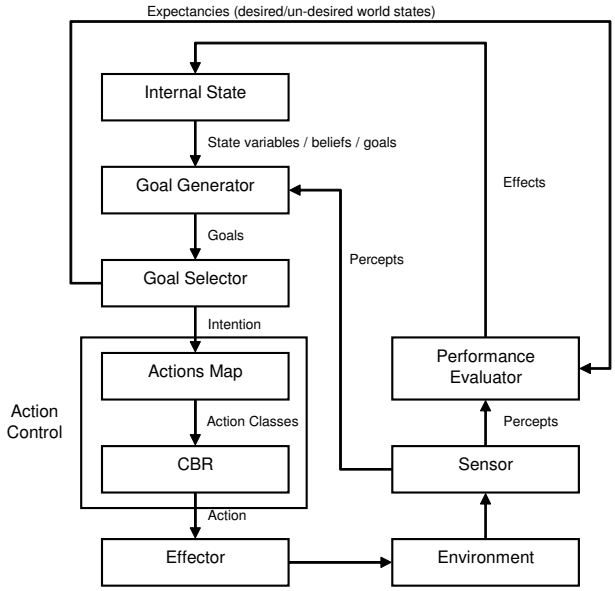


Figure 5. Brainstorm agent model

The domain action control knowledge of the agents of the simulation consists of the case bases extracted and stored during the organizational knowledge acquisition stage. A case based reasoner is used to solve the new problems faced by the agent during the simulated brainstorm. The CBR is influenced by the agent motivation, as determined by the agent motivational model.

According to the proposed motivational model, the agent motivation depends on the agent mental state (e.g., beliefs) and percepts, as suggested by Sloman [14], and also on a set of nuclear internal state variables (e.g., fatigue) [1][3].

We have adopted a BDI (Belief-Desire-Intention) [2][4] approach for representing the agent mental states, mainly for compatibility with a strong agent research community.

The agent acts to maintain the values of its state variables within satisfactory limits. The distance between the current value and the optimal value of a state variable influences the level of motivation to act (e.g. the motivation to sleep depends positively on the fatigue level). We assume that, in brainstorms, the agent's motives (i.e. states that led the agent to act) have two main natures: an individual nature, and a social nature. For this specific activity (brainstorms), we have identified the nuclear state variables in Table 1:

Table 1. Brainstorm agent state variables

Name	Type	Description
Intellectual performance	Individual	Perceived level of mental task performance (interventions quality)
Activity level	individual	Level of brainstorm activity (interventions quantity)
Fatigue	individual	Level of psychological fatigue
Empathy with agent X	social	Level of empathy with agent X
Trust in agent X	social	Level of trust on agent X interventions

The agent model implements three main processes: goal generation and selection, action control, and performance evaluation. The process repeats itself indefinitely while the agent is alive. First, one or more goals are generated with corresponding strengths. In the second step, the agent selects the strongest goal that it believes to be achievable. The selected goal and its strength become the agent current intention. The action control mechanism is guided by the generated intention, leading the agent to act. The performance evaluation process assesses the extent to which the agent's motives have been achieved by the enacted behaviour, and updates the agent mental state and state variables accordingly. These processes are described in detail in the next sub-sections.

3.1 Goal generation and selection

The goal generation and selection process described in this section is mainly inspired by the ideas put forth by Sloman [15].

At the goal generation stage, internal states (state variables, beliefs and goals) and external states (percepts) are evaluated, and goals are generated if certain conditions hold.

Goals consist of a proposition representing a desired state of affairs, a numeric value representing its importance, a numeric value representing its urgency, and a numeric value called strength, representing a heuristic combination of the urgency and importance of the goal.

```
goal(accepted_proposal(proposal),0.4,0.6,0.5)
```

The goal strength is calculated by the following formula:

$$strength = importance \times icf + urgency \times (1 - icf),$$

with $0 \leq strength \leq 1$; $0 \leq importance \leq 1$; $0 \leq urgency \leq 1$; $0 \leq icf \leq 1$.

The *icf* value represents the importance contribution factor for strength.

Goal generation is implemented through a rule-based system (see Figure 6) whose rules are supplied by the agent designer. The rules structure comprises: the generation conditions (beliefs, goals and percepts); a list of alternative mental state and percept conditions (msc/2) used to influence the goal importance; a list of fuzzy rules used to compute the goal importance by evaluating the state variable values; a list of alternative mental state and percept conditions (msc/2) used to influence the goal urgency; a list of fuzzy rules used to influence the goal urgency by evaluating the state variable values; and the goal information structure, which comprises: the goal proposition, the goal failure conditions, the list of goal achievement effects, and the list of goal failure effects.

```
goal_generation_rule(
    (
        percept(brainstorm_problem(Problem)),
        belief(competent_towards(Problem))
    ),
    [   msc(bel(importance(Problem,high)),0.5),
        msc(bel(importance(Problem,medium)),0.25)
    ],
    [   fuzzy_rule_1,
        fuzzy_rule_2
    ],
    [   msc(bel(brainstorm_available_time(0)),0.5)
    ],
    [   fuzzy_rule_3
    ],
    goal(
        accepted_proposal(Proposal, Problem),
        (
            rejected_proposal(Proposal, Problem)
        ),
        [  increase(intellectual_performance)
        ],
        [  decrease(intellectual_performance),
           remove(belief(competent_towards(Problem
           )))
        ]
    )
)
```

Figure 6 – Goal generation rule example

Goal importance and urgency are determined by a set of mental state conditions, and a set of fuzzy rules whose conditions refer to the agent state variables, for example:

- If activity_level is low and intellectual_performance is high then importance is high;
- If fatigue is high then urgency is high;

Only goals whose strength would exceed a certain threshold (i.e. generation threshold) are generated. This prevents the agent from having irrelevant goals.

The agent continuously evaluates its goals in order to generate intentions. At each moment the agent intention is its strongest goal. The importance and urgency of the generated intention is the same as those of the corresponding goal. Intentions are states of affairs that the agent is committed to achieve.

```
intention(accepted_proposal(proposal),0.4,0.6,0.5)
```

The importance and urgency of goals and intentions are not static. They may change as the result of new evaluations of the goal generation rules. Goals may be removed when their strength decreases beneath the generation threshold. Intentions may be removed and their associated action control processes abandoned when their strength decreases beneath the generation threshold. When an intention is replaced by other, the action control process serving that intention is suspended, possibly being resumed afterwards if the originating goal becomes again the strongest.

The described interruption mechanism is inspired on the ideas put forth by Simon [14].

In order to achieve intended states, the agent uses its action control component, described in the next section.

3.2 Action Control

The first step of this process, implemented by the Action Map component (see Figure 5), is to identify the action classes that are likely to contribute to the achievement of the current intention.

The second step is the selection of the action to be performed. This action selection process is provided by the motivated CBR. Traditional CBR systems [5][11] select the case whose problem is the most similar to the current state of affairs. In the motivated CBR model, there are two criteria for case selection: the similarity of the stored case with the current state of affairs, and the extent to which the action class of the action prescribed by the case (i.e., the case solution) may contribute to the achievement of the agent intention, as determined by the first step. Often the solution prescribed in the most similar case is not adequate to the current problem therefore it must be adapted to fit the current problem.

Intention importance and urgency are used to determine the amount of effort to be employed for seeking the case to be used for the current problem, for the comparison between the current problem and each case that is retrieved from the case base, and for the adaptation to the current problem of the solution prescribed by the selected case. In the presence of an important intention, it is reasonable to expect that the agent performs an exhaustive search process, performs a systematic comparison process, and spends the best of its efforts to adapt the prescribed solution to the current problem. However if the urgency of the intention demands a quick response, it is reasonable that the agent assigns less effort to systematic and thorough processes.

These processes may influence state variables. For instance, exhaustive search processes can increase fatigue; and the discovery of a suitable case can increase the perceived level of intellectual performance.

The intention strength influences the way some actions are expressed in the agent world [1]. For example if the agent has the intention "injured(Creature)" and the CBR chooses the action "attach(Creature)", it is expectable that the intention strength influences positively the motor strength used in the attach as well as the number of strikes. The influence of intention strength on the intensity of a given behaviour depends on the type of action to be performed therefore the agent designer must supply the strength influence model for each type of action.

For this specific application (brainstorms) we have chosen the CBR approach because it is suitable to generate new behaviours from the record of past cases. Besides, it is easy to capture specific interactions between organizational actors, in specific brainstorm instances as cases in which the case problem corresponds to the brainstorm state and the case solution corresponds to the action performed by the organizational actor in the specified state. For other types of applications, the agent designer has the possibility of extending the motivational model with the most suitable action control component (e. g. a planning algorithm or a rule based system).

3.3 Performance evaluation

The performance evaluator (see Figure 6) checks the extent to which the current/observed state of affairs satisfies the agent current goals. When a goal proposition is observed in the agent world, the agent performance is positively evaluated and the achievement effects specified in the goal generation rule are computed. When the goal failure conditions are observed in the agent world the performance is negatively evaluated and the failure effects are computed. The effects can be to increase or decrease the value of state variables, and the creation and deletion of beliefs. The magnitude of the effects in the state variables depends on the evaluated goal strength, and on the effort already spent by the agent trying to achieve the goal. The effort is calculated dividing the total time used by the action control processes serving this goal by the total time that the goal persisted. The effort is maximum when all the available time was used trying to achieve the goal, and minimum when no time was used.

3.4 Domain dependent components

Sensors and effectors are domain dependent components supporting the agent connection with its environment. The sensor role is to obtain the world sensorial information (percepts) and to deliver that information to the model components that require it. The effector converts the action descriptions (which include the originating intention strength) in effective actions in the agent environment.

3.5 Process description

In the presented model the goal generation and selection, the action control, and the performance evaluation processes run asynchronously. This option ensures that the agent is capable of identifying sudden internal and environmental changes and readjusting its course of action accordingly, at any time. Namely, it is capable of interrupting or suspending its current motives and attending new high priority motives.

4. SIMULATION AND EVALUATION

During the simulation stage, different teams of agents simulate the cooperative behaviour of the represented organizational actors with the purpose of selecting the best teams. Each team performance is then evaluated according to two sets of criteria: quality of the solution generated as the result of the brainstorm (domain dependent), and quality of the brainstorm problem solving process (domain independent). This section addresses these two steps.

4.1 Multi-agent simulation

The simulation process is carried out by a multi-agent platform populated with cognitive agents, each one representing a different organizational actor. The simulation agents interact within a brainstorm process according to the underlying models of the organizational actors they represent. The main idea is for them to fully interact with each other in the same way as the organizational actors would.

This simulation system is built on top of the agent platform JADE [18], which was built based on the FIPA specifications [19]. The simulation agents were built using component-based software developed within our research lab: XSP [20].

The simulation agents incorporate a brainstorm engine component built for the BAT system. This engine comprises the interaction capabilities needed for each agent to communicate with the remaining agents in the brainstorm.

The remaining capabilities of the simulation agents were developed as separated components that were assembled together with the brainstorm engine. These include the motivational and the domain action control (CBR) components. The emergence behaviour component will be developed as an external sub-system that, under certain circumstances, can influence all agents' activity.

4.2 Simulated Team Evaluation

The evaluation of each agent team brainstorm problem solving process involves the following domain independent criteria:

- Brainstorms taking less time are better;
- Shorter brainstorms (with less states) are better;
- Brainstorms revealing more cooperation among participants are better;
- Brainstorms revealing low levels of indecision and divergence are better.

The first two criteria are simply evaluated and therefore do not need any further explanation.

It is clearly undesirable that a single person proposes most of the solution elements in a cooperative context. Since state transitions are labelled with the identifications of the participants, it is possible to assess if actual cooperation is taking place.

The degree of indecision can be determined considering the number of times that a given solution element is introduced and subsequently removed. And the degree of divergence by evaluating the number of ramifications and reunifications in the brainstorm graph.

5. SIMULATION SCENARIO

We have defined and implemented a demonstration scenario in which to evaluate our proposals. The brainstorm task for the demonstration scenario was that of designing a cell phone specification (set of characteristics) for a specified customer profile. This task is an instance of brainstorm problems in which the solution is incrementally and cooperatively generated through a constellation of components. The same kind of decision process can be applied for instance in house design.

With this purpose, we have identified a set of critical cell phone characteristics, a set of customer's critical characteristics (defining customer's profiles), and also the type of arguments that are likely to be used in the mentioned decision process. The cell phone critical characteristics are those that are more likely to support the customer choice, for example: talk time and video capability. The customer profile is the set of observable user characteristics that can give clues about the customer expectations regarding a cell phone (e.g., the user occupation and the user age). The arguments are used to support or contest solution elements (mobile phone characteristics), based on the customer profile characteristics. For example, a brainstorm participant may argue that, for a customer with eighteen years old, it is necessary that the cell phone has a photo camera. In this case, the argument type is "necessary".

We have implemented a plug-in for the Brainstorm Assistance Tool (BAT) that provides a domain dependent graphical user interface for the described cell phone design task.

Besides the domain independent evaluation criteria (see Section 4.2), the agent team evaluation process is based on the user feedback regarding the cell phone design decision achieved through the brainstorm.

6. RELATED WORK

Agent based simulation is being used in a large variety of problems of different scientific areas, namely it has been used in modelling social processes with the purpose of structuring and exploring theoretical hypothesis. In the mainstream multi-agent based simulation approach, the knowledge used by the agents of the simulation is endowed to the agent by the agent designer [16] [17]. In our approach, the domain knowledge used by the agent is previously extracted from the actual human organizational actor being simulated. This knowledge extraction process is done through a computational system that monitors the actual human actors performing cooperative tasks of the same class of those used in the simulation.

To conveniently simulate the expected behaviour of organizational actors, it was essential to adopt a cognitive approach, namely an agent that has explicit and symbolic representation about itself, about the other agents and its environment, and reasoning capabilities. For those representations, the BDI architecture [2][4] has been chosen since it is a successful and respected theory of human behaviour.

BDI architectures do not formalize the way goals are generated, and they also do not formalize the way goals are selected to become intentions. Some authors [10] have implemented a motivational architecture that works like an extension to the BDI principles, bridging the previously mentioned gaps.

According to Savage [12], motivation is a process where the individual decides what is important to its continued existence and determines the means of achieving the relevant goals. This process is responsible for determining the deliberative human behaviour in all activities, including brainstorm activities.

Simon [14] and Sloman [15] have presented complex and fairly complete conceptual models of motivation, but none are described with the necessary detail for being expressed computationally. Our approach is inspired in the ideas put forth by these two authors but providing the necessary detail for implementation.

Avila-Garcia and Canãmero [1] and Breazeal [3] have presented computational motivation models. These models are far from implementing the properties identified by Simon and Sloman as optimal. No attention is paid to the influence of mental states (beliefs, goals and intentions) in the behaviour choice.

Motivation, besides influencing action control, also influences cognitive processes. This influence is determined by epistemic motivators [6], [7]. This kind of motivators influences the idea formulation, planning and any other kind of reasoning, including CBR. In our motivated CBR approach the amount of effort to be used in the various CBR processes (search, comparison and

adaptation), is determined by the importance and urgency of the motivation (intention) being handled, and the selection of the case to be used depends on the extent to which the prescribed action contributes to satisfy the agent motivation.

7. CONCLUSIONS AND FUTURE DEVELOPMENTS

This paper describes a decision making framework regarding human team formation in which a combination of cognitive multi-agent based simulation and an emergence model is used to simulate and evaluate the performance of all possible team compositions in order to identify a small set of the most promising ones.

In the described framework, the domain specific action control knowledge of each available organizational actor is acquired through a computational system designed to support his or her interaction with the other organizational actors in a selected collaborative task. The acquired models are used by cognitive agents in the simulation stage. This process considerably reduces the human decision-making workload.

The project selected the brainstorm as the test bed collaborative task, created a general model of domain independent brainstorms, implemented the computer system that supports distributed brainstorms, defined the models of the agents of the simulation, implemented the CBR system used for the agents model, and defined the initial set of domain independent criteria used in the performance evaluation of each simulated team composition.

An important contribution of this project is the proposed agent model. This model is an extension of BDI models, since its motivational model bridges two important gaps in that class of models: goal generation, and goal selection (i.e., intention formation).

Recently we have realised that the chosen brainstorm state representation for realistic examples becomes impracticably huge. So we will move towards a sentence based representation. Case comparison is also very resource consuming. Nevertheless, we expect the model of motivation will judiciously constrain the amount of computation assigned to each of the agent's tasks.

The project next steps are to implement the sentence based representation for the brainstorm states, to design and implement the emergence model, and to run the simulation process.

8. ACKNOWLEDGEMENTS

This work is supported by the FCT (Foundation for Science and Technology) National Project KnowledgeAndCulture.org contract nº POSI/SRI/46582/20027.

9. REFERENCES

[1] Avila-Garcia, O., Cañamero, L. Comparing a Voting-Based Policy with Winner-Takes-All to Perform Action Selection in Motivational Agents. In *Proc 8th Iberoamerican Conference on Artificial Intelligence (IBERAMIA'02)*, 2002.

[2] Bratman, M. E. *Intentions, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA, 1987.

[3] Breazeal, C., Scassellati, B. Infant-like social interactions between a robot and a human caretaker. *Adaptive Behavior*, 8, 1, (2000), 49–74.

[4] Georgeff, M. P., Barney P., Pollack, M. E., Tambe, M., Wooldridge, M. The Belief-Desire-Intention Model of Agency. *ATAL 1998*, (1998), 1-10.

[5] Kolodner, J. *Case-based reasoning*. San Mateo, CA: Morgan Kaufmann, 1993.

[6] Kruglanski, A. W., Klar, Y. Knowing what to do: on the epistemology of actions. In Kuhl, J. and Beckmann, J. (eds): *Action Control: from cognition to behaviour*. Springer, Heidelberg, 1985, 101-128.

[7] Kunda, Z. The case for motivated reasoning. *Psychological Bulletin*, 108, 3 (1990), 480-498.

[8] Lopes, A., Esteves, S., Gonçalves, B., Botelho, L. M., BAT:Brainstorm Assistance Tool. *Mobile Computing Meets Knowledge Management Workshop*, Prague, Czech Republic, 15-17 September, 2005.

[9] Louçã, J. *Cartographie Cognitive, Réflexion Stratégique et Interaction Distribuée: une Approche Multi-Agent*. PhD Thesis, Paris IX Dauphine University and Sciences Institute of Lisbon University, 2000.

[10] Norman, T. J., Long, D. Alarms: An implementation of motivated agency. In M. Wooldridge, J.P. Muller, and M. Tambe, editors: *Intelligent Agents: Theories, Architectures, and Languages*, LNAI 1037, Springer, 1996, 219-234.

[11] Pal, S., Shiu, S. Foundations of Soft-Case-Based Reasoning. John Wiley & Sons, 2004.

[12] Savage, T. The grounding of motivation in artificial animals: Indices of motivational behaviour. *Cognitive Science Research*, 4 (2003), 23-55.

[13] Shapiro S. C. An Introduction to SNePS 3. In Bernhard Ganter & Guy W. Mineau, Eds. *Conceptual Structures: Logical, Linguistic, and Computational Issues*. Lecture Notes in Artificial Intelligence 1867, Springer-Verlag, Berlin, 2000, 510-524.

[14] Simon, H. A. Motivational and emotional controls of cognition. *Psychological Review*, 74, 1 (1997), 29-39.

[15] Sloman, A. Motives, mechanisms, and emotions. *Cognition and Emotion*, 1 (1987), 217-234.

[16] Sycara, K.: MultiAgent Systems. *AI Magazine*, 19, 2 (1998).

[17] Wooldridge, M. An Introduction to Multi Agent Systems. John Wiley & Sons, 2002.

[18] JADE – Java Agent Development framework – "JADE, a White Paper" http://exp.telecomitalialab.com/articolo.asp?id=2&idart=1.

[19] FIPA – Foundation for Intelligent Physical Agents specifications - http://fipa.org/repository/index.html

[20] XSP – eXtended Service Platform - http://we-b-mind.org/EN/software.html#XSP

Performance of Pheromone Model for Predicting Traffic Congestion

Yasushi Ando,
Yoshiaki Fukazawa
Department of Science and Engineering, Waseda University
Tokyo 169-8555, Japan
ando@fuka.info.waseda.ac.jp,
fukazawa@waseda.jp

Osamu Masutani,
Hirotoshi Iwasaki
DENSO IT Laboratory, Inc.
Tokyo 150-0002, Japan
{omasutani,hiwasaki}@d-itlab.co.jp

Shinichi Honiden
National Institute of Informatics
Tokyo 101-8430, Japan
honiden@nii.ac.jp

ABSTRACT
Social insects perform complex tasks without top-down-style control, by sensing and depositing chemical markers called "pheromone". We have examined applications of this pheromone paradigm towards realizing intelligent transportation systems (ITS). Many of the current traffic management approaches require central processing with the usual risks of overload, bottlenecks and delay. Our work points towards a more decentralized approach that may overcome those risks. We use new category of the ITS infrastructure called the probe-car system. The probe-car system is an emerging data collection method, in which a number of vehicles are used as moving sensors to detect actual traffic situations. In this paper, a car is regarded as a social insect that deposits multi-semantics of (digital) pheromone on the basis of sensed traffic information. We have developed a basic model for predicting traffic congestion in the immediate future using pheromone. In the course of our experimentation, we have identified the need to properly tune the model to achieve acceptable performance. Therefore, we refined the model for practical use. We evaluate our method using real-world traffic data and results indicate applicability to prediction. Furthermore, we describe the practical implications of this method in the real world.

Categories and Subject Descriptors
I.6.3 [**Simulation and Modeling**]: Applications

General Terms
Algorithms, Performance, Experimentation

Keywords
pheromone, traffic congestion prediction

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8–12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005…$5.00.

1. INTRODUCTION
Many social insect species such as ants, bees, and wasps coordinate the activities of individuals in the colony without direct communication or complex reasoning. Instead, they deposit and sense chemical markers called "pheromone" in a shared physical environment (pheromone potential field) that participates actively in the system's dynamics. We will examine applications of this pheromone paradigm towards realizing intelligent transportation systems (ITS).

ITS have spread widely recently. Beyond car navigation systems and traffic information systems [22], various types of sensors and communication devices are being introduced for various kinds of applications. Solving traffic congestion is a serious problem among such applications.

One of the key technologies for solving this problem is traffic prediction. There are traffic information systems which are now operational in many countries. Drivers can easily grasp the current surrounding traffic situations through traffic information systems. However, the problem of traffic information system is that traffic information may be delayed due to communication between the road and data center or due to processing time at the data center. This leads to a driver often receiving traffic congestion data that are different from the actual traffic situation.

Another problem in solving traffic congestion is that current traffic data will become useless after a while on a given route. Traffic prediction techniques can solve this problem by predicting appropriate data for each stage of traffic data on the route. Traffic prediction techniques are widely studied in many institutions [4] and have already been introduced in commercial car navigation systems [23]. Some of such techniques deal with long-term predictions, such as predictions spanning 1 hour to 1 day based on statistical analysis of past data. However, long-term predictions cannot be applied to driving in a small area and cannot predict the innately sharp fluctuation of traffic. Short-term prediction is suitable for driving in a small area and for improving prediction accuracy.

Short-term prediction methods have also been studied [24, 26].However, most of the studies use statistical data of the past. As such, it does not work sufficiently with sharp or irregular fluctuations in traffic.

We propose a short-term traffic prediction method using the pheromone model, which has a relatively simple mechanism that

does not require statistical data. In our method, a car is regarded as an agent or an ant. Each car would deposit some pheromone into a virtual space with its amount based on their sensing of traffic situations.

To implement our idea as an actual service, we use new categories of the ITS infrastructure called the probe-car system (it is also called the floating car system) [19]. The probe-car system is an emerging data collection method, in which a number of vehicles are used as moving sensors to detect actual traffic situations. Our new method realizes traffic prediction by employing a pheromone mechanism and the probe-car information system without resorting to the use of a traffic control center.

We experimented with the pheromone prediction method the basis of real-world traffic data to evaluate its performance. The results indicate its applicability to the prediction of traffic congestion in the immediate future. Furthermore, we describe the relationship between model type and prediction accuracy.

2. PHEROMONE

2.1 Basic Mechanisms

The term “pheromone” (Greek for ``carrier of excitement'') was named by P. Karlson and M. Luscher in the 1950s [11]. Pheromone is defined as a communication chemical that possesses certain meanings that are mutually understood by individuals of the same species. Pheromones have various kinds of functions (flavors). For example, ants use trail pheromone (attractive sign) to guide other ants to food, and bees use alert pheromone (repulsive sign) to inform of disturbance to other bees.

The trail pheromone mechanism has already been applied to certain areas to efficiently solve problems such as the traveling salesman problem [6]. The mechanism works as follows: Ants find food and transport it to their nest, depositing trails of pheromone on their return path. The pheromone diffuses into the surrounding environment and its intensity weakens over time. However, the density of pheromone left on the path where ants have passed most frequently increases. Eventually, pheromone on the shortest path to the food becomes most dense (see Figure 1).

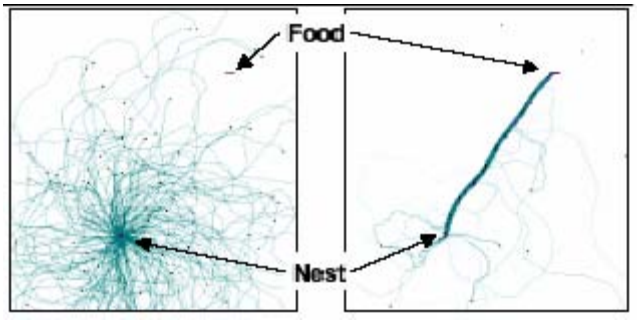


Figure 1: Action pattern of ants which used trail pheromone [20].

On the other hand, bees use alarm pheromone. A leading bee would inform of any danger using alarm pheromone and other bees that follow would avoid the alarm pheromone. This simple mechanism has also been applied to some technologies [16].

2.2 Characteristics

Pheromone is a simple chemical media for delivering information. Although each individual ant or bee possesses just a simple and local behavior rule, they are capable of enabling complex functions. This is largely related with the characteristics of “evaporation”' and “propagation” [5]. “Evaporation” represents old information disappearing as time passes due to the volatile nature of pheromone. This enables other ants to receive the most recent information. “Propagation” represents the characteristic of pheromone spreading out towards the surrounding environment. This allows other ants in different places to receive this information.

3. MODEL FOR TRAFFIC

In this paper, we discuss the use of this dynamism to predict traffic. The model used here will regard cars as ants or bees and will deposit pheromone on the pheromone potential field. Cars equipped with some kind of sensors would deposit multiple pheromone on the basis of sensed traffic information. Other cars that follow their route would avoid traffic congestion by checking the intensity of pheromone.

3.1 Digital Pheromone

We propose a traffic prediction method based on digital pheromone [21]. Digital pheromone is modeled on the pheromone fields that many social insects use to coordinate their behavior. A formal model of the essential components of these fields has been developed and applied to a variety of problems. Digital pheromone is defined as having the following characteristics:

1. Pheromone is information deposited on a certain place by agents when an event occurs (Information aggregation)
2. Deposits of a certain flavor are added to the current amount of that flavor of pheromone located at that place. Total amount represents the information potential of each location (Information fusion)
3. Pheromone evaporates over time thus decreasing its value (Truth maintenance)
4. Pheromone propagates towards the surrounding environment (Information diffusion and dissemination)
5. Each agent is affected by pheromone.(Decision making)

Following the above this definition of digital pheromone, we define traffic prediction pheromone as follows.

1'. Pheromone is information deposited by cars on the basis of the surrounding traffic situation.

2'. Deposits of a certain flavor (certain sensing traffic data) are added to the current amount of that flavor of pheromone located on the road. Total amount represents the traffic situation at each location.

3'. Pheromone evaporates over time. This corresponds to traffic moving towards another location over time.

4'. Pheromone is propagated towards nearby roads. This corresponds to traffic movement along the flow.

5'. Each driver is then affected by the traffic prediction based on pheromone.

3.2 Pheromone Vocabulary

Several "flavors" of pheromone are used to improve system performance. For example, in this study [17], multiple flavors of pheromone such as target, nest and threat pheromone are used to find the optimal path to a target.

We have explored several basic mechanisms essential to the engineering deployment of pheromone mechanisms. In our traffic scenario, there also exist various kinds of factors that affect near-future traffic. We will present three kinds of traffic pheromones and explore the performance of the system for various combinations of these three pheromones. The three pheromone flavors are as follows:

3.2.1 Basic Traffic Pheromone

This is a pheromone of repulsion which informs other cars about the possibility of an increase in traffic congestion. Each car, equipped with a speed sensor and a communication device, deposits basic traffic pheromone according to congestion rate. Generally, speed is used to represent congestion rate, so a car agent will deposit some amount of pheromone depending on its speed.

Let the external deposit of basic traffic pheromone of pheromone flavor *traffic* at time t and location p be $d(\Phi_{traffic},t,p)$. $v_i(t,p)$ is the speed corresponding car ID i and $C(t,p)$ is the set of car IDs within the interval $(t-1, t]$ at location p. Location p represents a road link or some span of a road between intersections. This function is represented as follows.

$$d(\Phi_{traffic}, t, p) = \frac{1}{|C(t,p)|} \sum_{i \in C(t,p)} \frac{1}{v_i(t,p)} \quad (1)$$

Equation (1) implies the following:

- The deposit increases with decreasing speed (Figure 2 (a)).
- The deposit decreases with increasing speed (Figure 2 (b)).

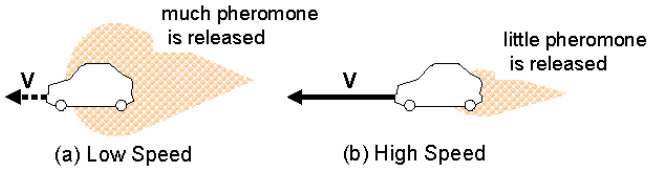


Figure 2: Image of car agents depositing basic traffic pheromone

3.2.2 Braking Pheromone

This is also a pheromone of repulsion which informs other cars about the possibility of an increase in traffic congestion. Each car, equipped with a brake sensor, deposits this repulsive pheromone. Generally, stepping on brakes causes traffic congestion; thus, a car agent will deposit some amount of pheromone depending on the number of times its brakes are applied.

Let the external deposit of braking pheromone of pheromone flavor *braking* at time t and location p be $d(\Phi_{braking},t,p)$. $braking_i(t,p)$ is the number of times stepping on the brakes is performed corresponding to car ID i and $C(t,p)$ is the set of Car IDs within the interval $(t-1, t]$ at location p.

$$d(\Phi_{brake}, t, p) = \frac{1}{|C(t,p)|} \sum_{i \in C(t,p)} braking_i(t,p) \quad (2)$$

3.2.3 Distance Pheromone

Different from above two pheromones, this is the pheromone of attraction which informs other cars about the possibility of a decrease in traffic congestion.

A car equipped with a millimeter-wave radar deposits this pheromone. Generally, the distance between cars increases with decreasing congestion level. Thus, a car agent will deposit some amount of pheromone depending on its distance between cars.

Let the external deposit of distance-between-cars pheromone of pheromone flavor *distance* at time t and location p be $d(\Phi_{distance},t,p)$. $dis_i(t,p)$ is the intercar distance corresponding to car ID i and $C(t,p)$ is the set of Car IDs within the interval $(t-1, t]$ at location p. $T_{distance}$ represents the threshold below which $dis_i(t,p)$ is set to zero.

$$d(\Phi_{distance}, t, p) = \frac{1}{|C(t,p)|} \sum_{i \in C(t,p)} dis_i(t,p) \quad (dis_i(t,p) > T_{distance}) \quad (3)$$

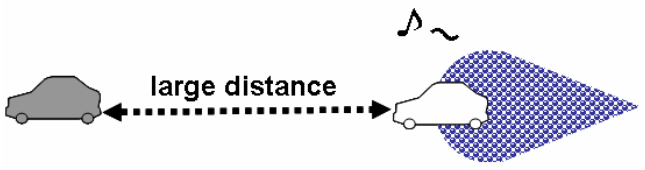


Figure 3: Image of car agents depositing distance pheromone

3.3 Transition Function

Each place maintains a scalar variable corresponding to each pheromone flavor. Our model is based on the pheromone transition model proposed by Brueckner [2]. Brueckner's model is a kind of coupled map lattice (CML) [12] or continuous cellular automaton. It performs the basic functions of aggregation (as mentioned above), evaporation and propagation.

Evaporation: Aggregated pheromone decreases over time due to evaporation. Let the amount of pheromone of pheromone flavor f at place p and time t be $s(\Phi_f,t,p)$. The predicted amount of a single pheromone flavor $s(\Phi_f,t+1,p)$ is represented as follows.

$$s(\Phi_f, t+1, p) = E_f * |(1 - G_f) * (s(\Phi_f, t, p) + d(\Phi_f, t, p)) + g(\Phi_f, t, p)| \quad (4)$$

E_f reflects evaporation of pheromone, the 1-G_f factor calculates the amount remaining after propagation to its neighbors, $d(\Phi_f,t,p)$ represents the total deposits made since that last update cycle, and $g(\Phi_f,t,p)$ represents the total pheromone propagated in from all the neighbors of p.

Propagation: Another fundamental equation describes the propagation received from neighboring places. The transition function of the propagated amount $g(\Phi_f,t,p)$ is shown as follows.

$$g(\Phi_f, t, p) = \sum_{p' \in N(p)} \frac{G_f}{|N(p')|} (s(\Phi_f, t-1, p') + d(\Phi_f, t-1, p')) \quad (5)$$

N(p') represents the upper stream neighbors of location *p* that affect the next status of location *p*. G_f is the propagation rate, which is a function of the pheromone value of the upper stream in our model.

3.4 Combination of Multiple Pheromones

In a previous section (section 3.2), we explained three kinds of pheromone and classified two types of signal (A sign to make a traffic congestion increase and a sign to make a traffic congestion decrease). In this section, we describe how to combine different flavors of pheromone. This mechanism is based on how each pheromone affects predicted link travel time. We propose a combination method of three patterns in addition to a basic model.

First, we describe only uni-flavor pheromone, that is, basic traffic pheromone. When using it, we derive Equation (6). Predicted link travel time is a scaled pheromone amount based on a scaling parameter and the length of each link. Let *Prt(t,p)* be the predicted link travel time, *S* be the global scaling parameter and *l(p)* be the length (*m*) in a link *p*.

$$\Pr t(t, p) = s(\Phi\ traffic\ , t, p) \times S \times l(t, p) \quad (6)$$

Second, we use two flavor pheromones, that is, traffic pheromone (repulsive pheromone) and distance pheromone (attractive pheromone). We derive Equation (7). Distance pheromone makes a traffic congestion decrease, so predicted travel time is given as follows.

$$\Pr t(t, p) = (s(\Phi traffic, t, p) - s(\Phi distance, t, p)) \times S \times l(t, p) \quad (7)$$

Third, we use another set of two pheromones, that is, basic traffic pheromone (repulsive pheromone) and braking pheromone (repulsive pheromone). We derive Equation (8). Braking pheromone makes a traffic congestion increase further, so predicted travel time is given as follows.

$$\Pr t(t, p) = (s(\Phi traffic, t, p) + s(\Phi braking, t, p)) \times S \times l(t, p) \quad (8)$$

Finally, we use the all above three pheromones, that is, basic traffic pheromone (repulsive pheromone), distance pheromone (attractive pheromone), and braking pheromone (repulsive pheromone). We derive Equation (9).

$$\Pr t(p) = (s(\Phi_{traffic}, p) - s(\Phi_{distance}, p) + s(\Phi_{braking}, p)) \times S \times l(t, p) \quad (9)$$

Here, we only used a linear combination of pheromones, and the combination ratio of each semantic of pheromone was tuned by sensitivity analysis.

4. PRACTICAL IMPLICATION

In this section, we discuss the above-described pheromone model as it might be implemented in actual services. To develop the pheromone system, we must first consider its system architecture. An implementation of a pheromone system has two components: the agents (which deposit and react to the field maintained by the environment), and the environment (which maintains the pheromone field and performs aggregation, evaporation, and propagation). Our implementations use an ITS infrastructure which currently exists.

The internet-based ITS project in Japan is moving ahead through public and private sector cooperation to create new categories of ITS services. One of the key technologies is the probe-car system (Figure 4). The probe-car information system utilizes vehicles as mobile sensors for collecting traffic data, which are stored and processed to produce new information useful for providing various public services, among other applications. It can collect data over a much larger area at a much higher precision data, than conventional fixed detectors. Many experiments are carried out using of "taxi-type" or "bus-type" probe-cars in many countries [7, 19]. Moreover, a commercial probe-car service mounted on common vehicles through a cellular network has started in Japan [8]. We use these probe-cars as agents (which deposit pheromone and calculate travel time by reacting to pheromone).

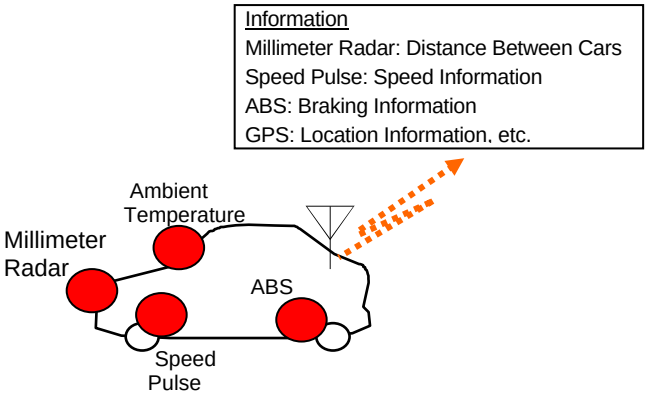
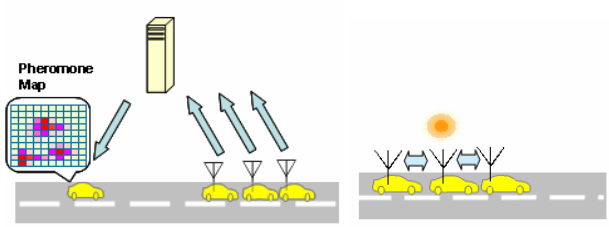


Figure 4: Probe-car system

On the other hand, there are two ways of implementing the environment. One method is to realize a virtual pheromone map on the car navigation system. Each car receives surrounding probe information from the probe-car information server, then maps it and calculates pheromone on the digital map in the car navigation system (Figure 5 (a)). The other method is to realize a virtual pheromone environment using intervehicle communication technology. Some researchers have proposed a virtual information server on an inter-vehicle communication system [18], such that these technologies would make a complete distributed system which is called a "center-less probe car system" (Figure 5 (b)). This will bring about effectiveness in pheromone calculation because no communication between the probe-server and cars in required.



(a) Server-type Probe-car System **(b) Center-less Probe-car System**

Figure 5: Realization of Pheromone System

5. EVALUATION

We conducted a simulation of our model using actual traffic benchmark data. Then we evaluated how accurate our model predicts traffic on the basis of error indices. This section describes the data used in the simulation and simulator specification.

5.1 Data

The data we chose has high time granularity and a relatively sharp fluctuation in traffic. The real data used in the simulation is the "Kichijoji-Mitaka benchmark dataset" provided by i-Transport Lab. Co., Ltd. This dataset consists of road link information, complete trail data of approximately 12000 cars in an area between Kichijoji and Mitaka stations on a certain day (see Figure 6) [9]. These data were collected by human observers at about 100 observation points in the area. Each trail data includes license plate data and time data to the minute, representing the time at which a car passed an observation points.

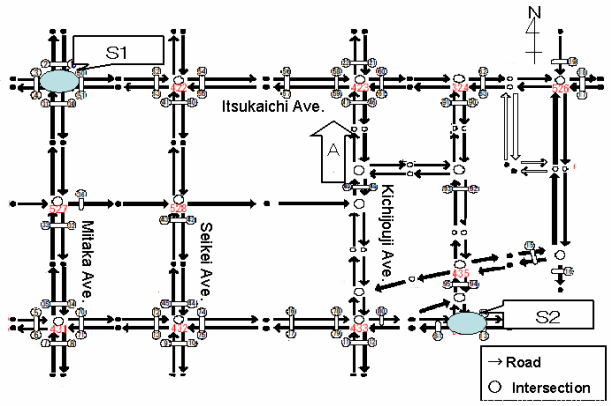


Figure 6: Map of the area where actual survey data were observed

5.2 Simulator

We use Pramics which is a widely used commercial microscopic traffic simulation platform [15] to evaluate the pheromone model and some other baseline model. It calculates the error rate of these models. It also has visualization features to read actual traffic and predicted traffic through animation (See Figure 7).

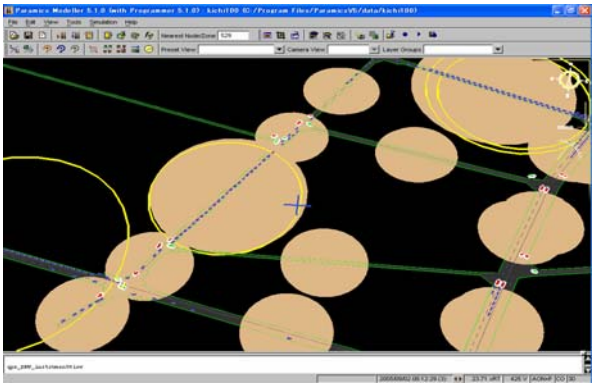


Figure 7: Screenshot of Simulator. (Shaded circles denote pheromone on each road link)

5.3 Assumptions for Simulation

The average link travel time is around 1 minute since the average link length on the map is around 1 km and the speed of the fastest car is around 60 km/h. Therefore, it would take 1 minute for a car to reach the next link after a prediction has been provided from a preceding car. Thus, in the simulation, the transition time step is set to 1 minute and the pheromone model predicts the situation 1 minute into the future (See Figure.8)

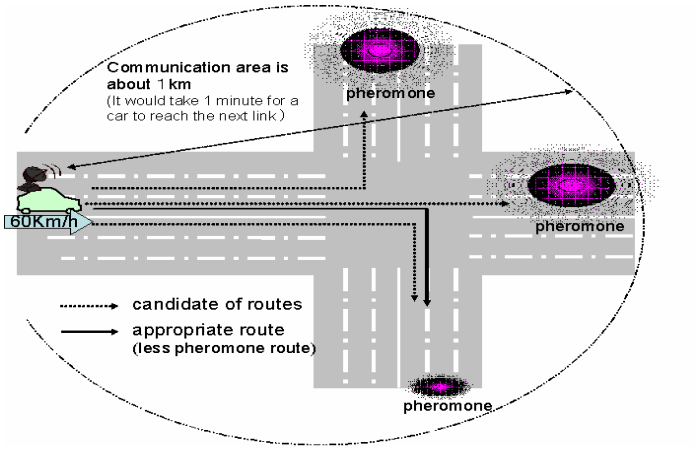


Figure 8: A Communication range of a car and relation of prediction time.

6. EXPERIMENTAL RESULTS

Evaluation has been performed from the standpoint of (i) the prediction accuracy of the basic model, (ii) the relationship between model type (uni-semantics of pheromone or multi-semantics of pheromone) and prediction accuracy, and (iii) the relationship between model parameters and accuracy.

6.1 Prediction Accuracy of Basic Model

Prediction accuracy is calculated as error rates between the predicted link travel time sequence and the actual link travel time sequence. The error rate indices we used are correlation coefficients. The prediction results are compared with two types of baseline predictions. One is a prediction based on a simple moving average time. The other is a prediction based on a persistent prediction model which assumes that the current situation will persist for some duration of time. The traffic information by current traffic control center can be received within 5 minutes or more after sensing; thus, we can assume that the persistent prediction model corresponds with the data of traffic information system.

Figure 9 shows fluctuations of actual traffic and traffic predictions in about 1 hour between 8:00 and 9:00 at a certain link. Actual traffic is the average link travel time of all cars that passed the link at the time the actual data was collected. Predicted link travel times of three types of prediction method are also drawn in the same graph. The correlation coefficient is shown in Table 1. These results show that our method outperforms baseline predictions. We can say that baseline predictions become worse when traffic fluctuations become very sharp.

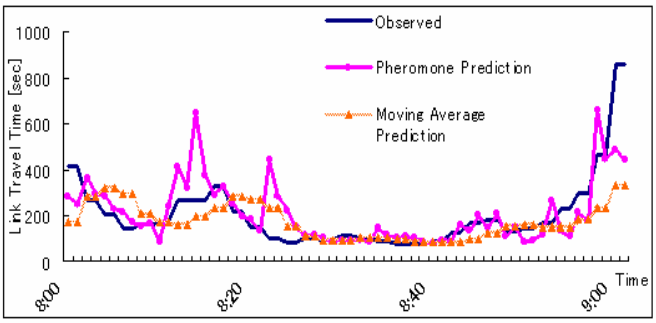


Figure 9: Prediction examples of each prediction methods in a certain link

Table 1. Accuracy of pheromone and other prediction schemes in a certain link

	Pheromone	Persistent	Moving Average
Correlation Coefficient	0.67	0.55	0.57

Additionally, the overall result of all links on the map, presented in Table 2, shows that pheromone prediction has higher accuracy than the other methods.

Table 2. Average accuracy of pheromone and other prediction schemes

	Pheromone	Persistent	Moving Average
Correlation Coefficient	0.36	0.34	0.30

6.2 Prediction Accuracy of Multiple Semantics of Pheromone

Next, we study the effect of different configurations of the pheromone model described in sections 3.2 and 3.4 (uni-semantics of pheromone or multi-semantics of pheromone).

The overall prediction accuracies of uni-semantic pheromone and multi-semantic pheromone are shown in Table 3. We confirmed the correlation coefficient of multi-semantic pheromone is significantly greater than that of the traffic pheromone.

Table3. Overall accuracy of uni-semantic and multi-semantic pheromone models

	Uni-semantics of Pheromone (only Traffic Pheromone)	Traffic + Braking	Traffic + Distance	Traffic + Braking + Distance
Corre-lation	0.36	0.44	0.38	0.45

The result implies that combining of some pheromones can be used to refine the prediction accuracy. Here, we only used a linear combination of pheromones that have different semantics. However, this scheme might contain nonlinear or logical combinations of semantics. Furthermore, more factors (such as ambient weather, traffic rules, and various events) might affect the prediction accuracy.

6.3 Parameter Modeling

In a previous work, the parameter of this pheromone model was tuned by learning algorithms such as GA [13, 14]. However, such a method is not so feasible for practical use because much learning data (empirical factor) and calculation costs are necessary.

In our experiment, we analyzed the relationship between the optimal parameters and real-world traffic data, and studied the effective parameters modeling schema to simplify determination of optimal parameters. In this section, we introduce a few example of the parameter modeling that got the most influential result in experimental results. The two examples of experimental results are as follows:

Modeling Example 1

First, we take the following assumption regarding the evaporation parameter and evaluated the relationship.

When the velocity of cars in a certain place is fast, evaporation occurs frequently (evaporate much pheromone). The model function is as follows.

$$E_f = 1 - \frac{v(t,p)}{v_{\max}(t,p)} \qquad (10)$$

Equation (10) implies the following.

- The evaporation decreases with decreasing speed at the given place (much congestion information remains behind).
- The evaporation increases with increasing speed at the given place (much congestion information disappears).

Modeling Example 2

Second, we take the following assumption regarding the propagation parameter and evaluated the relationship.

When the velocity of cars in a neighboring place is fast, propagation occurs frequently (propagate much pheromone). The model function is as follows.

$$G_f = 1 - \frac{v(t,p')}{v_{\max}(t,p')} \qquad (11)$$

Equation (11) implies the following.

- The propagation decreases with decreasing speed at the neighboring place (much congestion information remains behind).
- The evaporation increases with increasing speed at the neighboring place (much congestion information disappears).

Results of the parameter modeling of the evaporation parameter and the propagation parameter are shown in Table 4. We confirmed that modeling of pheromone parameters based on real-world traffic data can be used to refine the prediction accuracy.

We tried much variation of parameter modeling; however, they cannot be discussed here for lack of space.

Table 4. Impact of Parameter Modeling

	Basic Model	Example 1 After Modeling	Example 2 After Modeling
Correlation Coefficient	0.36	0.42	0.49

7. RELATED WORK

An application of the pheromone mechanism has been studied in the so-called swarm intelligence research field. On the other hand, traffic congestion prediction has various kinds of solutions such as the statistical method and simulation-based method. Such related works are discussed in the following sections.

7.1 Application of Pheromone Mechanism

There are various studies concerning applications of the pheromone mechanism. This study regards cars as mobile agents that deposit pheromone, which is similar to the approaches used in actual agents for sensors or planes.

One example of an application is an unmanned plane in a war zone [16, 20]. The unmanned plane would launch mobile agents which would deposit alarm pheromone when they encounter enemy missile sites. The plane would analyze the pheromone potential field and then determine a route to its destination.

Another example of an application is a mobile sensor network [10]. Studies in this field of application are concerned with methods to improve the efficiency of the overall monitoring task by making mobile sensors deposit alarm pheromone to indicate that it has observed a certain area so that other mobile sensors can avoid observing the same place again.

These researches use alarm (negative) pheromone while most of the researches regarding swarm intelligence use trail (positive) pheromone. These researches also deal with the flexible nature of pheromone appropriately. However, note that previous research studies involve evaluations using virtual data and not actual data. Our study has performed evaluations using real traffic data to confirm the applicability to a practical traffic system.

7.2 Traffic Congestion Prediction

There are two types of short-term traffic congestion prediction methods in the field of traffic engineering research. One is based on statistics of past traffic fluctuation patterns. The other is based on traffic simulation.

Statistical prediction is performed by matching current traffic patterns (time series data) to typical traffic patterns in the past. The most applicable pattern among the past traffic patterns is used for future-pattern prediction. The time span may vary from minutes to days depending on the application.

The matching process for short-term prediction uses algorithms such as those based on neural networks [24] or Bayesian networks [26]. The cost incurred for processing is significant since large amounts of past traffic history data are used. Furthermore, prediction of irregular traffic patterns cannot be achieved using only this pattern-matching-based method.

On the other hand, traffic simulation is used for short-term traffic prediction [1]. The simulation based model describes the mutual interference of cars that consists of running car dynamics, various road types, traffic flow model, and driver model. Thus, the simulation tends to be complicated and time consuming. Distributive methods of simulation such as cellular automata (CA) are also introduced into this traffic prediction scheme [3]. While prediction using CA has proved to be effective when information regarding incoming stream volume is available, it does not seem to be effective when the information is partially not available, such as in the case of normal road data.

Naturally, since these predictions are carried out based on information from a traffic information center, data will have low granularity (only three levels: "normal," "congested," and "stacked").

The traffic prediction method using pheromone which we have introduced does not require past traffic data and much processing cost, and it works with a relatively simple mechanism.

8. CONCLUSION

In this paper, we proposed a traffic prediction method that employs a pheromone mechanism and described the practical implication of this method. Cars are regarded as agents that deposit pheromone at virtual places. The pheromone would then evaporate and propagate on the basis of a modified version of the state transition model of digital pheromone. A car would be able to predict traffic on the road ahead from the information provided by preceding cars. On the basis of the results of the experiments that used real traffic data, we confirmed the applicability of our method to short-term traffic prediction.

Moreover, we introduced multiple semantics of pheromone as well as parameter modeling of the evaporation and propagation parameters. We confirmed that both approaches can refine the prediction accuracy of the method. However, we only used a linear combination of multiple pheromones; thus, we have to examine nonlinear or logical combinations of semantics because the semantics can be affected by factors such as traffic rule, some buildings, various events on road side, and weather. The contribution of such factors to traffic might be optimized by GA.

Furthermore, we have to validate the global optimization of this traffic system through a multi-agent traffic simulation. Because previous research efforts have revealed that individually optimizing performance with only traffic congestion information by traffic control center is difficult [25], thus, we believe that swarm-based decentralized traffic system is best approach that may achieve the global optimization.

9. REFERENCES

[1] M. Ben-Akiva, M. Bierlaire, H. Koutsopoulos, R. Mishalani. DynaMIT: a simulation-based system for traffic prediction, Paper presented at the DACCORD Short Term Forecasting Workshop February,1998

[2] S. Brueckner. Return from the Ant: Synthetic Ecosystems for Manufacturing Control. Thesis at Humboldt University Department of Computer Science, 2000

[3] E.G. Campari, G. Levi. A Realistic Simulation for Highway Traffic by the Use of Cellular Automata, ICCS2002,2002

[4] E.Chung, Classification of Traffic Pattern,Proc. of the 11th World Congress on ITS,2003.

[5] M. Dorigo, G. D. Caro,L. M. Gambardella, Ant Algorithms for Discrete Optimization, Arti.cial Life, MIT Press, 1999

[6] M. Dorigo, L. M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation, 1(1):53--66, 1997

[7] W. Huber, M. Ladke, and R. Ogger, Extended floating-car data for the acquisition of traffic information, In 6th World Congress on Intelligent Transport System, November 1999

[8] HONDA Motor Co.,Ltd http://premium-club.jp/PR/

[9] R. Horiguchi, T. Yoshii, H. Akahane, M. Kuwahara, M. Katakura, H. Ozaki and T. Oguchi., A Benchmark DataSet for Validity Evaluation of Road Network Simulation Models, Proceedings of 5th World Congress on Intelligent Transport Systems, 1998

[10] M. Howard, D. Payton, R. Estkowski. Amorphous Predictive Nets, International Conference on Complex Systems (ICCS2002),2002.

[11] P, Karlson, M. Luscher. Pheromones': a new term for a class of biologically active substances, Nature,1959

[12] K. Kaneko. Period-doubling of Kink-antikink Patterns, Quasi-periodicity in Antiferro-like Structures and Spatial Intermittency in Coupled Map Lattices --- toward a prelude to a ``Field Theory of Chaos"---,Prog. Theor. Phys. 72,1984

[13] O. Masutani, H. Sasaki, H. Iwasaki, Y.Ando, Y.Fukazawa, S. Honiden, Pheromone Model: Application to Traffic Congestion Prediction, Proc. 4th International Joint Conference on Autonomous Agents and Multi-Agent System(AAMAS'2005), 2005

[14] O. Masutani, H. Sasaki, H. Iwasaki, Y.Ando, Y.Fukazawa, S. Honiden, Traffic Prediction using Pheromone Model, Proc. 12th World Congress on Intelligent Transport Systems, 2005

[15] Paramics: Microscopic Traffic Simulator Developed by Quadstone Ltd. http://www.paramics-online.com/

[16] H.V. D. Parunak, S. Brueckner, J. Sauter, J. Posdamer. Mechanisms and Military Applications for Synthetic Pheromones, Proceedings of Workshop on Autonomy Oriented Computation, Agents 2001,2001

[17] H. V. D. Parunak, Sven A. Brueckner, John A. Sauter, Synthetic Pheromone Mechanisms for Coordination of Unmanned Vehicles, First International Conference on Autonomous Agents and Multi-Agent System(AAMAS'2002), 2002

[18] M. Saito, M. Funai, T. Umedu, T.Higashino, Inter-Vehicle Ad-hoc Communication Protocol for Acquiring Local Traffic Information, Proc. 11th World Congress Intelligent Transport System, 2004

[19] M. Sarvi, R. Horiguchi, M. Kuwahara, Y. Shimizu, A. Sato and Y. Sugisaki, A Methodology to Identify Traffic Condition Using Intelligent Probe Vehicles, 10th World Congress on Intelligent Transport Systems, Madrid, 2003.

[20] J. A.Sauter, R. Mattews, H.Van Dyke Parunak, S. Brueckner, Evolving Adaptive Pheromone Path Planning Mechanisms, Proc. First International Conference on Autonomous Agents and Multi-Agent System(AAMAS'2002), 2002

[21] J. A. Sauter, R.Matthews, H. V. D. Parunak, S. Brueckner, Performance of Digital Pheromones for Swarming Vehicle Conatrol, Fourth International Conference on Autonomous Agents and Multi-Agent System(AAMAS'2005)

[22] Traffic.com, http://www.traffic.com/

[23] TOYOTA Motor Corporation, G-BOOK Alpha, http://g-book.com/

[24] VICS(Vehicle Information and Communication System) http://www.vics.or.jp/english/index.html

[25] R. Yasdi, Prediction of Road Traffic using a Neural Network Approach,Neural Comput Applic (1999) Springer-Verlag London Limited,1999.

[26] T. Yoshii, H. Akahane, M. Kuwahara, Impacts of the Accuracy of Traffic Information in Dynamics Route Guidance Systems, Proc. Of the 3rd Annual World Congress on Intelligent Transport Systems, 1996

[27] C. Zhang, S. Sun, G. Yu, Short-Term Traffic Flow Forecasting Using Expanded Bayesian Network for Incomplete Data, Lecture Notes in Computer Science Volume 3174, 2004.

Columns on Last Page Should Be Made As Close As Possible to Equal Length

Modeling Agents and Interactions in Agricultural Economics

Daisuke Torii
Department of Social Informatics, Kyoto University
Kyoto, 606-8501, Japan
torii@kuis.kyoto-u.ac.jp

Toru Ishida
Department of Social Informatics, Kyoto University
Kyoto, 606-8501, Japan
ishida@i.kyoto-u.ac.jp

Francois Bousquet
CIRAD
GREEN Research Unit,
Cirad-Tera, Montpellier, France
bousquet@cirad.fr

ABSTRACT

When multiagent simulations are used for consensus building among stakeholders, it is important not only that the domain experts can deeply understand stakeholders' actual behavior but also that the stakeholders can feel the simulation result as their solution. To this end, we propose a modeling methodology which combines several techniques with the participatory method which takes stakeholders into the modeling process using role playing games (RPG).

There are two types of model required to simulate a social system as a multiagent system: agents (internal models) and interactions. Hence, we considered a modeling method according to each character. In modeling an agent (e.g. decision making) which is implicit in human, the identification of the model greatly depends on the modeler's ability. Therefore we propose a modeling method wherein classification learning creates an alternative model from RPG log data for validating the domain experts' hypothesis. On the other hand, in modeling interactions (e.g. negotiation) which are emerged outside of human, it is rather important to show and capture continuously appeared interactions. Therefore we propose a modeling method with participatory simulation where a stakeholder participates as an avatar and agents act as the other stakeholders in order to deeply understand the stakeholders' interactions. Our methodology was effective to give the domain experts a deeper understanding through a real case study of agricultural economics in the northeast of Thailand [17].

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Design

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Keywords

multiagent simulation, participatory simulation, machine learning, agent interaction, modeling methodology

1. INTRODUCTION

Multiagent simulation has been used to analyze and design social systems (traffic [1], disaster management [6][11], market [14] etc.). The general modeling process is that a model created by the domain knowledge is adjusted to output the same kind of results as the real data. However, when using multiagent simulations for consensus building among stakeholders, it is important not only that the domain experts can deeply understand stakeholders' actual behavior but also that the stakeholders can feel the simulation result as their solution. In this paper, we aim to provide a modeling methodology to satisfy these points.

The participatory method by social scientists is suitable for this purpose [3][4]. The key idea is that stakeholders can understand the simulation result as their solution because they share the modeling process with the domain experts and can believe that the model reflects their actual behavior. A role playing game (RPG) where stakeholders play on a board representing their environment is used to share the modeling process and to well understand their actual thinking and behavior. "Participation" is the key aspect of our modeling methodology.

To simulate a social system as a multiagent system, two types of model are required: agents (internal models) and interactions. For example, in an economic system in agriculture described in this paper, decision making models and negotiation models of farmers and seed suppliers are required. Hence, we have to consider a modeling method according to each character.

In modeling an agent (e.g. decision making) which is implicit in human, the identification of the model greatly depends on the modeler's ability. Therefore, a method for model validation is very important. On the other hand, in modeling interactions (e.g. negotiation) which are emerged outside of human, it is rather important to show and capture continuously appeared interactions. RPG using a board in the participatory method is not enough to express such dynamic process as interactions, and it is important to give a method for this purpose.

In this study, to give solutions of each issue above, the following technological methods are applied in the notion of the participatory method.

Modeling agents with classification learning

A hypothesis of domain experts is validated by a model which classification learning creates from RPG log data. The merits are 1) classification learning creates an objective model without influence of the modeler's ability because the creation is based on the algorithm, and 2) classification learning is more robust than human in performance even when the number of elements to be compared is large. What is important here is to consider a method that classification learning creates a model which logically explains the stakeholders' behavior even if the data available is sparse (it is difficult to gather enough data for classification learning because of the cost of RPG).

Modeling interactions with participatory simulation

In participatory simulations, stakeholders participate as avatars and agents act as the other stakeholders. The merits of using this kind of simulation are 1) it is easy to express and record every interaction on a computer, and 2) domain experts can concentrate on a specific stakeholder because agents act as the other stakeholders. What is important here is to create an environment that supports the domain expert in designing agent-avatar interaction so as to elucidate the behavior of the stakeholder.

Our modeling methodology was actually evaluated in an actual case study of agricultural economics in the northeast of Thailand in co-project of IRRI[1] and CIRAD[2] [17].

The following section will first explain the related works. Section 3 will explain the agricultural economics used in our evaluation. Sections 4 and 5 will explain the modeling method of agents with classification learning and interactions with participatory simulations, respectively, using our example.

2. RELATED WORKS

Two studies have used machine learning with participatory simulation for modeling. 1) One applied machine learning to the logs of participatory simulations for developing diverse agent models in a virtual training system [12]. 2) The other allowed agents to learn effective negotiation strategies for the participatory simulations of a virtual market [9]. In these two studies, participatory simulation is not used to share the experience of domain experts and stakeholders. Moreover, machine learning is not used for validating a hypothesis of domain experts.

We know of two studies which use participatory simulation. 1) One developed a web forum of agents and stakeholders for holding a continuous RPG free from the constraints of time and place [4]. 2) The other used participatory simulations in which only humans participated as avatars so as to eliminate a drawback with the RPG approach in which roles of participants are fixed and emergence of new roles cannot be handled [5]. In the first study, participatory simulation is intended to create a place for continuous meeting of stakeholders and this differs from our use of participatory simulation: to exploit interaction models of stakeholders. The second also mentions design processes of participatory simulation, but they are not for constructing an environment where agents and avatar participate.

[1]International Rice Research Institute, http://www.irri.org/

[2]Centre de Coopération Internationale en Recherche Agronomique pour le Développement, http://www.cirad.fr/

3. AGRICULTURAL ECONOMICS OF NORTHEASTERN THAILAND

In this research, we applied our proposed processes to the agricultural economics of northeastern Thailand in a collaborative research project with IRRI and CIRAD [17].

The background is as follows. In the northeast of Thailand, the Thai government distributes a few different rice varieties. The problem is that the organization of this distribution is not always efficient: farmers do not always get the seeds they want while the seeds of some varieties are produced in excess. Also, it seems desirable that this distribution should not reduce the use of local varieties. The problem is how to develop a system that can both deliver good quality seeds and also conserve rice biodiversity. Researchers have been studying two subjects: 1) understand farmers' needs concerning rice varieties and the selection model of rice varieties and 2) to identify the problems of the current delivery system of rice seed by acquiring a farmer's selection model of seed suppliers and a flow model of rice seeds among the seed suppliers and farmers.

To realize this economic system in multiagent simulation, we need to have a) decision making model of farmers for rice varieties, b) decision making model of farmers for seed suppliers, c) negotiation model of suppliers for rice seeds and d) negotiation model of farmers for rice seeds. In this study, we will apply our modeling process of decision making and negotiation to b) and c), respectively.

4. MODELING AGENTS WITH CLASSIFICATION LEARNING

In this section, we will discuss the issues and the solutions in using classification learning to model agents (internal models) and explain the processes involved with an example: decision making of farmers for seed suppliers. The decision making model is represented in a decision tree and we selected the well-known decision tree learning algorithm "C4.5" for classification learning [13].

4.1 Issues

The characteristics as a classification learning problem are as follows:

1. The number of features tends to be large because the experts take into account all elements that they think might be related to the classification. However, the more features that are included, the more data sets tend to be needed to assure reasonable learning performance [10]. Unfortunately, it is difficult to gather enough data because the cost of RPG is high (an RPG session usually engages 12-15 players for 2 days).

2. Noise data can affect the learning result because the amount of data may be insufficient.

3. The classification knowledge gained through learning should logically explain the real behavior of stakeholders and satisfy the experts.

4.2 Approaches

In our approach for these issues, expert knowledge is effectively used for obtaining a model even from insufficient amount of data sets. Here, the unneeded features are winnowed in two ways:

Feature Selection Method

The wrapper approach[3] [10] is used for eliciting a model with higher classification accuracy for unknown data; that is, a model with more generality. This method identifies a feature subset that maximizes the prediction accuracy as determined by cross validation[4], by eliminating irrelevant features from the initial feature set through hill-climbing search. In this method, however, when the quantity of data sets is insufficient, there is a possibility that some features needed for adequately reproducing reality will be eliminated due to noise data. Therefore, in our approach, the features contained in the hypothesis (in a decision tree) are regarded as being candidates to adequately explain reality and retained in the process (e.g. five features in figure 2 are not eliminated in this process). These features are always contained in the feature subset, but the learning algorithm selects a part of features from the subset, so the important features are not always contained in the learning result. This means that the wrapper approach can winnow the features to prevent the learning from selecting irrelevant features which do not ensure generality.

Visualization Method

In order to obtain a result which logically explains the stakeholders' behavior, the learning model is refined by eliminating irrelevant features contained in it and applying the classification learning again. Such conditions are selected by the domain expert. Here, It is important to consider an evaluation method that gives domain experts intuitive understanding of the model and excludes their bias to the model. To this end, the environment and participants of the RPG are reproduced on a computer using a graphical interface and computer agents. The expert is shown the results made by the agents who have the learning model and the RPG log data itself (the expert is not told which is from the RPG log data). Then he puts comments to each result. After this, the model itself is directly shown to the expert. He selects irrelevant features, comparing his comments to the visualized result with the model.

There are three merits on this method. First, a visualized method helps the expert to find apparently strange results and eliminate irrelevant features. Second, the learning result can be neutrality evaluated, while the expert tends to criticize the learning result when the model is shown from the beginning because he is obsessed with his hypothesis. Third, he can scrutinize the learning result because a model is re-evaluated based on his comments to the visualized result. The reason why RPG log data is also shown to the expert is to convince him of validity of the learning result (he can accept a part of the learning result when the output of it resembles the one from the RPG log even if it does not support his hypothesis.).

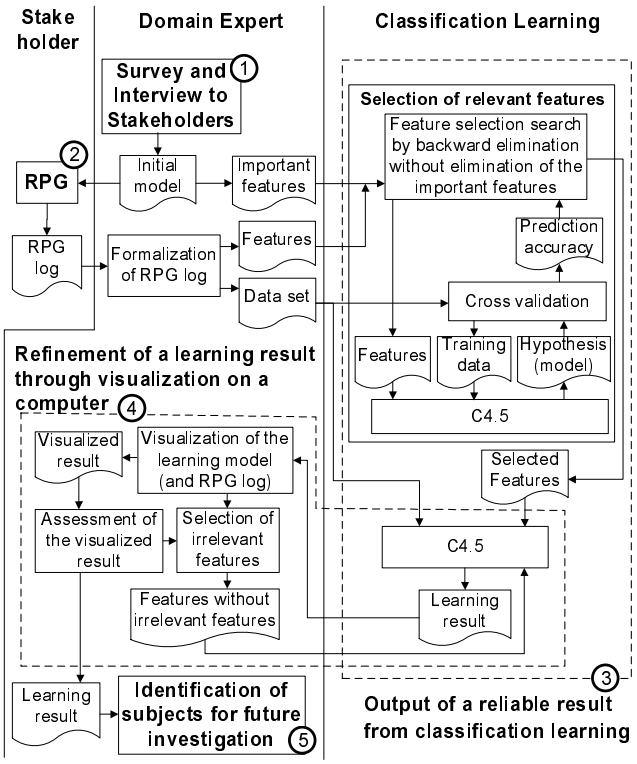


Figure 1: Modeling process for agents with classification learning

4.3 Modeling Process: a Real Example

We constructed five steps in which the approaches above are combined (Figure 1 shows the detail of this process); 1) Survey and Interview to Stakeholders, 2) RPG, 3) Output of a reliable result from classification learning, 4) Improving the result by expert knowledge and classification learning, 5) Identification of subjects for further investigation. This process will be explained with the results gained by applying it to a real example (see [15] for the detail explanation of the decision making model).

STEP1: Survey and Interview to Stakeholders

Domain experts create an initial model (decision tree) from relevant literature and interviews with stakeholders. Important features are extracted from the initial model. In our example, five features were extracted for the feature selection method (Figure 2).

[3]A method using a searching algorithm that identifies a feature subset which gives the best performance of a model outputted from an inductive learning. In many cases, cross validation is used to estimate the performance and hill-climbing is used to search a feature subset with the best estimated performance. In hill-climbing, there are two searching direction: one is backward elimination which begins at the full set of features and eliminates one feature which makes the estimated performance the best. Another is forward selection which begins at the empty set of features and adds one features which makes the estimated performance the best.

[4]Data sets are divided into n groups without overlapping. A machine learning program is applied to the n-1 groups and the result is tested by one unused group. The classification accuracy of the test data is used as criterion for the performance estimation. Switching the group used for the test data, this process is repeated n times. We assigned n to 10 which is the frequency generally used.

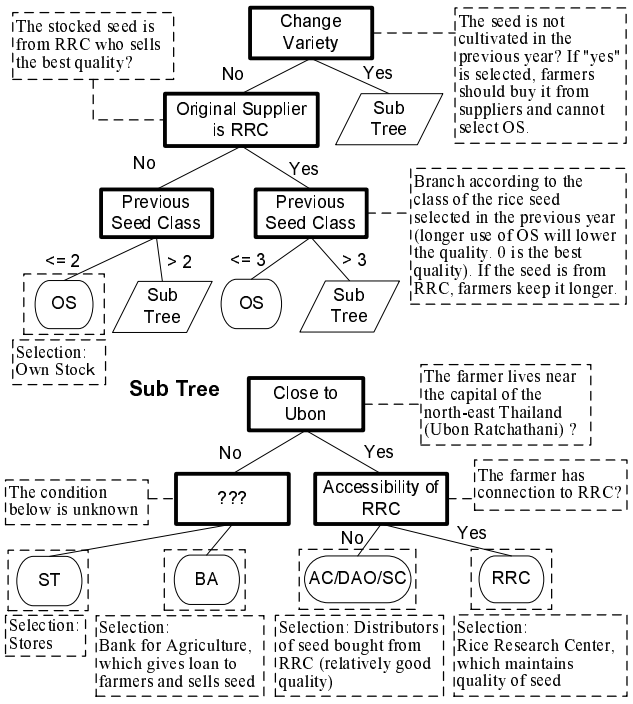


Figure 2: The initial decision tree model about farmers' selection of a rice supplier: The rectangles surrounded by bold lines are features. Five kinds of feature except "???" are the important one.

STEP2: RPG

RPG sessions using a board that represents the stakeholders' environment are organized. Log data for each stakeholder is recorded.

In our example, two RPG sessions were held. A board representing farm land was created (Figure 3 (a)). Totally, we got log data of 25 stakeholders.

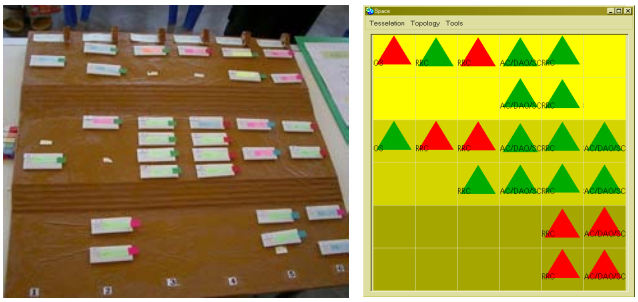
STEP3: Output of a reliable result from classification learning

Learning data is created by transforming the RPG log data into a format that the learning algorithm can understand. Next, irrelevant features are eliminated by the feature selection method (mentioned in section 4.2). In this process, important features identified in STEP1 are not eliminated. Finally, the machine learning outputs a model from the feature subset and the data set.

In our example, after formatting the log data, we had 16 features and 80 data sets. Next, the feature selection method was applied. Here, five of the features identified in STEP1 were not eliminated. Finally, the feature selection method eliminated 3 features and we got the first result from the remaining 13 features.

STEP4: Refinement of the learning result through visualization on a computer

The environment and participants are reproduced on a computer using a graphical interface and computer agents. In this evaluation, the experts are shown two types of results: results from the learning model and RPG data. After this,



(a) RPG board [17] (b) Visualized Interface

Figure 3: Interfaces for modeling: These are examples used in a case study of rice production in the northeast of Thailand. Both show upper, middle, and lower lands.

the model is directly shown to the experts. When there are irrelevant classification conditions in the model, the expert eliminates that feature which consists of the conditions and C4.5 will be applied with the resulting feature subset. This process is repeated until the expert is satisfied with the learning result.

We implemented the first learning model in CORMAS [2] (Figure 3 (b)) and evaluated the model by showing the results to the expert. At each step (equivalent to one year), the expert carefully observed the results and commented on the decision making of each agent.

After this, the first learning result was evaluated by directly showing the decision tree to the expert and an irrelevant condition was pointed by the expert (one feature was selected here and was eliminated from the feature set). The expert understood the reasons for the phenomena seen in the visualization. Classification learning was applied again with the reduced set of features. After applying the same processes in STEP4, the expert thought the learning result (Figure4) is logical to explain the decision making of the stakeholders in RPG and was satisfied with it, then we went to the next step.

STEP5: Identification of subjects for further investigation

By comparing the experts' hypothesis (Figure 2) with the learning result, the experts investigated their hypothesis and find subjects for investigating in the future. These will be discussed among the domain experts or investigated by repeating the process from STEP1.

In our example, the expert found that some parts supported his hypothesis: a) A condition "Close to Ubon = yes" leads a branch by "Accessibility of RRC", b) The bigger value of "Previous Seed Class" leads not *OS* but *RRC* or *AC/DAO/SC*. Meanwhile, other parts did not support it and the following subjects were identified: a) The value of "Previous Seed Class" (2 or 3 in the hypothesis while 0 in the result), b) The importance of two features which contains in the hypothesis but do not contain in the result ("Change Variety" and "Original Seed Supplier is RRC"). c) The possibility of the other important features (there should exist several features for the decision of *OS* except "Close to Ubon = No").

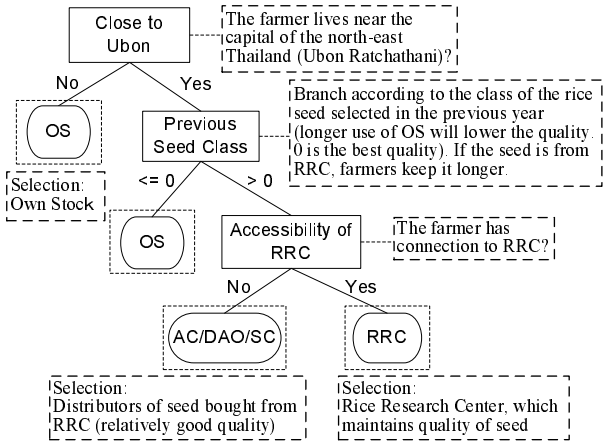


Figure 4: The final learning result

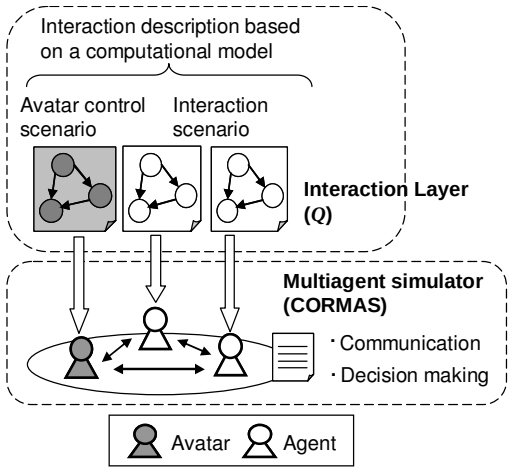


Figure 5: Agent Control from Interaction Layer

4.4 Evaluation

We can explain the effectiveness of our approach in the following way: first, the feature selection method reduced the cost of the expert refining the learning model. In our example, when the expert eliminates features without the feature selection method, it is necessary for the expert to iterate the refinement process three times, but when using the feature selection method, just one round of refinement was enough, because the feature selection method eliminated three features. In our approach, considering the preparation costs of the visualization method, it was desirable for the expert to reduce the refinement process. Moreover, we confirmed the effectiveness of our proposed method in which the features the expert thinks important are not eliminated in the feature selection method process. In our experiments, important features like these were often eliminated when this method was not used, and the experts could not explain the results as a decision making model of the stakeholders.

Second, we consider that using visualization to evaluate the learning model was useful for gaining a better understanding of the characteristics of the model and for facilitating the collaboration with the expert. The expert was able to make various comments about the decision making of agents who used the learning model because the interface had the same characteristics as the RPG. This is effective because it leads to more comments than when the learning model was shown directly to the expert, an approach that was tried in the early stage of our research.

5. MODELING INTERACTIONS WITH PARTICIPATORY SIMULATION

In this section, we will discuss the issues and the solutions in using participatory simulation to model interactions and explain the processes involved with an example: negotiation of seed suppliers. The negotiation model is represented in a state automaton and we implemented our participatory simulation in CORMAS/Q [16].

5.1 Issues and Approaches

Issues

For coaxing the actual behavior from stakeholders, it is important to design how the stakeholder controls an avatar and how agents interact with the avatar. Therefore, it is indispensable to give a development environment where domain experts are able to focus on designing interactions and the interaction design is directly used in system implementation. To this end, we developed a simulation system that uses the system architecture described in the next section.

Agent Control from Interaction Layer

We use a system architecture shown in Figure 5 where an interaction layer constructed outside of a simulation system controls the agents in the system[7][16]. This architecture makes it possible to use descriptions based on a computational model that makes it convenient to write interactions. Therefore, users can focus on interaction design.

As an actual simulation system based on this architecture, we combined the scenario description language Q [7] with the multiagent simulator CORMAS [2] which is often used in research based on the participatory method [3] (the system is called CORMAS/Q [16]). The computational model behind a Q scenario is an extended finite state automaton, which is commonly used for describing communication protocols. By using Q, users can directly create scenario descriptions from extended finite state automata. In Q scenarios, we can use sensing functions (cues) and action functions (actions). Scenarios are interpreted by Q interpreter, while cues and actions are executed by CORMAS. Cues/actions become the interface between domain experts and system developers, which enables collaboration in the design of interactions.

5.2 Interaction Description

For realizing participatory simulations where agents and avatars interact, it is necessary to assume the three kinds of interaction descriptions shown in Table 1. These descriptions are designed according to the modeling process shown in the next section. Interaction protocols are very important because they define the activity of avatars (stakeholders) in the simulation world. An interaction scenario regulates an agent's behavior about interactions (e.g. a negotiation

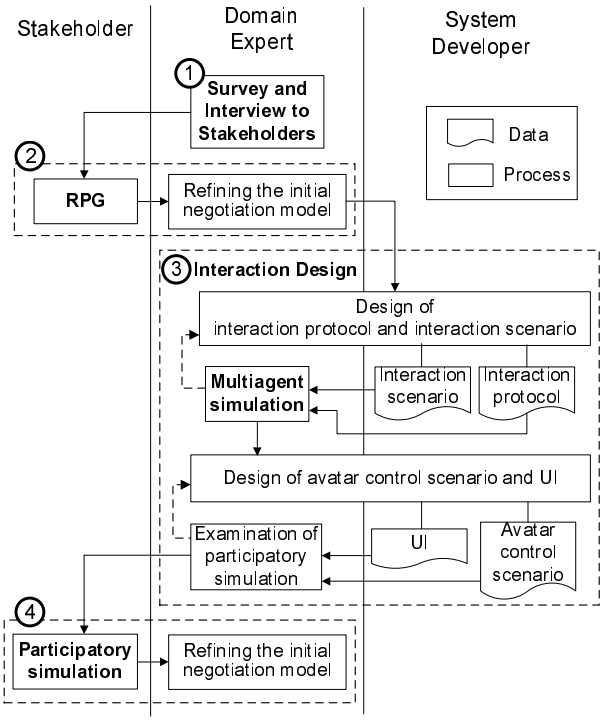


Figure 6: Modeling process for interactions with participatory simulation

model for agents who access the other agents and avatars). The avatar control scenario is an extended scenario of the interaction protocol. There are four elements in this scenario: 1) control of user interfaces (UI) (display etc.), 2) observation of input by a user through UI, 3) sensing from other agents, 4) action with other agents; this is triggered by input to the UI by a user.

Table 1: Interaction Descriptions

Classification	Definition of interaction description
Interaction protocol	Procedures which should be abided to negotiate with another agent or avatar
Interaction scenario (negotiation model)	Interaction procedures (e.g. access priority of trading or condition for completion of the deal) are described within the framework of the interaction protocol
Avatar control scenario	Control procedures of UI and an avatar along the interaction protocol

5.3 Modeling Process: a Real Example

The following four steps aim to refine an interaction model of a stakeholder and show a process to design three interaction descriptions shown in Table 1; 1) Survey and Interview to Stakeholders, 2) RPG, 3) Interaction Design, 4) Participatory Simulation (see Figure 6).

In this real example, we aimed to refine a negotiation model of *SC (Seed Center)* who is one of the important suppliers in northeastern Thailand. There are four types

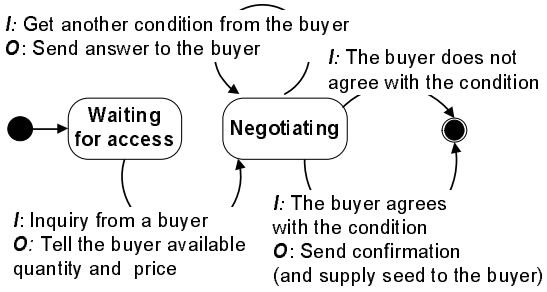


Figure 7: Example of interaction protocol (for sellers): Conditions are exchanged between the seller and the buyer in enough times to agree and the negotiation is settled if the buyer agrees/disagrees with the condition.

of supplier except SC and they are realized as agents in the participatory simulation (they are shown as A, B, C and D for descriptive purpose.).

STEP1: Survey and Interview to Stakeholders

Domain experts create an initial model from relevant literature and interviews to stakeholders. (In our example, the initial model was very abstract which just specified a contact list of each supplier).

STEP2: RPG

RPG based on STEP1 is held with stakeholders. In the process, RPG is very important because RPG is a good place for communication between domain experts and stakeholders unfamiliar with the research process and sometimes unfamiliar with computer systems. Through RPG, behavioral model and the reason behind the behavior are exposed. The initial model is improved by knowledge acquired through RPG.

The refined model of SC in our example is that rice seed is supplied according to a priority order (A, B, C and D)

STEP3: Interaction Design

An interaction protocol and interaction scenarios are described based on observation in RPG. What is important here is that several kinds of interaction scenarios are created for each agent to observe reaction of a stakeholder in participatory simulation. These descriptions are designed by a computational model of the interaction description through collaboration of a domain expert and a system developer. Finally, these descriptions are implemented in a computational language and tested through several runs of multiagent simulation. The processes of design and simulation are repeated until a domain expert is satisfied.

Next, UI and an avatar control scenario are designed by the domain expert and system developer for participatory simulation. In the avatar control scenario, the control procedures of UI are considered based on the interaction protocol. Next, UI and the avatar control scenario are implemented by the system developer and the participatory simulation system is tested by the domain expert. The processes above are repeated until the domain expert is satisfied.

In our example, through discussion between a domain expert and a system developer, an interaction protocol and

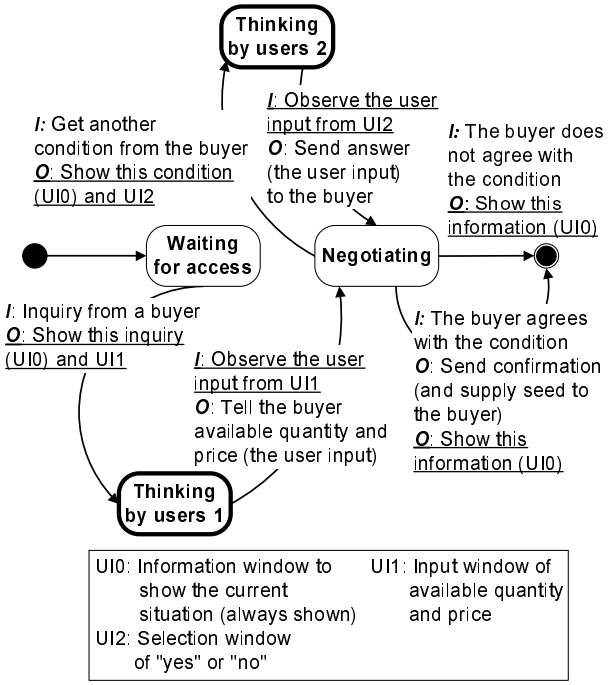


Figure 8: Example of avatar control scenario (for *Seed Center*): the interaction protocol shown in figure 7 is extended by putting controls concerning to UI (states surrounded by a bold line and conditions/actions with underlines)

interaction scenarios for each agent were designed by a state transition model which is a computational model of Q. The interaction protocol is shown in Figure 7. Three kinds of interaction scenarios are also prepared: a) access in a fixed order, b) access in a cheap price order and c) access in a short distance order. Next, the domain expert and system developer defined cues/actions of Q and the system developer implemented corresponding behavior of the cues/actions in CORMAS (this part of design process are described in [11]).

UI for our participatory simulation was not one big panel with all functions but small panels with only necessary functions at each scene, which are opened and closed according to the avatar control scenario. As in the interaction scenario, the avatar control scenario was designed by state transition diagram (Figure 8). Then, cues/actions are designed and corresponded to the control of UI or avatar behavior in CORMAS.

STEP4: Participatory Simulation

Participatory simulation with stakeholder is held. The domain expert refines the interaction scenario by analyzing the log data and knowledge acquired through conversation with the stakeholder.

In our example, the refining process of the model is as follows. The result after several times of participatory simulation was that supplier B was always selected first then the selection of A, C and D was not fixed. This was observed every time, though we changed the interaction scenario given to the agents in every run of simulation. After the domain expert recognizes this phenomenon, he tried one

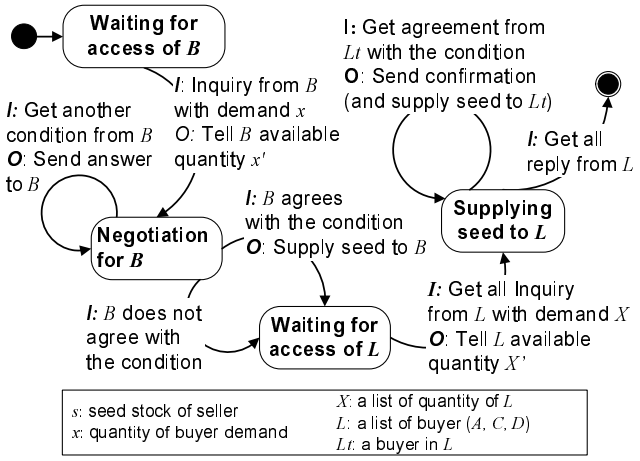


Figure 9: Refined negotiation model of *Seed Center (SC)* after Participatory Simulation: B is first chosen then A, C and D are equally distributed. If the stock remains, the other buyers will be supplied.

more around of simulation while he gave questions to the stakeholder. From the conversation with the stakeholder, the domain expert could refine the model as shown in Figure 9.

5.4 Discussion

We confirmed the merits to use participatory simulation. 1) The domain expert could focus on observing specific stakeholders (only SC in our example) because agents took roles instead of the stakeholders in RPG and a computer program automatically conducts the simulation. 2) The domain expert could give questions according to the progress of simulation. The stakeholder could monitor the simulation progress and see the impact of his own decision, which enabled him to give more concrete answer.

We consider that this participatory simulation operated as intended. This is because the system architecture could give the environment for the domain expert to focus on interaction descriptions and our proposed processes of designing three kinds of interaction descriptions were effective to reflect his knowledge acquired through surveys and RPG to the simulation world.

6. CONCLUSION

In this study, we proposed a novel methodology of modeling agents (decision making) and interactions (negotiation) in multiagent simulations for consensus building among stakeholders. This methodology combines several techniques with the participatory method, which takes stakeholders into the modeling process. We tackled the following two problems.

Modeling agents with classification learning

A modeling method wherein classification learning is applied to RPG log data was established for validation of domain experts' hypothesis (agent's internal model). The key ideas are 1) a feature selection method for enhancing reliability of the learning result by classification learning, and 2) visualization of the learning result on a computer to promote

understanding of domain experts in refining the learning result. The modeling process with these methods was applied to farmers' decision model of seed suppliers. As a result, the domain expert's hypothesis was validated and several subjects for further investigation were found.

Modeling interactions with participatory simulation

A modeling method with participatory simulation was established for deep understanding of stakeholders' interactions (e.g. negotiation). The key ideas are 1) a system architecture where interaction layer controls agents gives an environment for the domain expert to focus on interaction descriptions, and 2) three kinds of interaction descriptions (interaction protocol, interaction scenario, avatar control scenario) are defined and designed through the modeling process. Actually, a participatory simulation was developed on CORMAS/Q [16] and the domain expert could get deeper understanding of a negotiation model among seed suppliers.

Our modeling methodology has the following two important features for multiagent simulations to build consensus among stakeholders. First, RPG and participatory simulation give shared experience of modeling with domain experts, which is important for stakeholders to find their solutions in the simulation results. Second, the modeling method with classification learning and participatory simulation enables domain experts to deeply understand and model stakeholders' actual behavior in the system. Indeed, the impact was confirmed in the real case study of agricultural economics in the northeast of Thailand [17].

In this study, we could use a decision tree and a state automaton to represent decision making and negotiation, respectively. In the other cases, however, we might find or develop suitable models, the learning algorithms, systems providing the interaction description. This will be the future works, which also give new seeds of artificial intelligence and multiagent research. Another future work is use of new technologies for user participation. For example, *Augmented experiment* [8] can give an environment where stakeholders take part in the simulation world from their real work space, which enables domain experts to understand more actual behavior of stakeholders.

7. ACKNOWLEDGMENTS

This research was supported by the young researcher development program of the JSPS 21st century COE program "Informatics Research Center for Development of Knowledge Society Infrastructure". We would like to thank Dr. Guy Trébuil and Dr. Chirawwat Vejpas who gave valuable comments as a domain expert, and farmers and seed suppliers who joined our experiment in Ubon Ratchathani.

8. REFERENCES

[1] M. Balmer, N. Cetin, K. Nagel, and B. Raney. Towards truly agent-based traffic and mobility simulations. In *Proc. Int. Joint Conf. Autonomous Agents and Multiagent Systems (AAMAS-04)*, pages 60–67, 2004.

[2] F. Bousquet, I. Bakam, H. Proton, and C. L. Page. Cormas: common-pool resources and multi-agent systems. volume 1416 of *LNAI*, pages 826–838. Springer-Verlag, 1998.

[3] F. Bousquet, O. Barreteau, P. d'Aquino, M. Etienne, S. Boissau, S. Aubert, C. L. Page, D. Babin, and J. Castella. Multi-agent systems and role games: an approach for ecosystem co-management. In *Complexity and Ecosystem Management: The Theory and Practice of Multi-agent Approaches*, pages 248–285. Edward Elgar Publishers, 2002.

[4] N. Gilbert, S. Maltby, and T. Asakawa. Participatory simulations for developing scenarios in environmental resource management. In *Proc. Third Workshop on Agent-Based Simulation*, pages 67–72, 2002.

[5] P. Guyot, A. Drogoul, and C. Lemaitre. Using emergence in participatory simulations to design multiagent systems. In *Proc. Int. Joint Conf. Autonomous Agents and Multiagent Systems (AAMAS-05)*, pages 199–203, 2005.

[6] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, 2000.

[7] T. Ishida. *Q*: A scenario description language for interactive agents. *IEEE Computer*, 35(10):42–47, 2002.

[8] T. Ishida. Multiagent simulation meets the real world. In *Proc. Int. Joint Conf. Autonomous Agents and Multiagent Systems (AAMAS-06)*, 2006.

[9] H. Kita, H. Sato, N. Mori, and I. Ono. U-mart system, software for open experiments of artificial market. In *Proc. of IEEE Int. Symposium on Computational Intelligence in Robotics and Automation*, pages 1328–1333, 2003.

[10] R. Kohavi and G. H. John. Wrappers for feature selection. *Artificial Intelligence*, 97:273–324, 1997.

[11] Y. Murakami, T. Ishida, T. Kawasoe, and R. Hishiyama. Scenario description for multi-agent simulation. In *Proc. Int. Joint Conf. Autonomous Agents and Multiagent Systems (AAMAS-03)*, pages 369–376, 2003.

[12] Y. Murakami, Y. Sugimoto, and T. Ishida. Modeling human behavior for virtual training systems. In *Proc. National Conf. Artificial Intelligence (AAAI-05)*, pages 127–132, 2005.

[13] J. R. Quinlan. Induction for decision trees. *Machine Learning*, 1(1):81–106, 1986.

[14] L. B. Said, T. Bouron, and A. Drogoul. Agent-based interaction analysis of consumer behavior. In *Proc. Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS-02)*, pages 184–190, 2002.

[15] D. Torii, F. Bousquet, T. Ishida, G. Trébuil, and C. Vejpas. Using classification learning in companion modeling. In *Proc. of the 8th Pacific Rim Int. Workshop on Multi-Agents (PRIMA-05)*, pages 327–341, 2005.

[16] D. Torii, T. Ishida, S. Bonneaud, and A. Drogoul. Layering social interaction scenarios on environmental simulation. In *Multi-Agent and Multi-Agent-Based Simulation*, volume 3415 of *LNCS*, pages 78–88. Springer-Verlag, 2005.

[17] C. Vejpas, F. Bousquet, W. Naivinit, G. Trébuil, and N. Srisombat. Participatory modeling for managing rainfed lowland rice variety and seed systems in lower northeast thailand. In *Mekong Rice Conference*, 2004.

Learnable Behavioural Model for Autonomous Virtual Agents: Low-Level Learning

Toni Conde
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Virtual Reality Lab (VRLab)
CH-1015 Lausanne, Switzerland
toni.conde@epfl.ch

Daniel Thalmann
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Virtual Reality Lab (VRLab)
CH-1015 Lausanne, Switzerland
daniel.thalmann@epfl.ch

ABSTRACT
In this paper, we propose a new integration approach for simulation and behaviour in the learning context that is able to coherently manage the shared virtual environment for the simulation of autonomous virtual agents. Our low-level learning technique has proved fast, simple and robust. It is also able to automatically learn behavioural models for difficult tasks. Thus, we believe it will be more useful to the computer graphics community than a technique based on the classical Q-learning approach. The results are illustrated in two case studies that require effective coordination of behaviours.

Categories and Subject Descriptors
I.2.6 [**Artificial Intelligence**]: Learning; I.3.6 [**Computer Graphics**]: Methodology and Techniques – *Interaction techniques*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism - *Animation*.

General Terms
Algorithms, Human Factors, Performance.

Keywords
Computer animation, character animation, reinforcement learning, behavioural modelling, AI-based animation, machine learning.

1. INTRODUCTION
Autonomous Virtual Agents (AVAs) play an important role in computer graphics. Animating them is one of the fundamental challenges.

In this paper we are presenting research in the domain of behavioural animation using a low-level learning technique. This is accomplished with a behavioural model defining how an AVA reacts to stimuli from its environment.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8–12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005…$5.00.

Human-like behaviour is achieved through the use of a complex "cognitive map" and the application of a hierarchy of behavioural strategies [1]. The overall cognitive mapping process involves acquisition, coding, storage, recall and decoding of environmental information [2]. In fact, an individual "cognitive map" will often contain numerous inaccuracies or distortions [3]. Many of these anomalies result from the fact that humans use a predominantly visual perception system and are not capable of processing everything they see due to the tremendous amount of incoming information. Other errors result from the way the information is processed and stored within the "cognitive map" structure itself [4].

Previously, [5] have proposed new methodologies to carry out the mapping of all information coming from virtual sensors of vision, audition and touch as well as from the Virtual Environment (VE) in the form of a "cognitive map". The approach enables the partial re-mapping of cognitive and semantic information at a behavioural level. For example, when spatial attention is primed with tactile stimulation, the location of the attention spotlight is only partially remapped in visual coordinates. This framework will generate the multi-sensory information necessary for our *behavioural learning*.

In summary, this paper presents an original contribution with a novel approach allowing an AVA to independently learn a behavioural model.

This paper is organized as follows: Section 2 – State of the Art; Section 3 – Methodology; Section 4 - Integration; Section 5 – Experimental Results with two Case Studies; Section 6 – Discussion.

2. STATE OF THE ART
A great deal of research has been performed on the control of animated autonomous characters by [6], [7], [8] and [9]. The various techniques have produced impressive results, but they are limited. They have no learning ability and are therefore limited to explicit pre-specified behaviours.

On-line *behavioural learning* has begun to be explored in computer graphics [10] and [11]. A notable example is in [12], where a virtual dog can be interactively taught by the user to

exhibit a desired behaviour. This technique is based on *Reinforcement Learning* (RL) and it has been shown to work well in [13]. Also, since all these learning techniques are designed to be used on-line, they have, for the sake of interactive speed, a limited learning capacity.

In RL, the machine learning of an input-output mapping is performed through continued interaction with an environment in order to maximize a scalar index of performance. This performance index is called a fitness function. Some of the earliest research in computer graphics involving RL sought to make AVAs automatically learn how to walk, swim, or jump [14] and [15]. However, whilst interesting and useful, this type of learning does not provide AVAs with decision-making abilities. On-line *behavioural* (reactionary) *learning* in computer graphics is a new field of exploration.

The goal of RL is to automatically learn an optimal policy. Optimal implies that the policy always maps the current state to the best possible action according to the fitness function. Since it is impractical to store a large Q-factor table explicitly, it must be approximated. We have performed several experiments using *Q-learning* for AVA learning, where the only information available to an AVA was a fitness function. Previously, [13] have demonstrated the difficulty of this approach. Firstly, to obtain stable results, the Q-factor table must be approximated to a high degree of accuracy. Secondly, *Q-learning* is harder to perform if there are no terminal states. Also, as discussed in [16], *Q-learning* can be challenging to successfully use, especially since it requires the animator to visit every state-action many times.

The technique presented in this paper is built on the success of traditional behavioural modelling with the goal of alleviating two of its major weaknesses: performance and time-consuming construction. In our opinion, the difficulties associated with *Q-learning* make it an undesirable approach for our application. For this reason, we have developed an alternative technique for AVA low-level learning. The objective is to allow an AVA to explore its unknown VE and to build structures, in the form of cognitive models or maps, based on this exploration. Once its representation has been constructed, the AVA could then easily communicate its knowledge to, for example, other naive AVAs.

3. METHODOLOGY

In this section we introduce our low-level learning methodology. This means that an AVA can automatically learn a behavioural model. We have developed a new technique to perform AVA learning, using a tree search with *Inverse Reinforcement Learning* (IRL).

3.1 Planning-based Reinforcement Learning (RL) with Subagents

Given the shortcomings of *Q-learning* discussed above, we have developed an alternative approach to AVA low-level learning by computing discrete examples of a policy.

A planning-based RL technique will only learn a sub-optimal policy, the quality of which depends on the search depth limit used [17] and [18]. To ensure that on the whole an AVA's behaviour is optimal, we utilize the *Q-decomposition* approach proposed by [19], where only the sum of the subagent *Q-values* for a particular state determines the optimal action. This requires each subagent to indicate, from its perspective, a value for every action. Subagent *j* reports its action values $Q_j(s, a)$ for the current state *s* to the *arbitror*. Then, the *arbitror* chooses an action maximizing the sum of the *Q-values* (see Figure 1). In our proposed methodology we use *pseudo values* – instead of *Q-values* – for vision (Pv_{vision}), avoidance ($Pv_{avoidance}$) and navigation ($Pv_{navigation}$) to trade-off different features of virtual sensors.

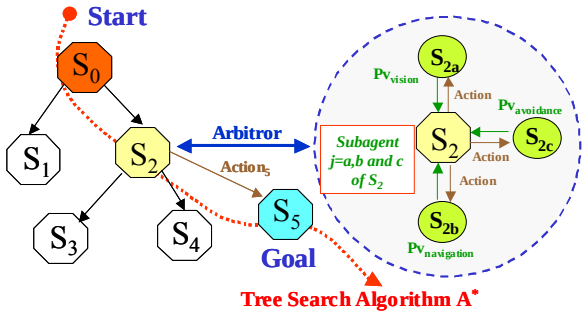


Figure 1: (left) Planning is performed with a tree search algorithm A* of all proposed actions throughout the time limit. (right) The *arbitror* chooses an action maximizing the sum of the *pseudo values* ($Action_5$).

3.2 Apprenticeship Learning via Inverse Reinforcement Learning (IRL)

It may be impossible to use straightforward RL since an AVA may only have an approximate idea of the reward function which, when optimised, should generate a "desirable" behaviour. We believe that the difficulty of manually specifying a reward function represents a barrier to the broader applicability of RL and the optimal control of the algorithms.

As the entire field of RL is founded on the supposition that the reward function, rather than the policy or the value function, constitutes the most succinct, robust and transferable definition of the task, it seems natural to consider an apprenticeship learning approach via IRL [20] to learn the reward function.

In the learning process, one source of information consists of observing the behaviour of other "expert" AVAs as in *imitation* and *apprenticeship learning*. In this context, it is commonly assumed that the purpose of observation is to learn a policy which is represented by the mapping of states to actions.

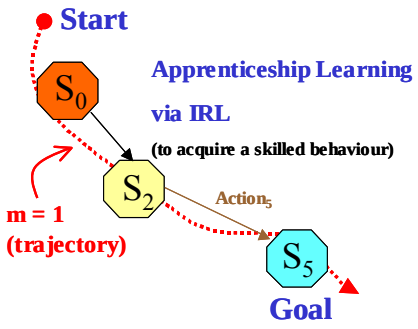


Figure 2: *Apprenticeship learning* via IRL is performed after *m* trajectories generated by the expert.

In our methodology, IRL is used in *apprenticeship learning* to acquire a skilled behaviour and to ascertain the optimisation of a reward function through a natural system. Yet, in using IRL, it is difficult to extract an observed reward function [21]. This is a fundamental challenge in theoretical biology, econometrics and other fields.

Using this method, an AVA can automatically learn a behavioural model. For the animator, this alleviates the workload of programming an explicit behavioural model (see Figure 2).

4. INTEGRATION

The integration of our low-level learning technique combined with different machine-learning techniques – but with several greatly improved – will be more useful to the computer graphics community than techniques based on a purely machine-learning approach (see Figure 3).

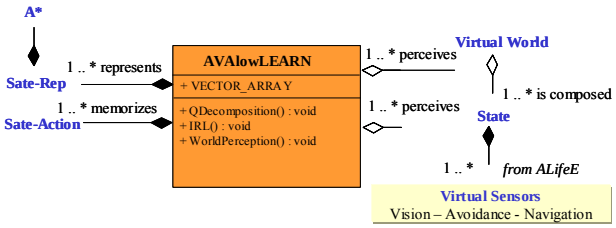


Figure 3: Comprehensive UML design of AVA low-level Learning (AVAlowLEARN). (left) State and action space. (right) Multi-sensory information necessary for our *behavioural learning*.

Firstly, we used a tree search algorithm A* [22] to observe the trajectories (state sequences) generated by the user (expert) and which take actions according to π_E (π_{expert}).

Secondly, we integrated the same *Q-decomposition* approach as [19], which uses all *pseudo value* function components (vision, avoidance and navigation) to choose the actions. *Q-decomposition* allows for multiple value functions, with each of these residing in a subagent, but it still requires each subagent to report its value estimates to an *arbitror* and to receive in turn the action chosen by the *arbitror*.

Thirdly, we took the *apprenticeship learning* via IRL with the projection algorithm proposed by [20] and adapted it to our behavioural animation.

A policy π is a mapping from states to probability distributions over actions. Given that the reward R is expressible as a linear combination of the features ϕ, the expectation values of the features for a given policy π completely determine the expected sum of discounted rewards for acting according to that policy (see Figure 4).

Given a MDP\R (denotes a Markov Decision Process without a reward function), a feature mapping ϕ and the expert's feature expectations μ_E, the problem is to find a policy whose performance is close to that of the expert's, on the unknown reward function $R^* = \omega^*.\phi$.

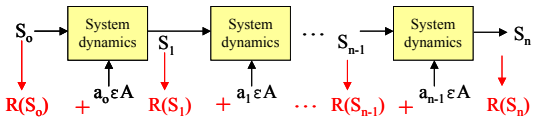


Figure 4: Reinforcement Learning (RL) description with a finite-state Markov Decision Process (MDP).

The *apprenticeship learning* projection algorithm proposed by [20] for finding a policy is as follows (pseudo-code):

1. Randomly we pick some policy $\pi^{(0)}$, $\mu^{(0)} = \mu(\pi^{(0)})$, and set i = 1.

   $\mu(\pi) = E[\sum_{t=0}^{\infty} \gamma^t . \phi(s_t) \mid \pi] \varepsilon \Re^k$

   ($\mu(\pi)$ is the expected discounted accumulated features value vector)

2. Set $\bar{\mu}^{(i-1)} =$

   $$\bar{\mu}^{(i-2)} + \frac{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu_E - \bar{\mu}^{(i-2)})}{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu^{(i-1)} - \bar{\mu}^{(i-2)})} (\mu^{(i-1)} - \bar{\mu}^{(i-2)})$$

   (This computes the orthogonal projection of μ_E onto the line through $\mu^{(i-2)}$ and $\mu^{(i-1)}$).

   Set $\omega^{(i)} = \mu_E - \bar{\mu}^{(i-1)}$

   Set $t^{(i)} = \left\| \mu_E - \bar{\mu}^{(i-1)} \right\|_2$

3. If $t^{(i)} \leq \varepsilon$, then terminate.
4. Using the RL algorithm, compute the optimal policy $\pi^{(i)}$ for the MDP using rewards $R = (\omega^{(i)})^T \phi$.
5. Estimate $\mu^{(i)} = \mu(\pi^{(i)})$.
6. Set i = i + 1, and go back to step 2.

Upon termination, the algorithm returns $\{\pi^{(i)}: i = 0 \ldots n\}$.

The performance of this algorithm only depends on (approximately) matching the feature expectations, not on recovering the true underlying reward function.

5. EXPERIMENTAL RESULTS WITH TWO CASE STUDIES

To test whether our low-level learning model was qualitatively able to learn the demonstrated AVA's behaviour, we created two environments requiring effective coordination of behaviours: a virtual apartment and a more complex situation with a car driving simulation inside a virtual city.

5.1 Case Study No. 1: an AVA learning to avoid obstacles and to navigate inside a virtual apartment

We chose to simulate an AVA in an apartment where he can "live" autonomously by perceiving his VE and generating a few behaviours at specific locations.

We can test our *behavioural model* applied to an AVA by implementing the model in C++ and Python [23] thanks to the Python module of the real time framework [24] for advanced virtual human simulations.

As depicted in Figure 5, an interface has been designed to monitor and change, if necessary, the evolution of the AVA's states and actions in real-time inside the virtual apartment. The path planning module [25] is applied for obstacle avoidance when the AVA walks to a specific location (see Figure 6). It is also utilized to the decisions the AVA makes. Its 3D viewer shows, in real-time, what the AVA decides to do at each moment and can play keyframes for the learned actions using the walk engine [26].



Figure 5: Top view of the virtual apartment.



Figure 6: (upper left corner) Top view of the virtual apartment with path planning after low-level learning process. The AVA is now inside the kitchen.

Figure 7 shows the inputs and outputs of the behavioural model for the navigation inside a virtual apartment. The input for Vision (r, φ) is the spherical coordinates of the AVA's head. Our Artificial Life Environment (ALifeE) [5] gives multi-sensory information to the AVA's sensors of vision, audition and touch (see Figure 3).

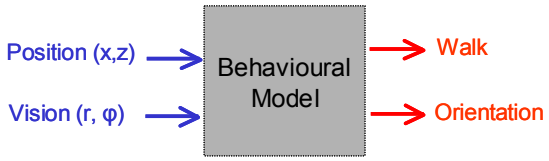


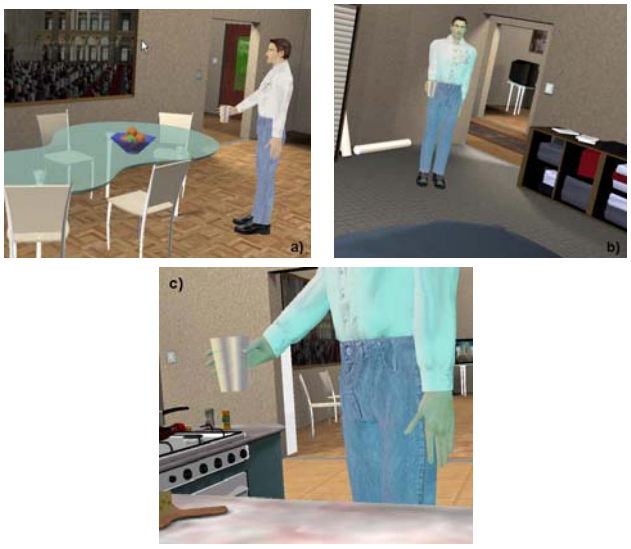
Figure 7: Behavioural model for the navigation inside a virtual apartment with inputs and outputs. The MDP has two different actions: walk and orientation.

In the first scenario (see Figures 8 a), b) and c)), an AVA learns the behaviour of taking a *CD box* from the kitchen table. Then, the AVA walks around inside the different rooms whilst taking care to observe and avoid virtual objects and also navigates with a path planning between doors. Finally, the AVA puts the *CD box* on the desk in the office.



Figures 8 a), b) and c): Snapshots of the AVA learning (through seeing, avoiding and navigating) to take a *CD box* from the kitchen table and to put in on the office desk.

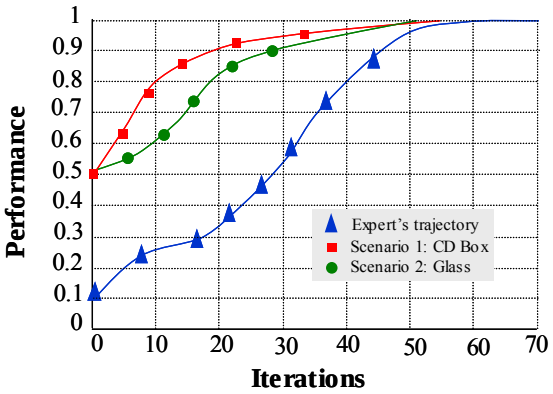
In the second scenario (see Figures 9 a), b) and c)), an AVA learns to take a *glass* from the living room table or from the bedroom shelf and to bring it to the kitchen.



Figures 9 a), b) and c): Snapshots of the AVA learning (through seeing, avoiding and navigating) to take a *glass* from the living room table and to bring it to the kitchen.

The two scenarios were reported in Table 1 to illustrate the performance of our low-level learning methodology. Using our methodology, only a few sample trajectories were needed to attain a performance mimicking that of the expert. Thus, by learning a compact representation of the reward function, our methodology outperformed.

Table 1. Plot of performances obtained with our low-level learning methodology. All animations were rendered in real time using OpenGL on a 3.0 GHz PC with an nVIDIA GeForce FX Go5350 video card.



5.2 Case Study No. 2: an AVA learning to drive a car inside a virtual city

We implemented our approach to learn the behavioural model applied to a car driving simulation inside a virtual city. An AVA is the driver and has dual control over the acceleration and the wheel of the car (see Figure 10). The controls are real-value (e.g., the action space is continuous) and the car can move to any location or take any orientation.

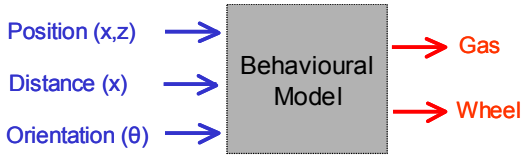


Figure 10: Behavioural model for the car driving simulation inside a virtual city with inputs and outputs. The MDP has two different actions: gas and wheel.

Figure 10 shows the inputs and outputs of the behavioural model for the car driving simulation inside a virtual city. The input for Distance (x) is the distance between the driver and the traffic signal.

The continuous action space was quantified to achieve real-time performance. The simulation ran at 15 [frames/s] and the expert's features were estimated from a single trajectory of 2'700 samples, corresponding to 3 minutes of driving time.

Our approach communicates the semantic information required for the "right way" behaviour of our low-level learning method in two ways:

- adding semantics to dynamic objects, and
- adding tools to sound, pedestrians and semantic objects (e.g., road signals).

Adding Semantics to Dynamic Objects

The appropriate description of tools for autonomous cars and pedestrians is available inside our simulation application. We use scripts to configure the VE application and to determine the behaviour of the autonomous cars in each different situation. The behaviour of the autonomous cars is controlled with Python scripts [23]. To simulate the life of a virtual city in a realistic manner, the VE also integrates some semantic notions about specific areas.

Adding Tools to Sound, Pedestrians and Semantic Objects

We have defined three types of plug-ins:

1. The plug-in for sound permits the definition of the properties of the sound environment in 3Ds max® (3D render software). To achieve a realistic illusion, it is important to generate a 3D spatialization in sound. The geometry of each object is transformed into a list of points outlining the sound obstacle. We define four types of sound primitives: environment, obstacle, listener and sound source. Inside the VE, we can have several sound sources with the following essential parameters: position, orientation (e.g., the direction of propagation) and ray configuration of the two cones of propagation. A sound source can be attenuated, absorbed or reflected depending on when the sound strikes an obstacle. The

obstacles are characterized by a two dimensional contour.

2. The plug-in for pedestrians covers a crowd composed of two types of people: those who walk in the virtual city with a precise goal and those who stroll.
3. The plug-in for semantic objects, inspired by the smart objects concept [27], gives significance to the road signals and traffic lights so that the sensory module of vision [5] can recognize them.

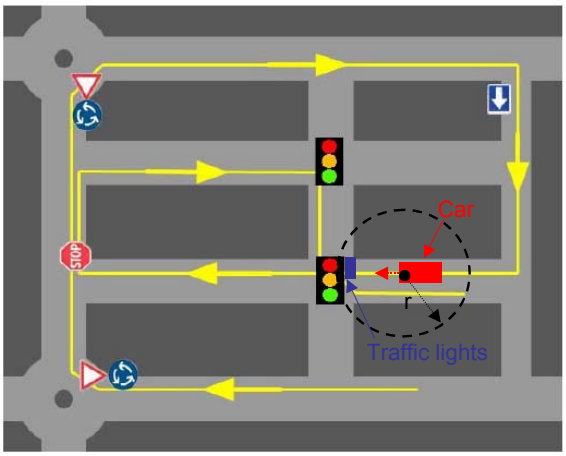


Figure 11: AVA learning to drive a car inside a virtual city with visual pseudo-perception.

Using our plug-ins along with 3Ds max®, designers have the information necessary to construct all types of VEs. We are now able to create the semantic information for our simulation but it will have a static character. However, we wish to obtain dynamic semantic information and we will use pseudo-perception (see Figure 11) or virtual perception [5] to achieve this goal.

In our car driving simulation, ϕ is the vector of the features indicating the different driving desiderata that need to be accounted for, such as a collision with another car, a pedestrian or a virtual object, or driving on the sidewalk, for example. The unknown vector ω^* specifies the relative weight of these various desiderata.

Blindly following the expert's trajectory would not work, because the pattern of traffic encountered is different each time.

We applied *apprenticeship learning* to try to learn behaviours to avoid pedestrians and to take traffic lights into account (e.g., stop at a red light, decelerate at an orange light and keep driving at a green light).

The final result was a good driving behaviour, since the driver could plan far enough ahead to manoeuvre the car adequately around the virtual city, avoiding pedestrians and reacting correctly to traffic lights (see Figures 12 a) and b)).



Figure 12 a): The car "sees" the traffic lights and the pedestrian inside a circular zone. The panel informs the driver that he/she must brake (B – in red at bottom right corner).



Figure 12 b): A pedestrian wishes to cross the road according to the rules defined.

We achieved our best results by performing the low-level learning method for 40 iterations and a policy was selected by inspection (see Table 2).

Table 2. Results obtained with our low-level learning methodology and one driving style (as estimated from 3 minutes of driver demonstration). All animations were rendered in real time using OpenGL on a 3.0 GHz PC with an nVIDIA GeForce FX Go5350 video card.

	Gas	Wheel
μ_e	0.060	0.290
$\mu(\pi)$	0.059	0.266
ω	0.107	0.059

Our method was qualitatively able to mimic the demonstrated driving style. Table 2 shows the feature expectations of the expert (driver) μ_e, the feature expectations learned $\mu(\pi)$ and the weights ω used to generate the policy.

Although our technique performed well in our two case studies, there is no guarantee that it will work for every imaginable AVA and simulated VE.

6. DISCUSSION

Our low-level learning technique has proved to be fast, simple and robust. It has also succeeded in automatically learning behavioural models for difficult tasks. Thus, we believe it will be more useful to the computer graphics community than a technique based on the classical *Q-learning* approach. Unlike explicit *Q-learning*, our technique is not guaranteed to find an optimal policy. But *Q-learning* is usually impractical because of a large Q-factor table

Our AVA low-level learning technique will only learn a sub-optimal policy, whose quality depends on the selected search depth limit. However, optimality is probably not necessary to achieve the appearance of intelligent behaviour in an AVA. Indeed, good behaviour usually looks just as, if not more, realistic than perfect behaviour

With our method, an AVA can automatically learn a behavioural model. For the animator, this relieves the workload of designing an explicit model. Moreover, it allows to model tasks for which it would be difficult or virtually impossible, to develop an explicit model.

Our work needs to be distinguished from that of [28]. In fact, when they performed off-line AVA learning, it could be difficult to design a fitness function that resulted exactly in the desired behavioural model. Our contribution in this paper is important because we present a solution for this previously unsolved problem and also we perform on-line AVA low-level learning.

In the robotics field, various methods like *teaching by imitation*, *imitation learning* or *learning by observation* are effective. However, since they are not automatic, they cannot be used for on-line adaptation. The technique presented by [29] for teaching a robot simple navigational behaviours is too limited for AVA behavioural learning. By comparison, our mimicking method is less general, but is fully automatic, learns quickly and can easily encapsulate very complex behaviours. One of its most original aspects is indeed to detect automatically novel behaviours.

However, there are some weaknesses in our approach. When performing AVA low-level learning, it can be difficult to find a policy that exactly corresponds to the desired behavioural model. Also, the reward function should be expressible as a linear combination of known features.

The methodology behind automatic learning behaviour is difficult but interesting. This could take interactive computer graphics, especially in the entertainment market, to a completely new level. It could also be interesting if an animator could interactively train an AVA for *behavioural learning*, rather than using a reward value.

The approach presented here is part of a more complex model that is the object of our research. The goal is to realize an Artificial Life environment for an AVA including different interfaces and sensorial modalities coupled with various evolving learning methodologies.

7. ACKNOWLEDGMENTS

This research has been partially funded by the Swiss National Science Foundation.

8. REFERENCES

[1] P. Larkin. Achieving human style navigation for synthetic characters. A survey. In *Proceedings of Neural Networks and Computational Intelligence*, pages 30-37, 2003.

[2] R. Downs and D. Stea. Cognitive maps and spatial behavior. *Image and Environment*. R. Downs and D. Stea, Eds., pages 8-26, Chicago: Adline Publishing, 1973.

[3] D. Griffin. Topographical Orientation. *Image and Environment*. R. Downs and D. Stea, Eds., pages 296-299, Chicago: Adline Publishing, 1973.

[4] N. Kamwisher and P. Downing. Separating the wheat from the chaff. *Science*, vol. 282, pages 57-58, 1998.

[5] T. Conde and D. Thalmann. An Artificial Life Environment for Autonomous Virtual Agents with multi-sensorial and multi-perceptive features. *Computer Animation and Virtual Worlds*, **15**(3-4), pages 311-318, John Wiley, 2004.

[6] C. Reynolds. Flocks, herds, and schools: A distributed behavioural model. In *Proceedings of ACM SIGGRAPH*, pages 25-34, 1987.

[7] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behaviour. In *Proceedings of ACM SIGGRAPH*, pages 43-50, 1994.

[8] B. Blumberg and T. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. In *Proceedings of ACM SIGGRAPH*, pages 47-54, 1996.

[9] K. Perlin and A. Golberg. A improv: a system for scripting interactive actors in virtual worlds. In *Proceedings of ACM SIGGRAPH*, pages 205-216, 1996.

[10] R. Burke, D. Isla, M. Downie, Y. Ivanov and B. Blumberg. Creature smarts: The art and architecture of a virtual brain. In *Proceedings of the Computer Game Developers Conference*, pages 147-166, 2001.

[11] B. Tomlinson and B. Blumberg. Alphawolf: Social learning, emotion and development in autonomous virtual agents. In *Proceedings of First GSFC/JPL Workshop on Radical Agent Concepts*, pages 35-45, 2002.

[12] B. Blumberg, M. Downie, Y. Ivanov, M. Berlin, M. Johnson and B. Tomlinson. Integrated learning for interactive synthetic characters. In *Proceedings of ACM SIGGRAPH*, pages 417-426, 2002.

[13] T. Conde, W. Tambellini and D. Thalmann. Behavioral Animation of Autonomous Virtual Agents helped by Reinforcement Learning. *Lecture Notes in Computer Science*, vol. 272, pages 175-180, Springer-Verlag: Berlin, 2003.

[14] K. Sims. Evolving virtual creatures. In *Proceedings of ACM SIGGRAPH*, pages 15-22, 1994.

[15] R. Grzeszczuk and D. Terzopoulos. Automated learning of muscle-actuated locomotion through control abstraction. In *Proceedings of ACM SIGGRAPH*, pages 63-70, 1995.

[16] R. S. Sutton and G. Barto. *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[17] T. Mitchell. *Machine Learning,* McGraw Hill, 1997.

[18] R. Pfeiffer and C. Scheier. *Understanding Intelligence*, MIT Press, 1999.

[19] S. Russell and A. L. Zimdars. Q-Decomposition for Reinforcement Learning Agents. In *Proceedings of Conference on Machine Learning ICML*, pages 656-663, 2003.

[20] P. Abbeel and A.Y. Ng. Apprenticeship Learning via Inverse Reinforcement Learning. In *Proceedings of Conference on Machine Learning ICML*, vol. 69, ACM Press, New York, 2004.

[21] A. Y. Ng and S. Russell. Algorithms for Inverse Reinforcement Learning. In *Proceedings of Conference on Machine Learning ICML*, pages 663-670, 2000.

[22] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, Prentice-Hall, 1995.

[23] M. Lutz. *Programming Python*, O'Reilly Ed., Sebastopol, CA, 1996.

[24] M. Ponder, G. Papagiannakis, T. Molet, N. Magnenat-Thalmann and D. Thalmann. VHD++ Development Framework: Towards Extendible, Component Based VR/AR Simulation Engine Featuring Advanced Virtual Character Technologies. *Computer Graphics International (CGI)*, pages 96-104, 2003.

[25] M. Kallmann, H. Bieri and D. Thalmann. Fully Dynamic Constrained Delaunay Triangulations. *Geometric Modelling for Scientific Visualization*, **4**:74-123, Heidelberg, Germany, 2003.

[26] R. Boulic, N. Magnenat-Thalmann and D. Thalmann. A Glogal Human Walking Model with Real-Time kinematics Personification. *Visual Computer*, **6**:344-358, 1990.

[27] L. M. G. Gonçalvez, M. Kallmann and D. Thalmann. Defining Behaviors for Autonomous Agents based on Local Perception and Smart Objects. *Computer & Graphics,* **26**(6):887-897, 2002.

[28] J. Dinerstein, P.K. Egbert and H. de Garis. Fast and learnable behavioural and cognitive modelling for virtual character animation. *Computer Animation and Virtual Worlds*, **15**:95-108, 2004.

[29] M. Kasper, G. Fricke, K. Steuernagi and E. von Puttkamer. A behavior-based mobile robot architecture for learning from demonstration. *Robotics and Autonomous Systems*, vol. 34, pages 153-164, 2001.

A Flexible Framework for Representing Personality in Agents

André M. C. Campos [1], Emanuel B. Santos [1], Anne M. P. Canuto [1], Rodrigo G. Soares [1], João Carlos Alchieri [2]

[1] *Informatics and Applied Mathematics Department, Federal University of RN Natal, RN - BRAZIL, 59072-970*
Telephone: +55-84-3215-3813

[2] *Psychology Department, Federal University of RN Natal, RN - BRAZIL, 59072-970*
Telephone: +55-84-3215-3590

{andre, anne}@dimap.ufrn.br, {emanuel, rodrigo}@ppgsc.ufrn.br, jcalchieri@brturbo.com

ABSTRACT
Aiming to represent individuals in a very realistic way, several works have attempted to introduce personality characteristics into artificial agents, usually based on theories of human personality. However, different theories require different structures in the agent architecture, reflecting on the way they influence agent behavior. This paper presents a flexible framework to generalize the influence of personality traits on the behavior of the agents. The framework provides an easy way to test and compare different personality models. In order to validate the framework, a simulation of working environment was developed and three personality theories (Big Five, Jung and Millon) were tested.

Categories and Subject Descriptors
I.2.11 [**Distributed Artificial Intelligence**]: Intelligent agents, Languages and structures, Multiagent systems. I.6 [**Simulation and Modeling**]: Applications.

General Terms
Design, Standardization.

Keywords
Agent architecture, Agent-based framework, Personality-based agents.

1. INTRODUCTION

The notion of autonomy is an essential concept for social systems. From the social point of view, autonomy is an important aspect to distinguish individuals within a society. This concept is largely used in the agent paradigm when modeling social systems [6].

When modeling human social systems, autonomy becomes intrinsically related to how the agents behave and what make them to behave differently for each other. In other words, it is necessary to identify what makes two individuals playing the same roles in a society (or in a group) and facing the same situations to behave differently.

Psychologists have tackled this issue for several decades through what they named *Personality* [2]. Personality represents the structured and dynamic set of characteristics of an individual, normally acquired from the environment and personal experiences [4]. The study of personality aims the understanding of the dimensions that allow individuals to be psychologically distinguished from another [13].

Therefore, for social systems, autonomy of an agent is also a result from its personality. In order to take the autonomy of agents into account when modeling human social systems in a low-level of abstraction, it is necessary to include the notion of individual personality. Several agent actions and characteristics, even the internal ones, may emerge from the encounter of the event perceived by the agent and its internal personality.

The use of the multi-agent paradigm for building social system simulations leads to a more natural way to represents an observed system, since its dynamics is expressed in terms of interactions between autonomous agents. However, most of the agent architecture used in multi-agent simulations considers the agent behavior based only on rational processes (e.g. BDI [5]). This does not provide tools for representing the chaotic nature of social systems. The use of personality in multi-agent applications for simulating social systems can lead to more chaotic (and realistic) results. This is one of the objectives of works dealing with synthetic actors with personality [8][12][10] and also in crowd simulations [7].

Another example of social scenario where the personality of the individuals has strong influence on the emergent behavior is in a working environment [14]. Individuals playing the same role in a working organization may have very different reaction (and consequently different performances in their tasks). This happens, among other reasons, because they have different personalities, affecting, as a consequence, the behaviors of these individuals.

There are several psychological theories and paradigms highlighting the concrete aspects of the influence of personality on the way in which we behave. However, none of them has reached a complete consensus among psychologists. Indeed, personality is subjective and subject to different interpretations. Specifically for working environment, in the Organizational

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. to copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

Psychology area, several models have already been proposed. Different companies may use different approaches for understanding their employees. A simulation application aiming to be generic enough to be able to represent different models will certainly face the intrinsic characteristics that is present in a model, but not in another one.

In order to solve this issue, it is necessary to conceive an agent architecture able to be adaptable to different models of human personality. Surrounding this architecture, a framework must be placed for providing enough flexibility to take into account several facets of the human personality. This paper describes the framework developed to smooth out the aforementioned problem. Flexibility is provided by description files and action scripts, separating the agent structure from its personality and behavior. Three models were implemented for testing purposes, based on the theories of Big Five [3], Jung [9] and Millon [11]. Also, a simulation application, named SimOrg – Simulation of Human Organizations, was adapted to this framework. This paper presents the framework structure and how personality-based behaviors are modeled in it.

Before presenting how different personality models can be handled in the proposed framework, it is necessary to analyze some characteristics usually found in different theory models. The Section 2 presents three theories already used in previous works related to agents. Models of those theories are also used for validating the framework. With those concepts in mind, it is possible to more easily understand the choices for the framework, which is presented in the following section (Section 3). In this section, it is pointed out how to generically represent personality in agents. The generic framework for personality-based agents is described in Section 4. In this section, it is also presented a tool to generate the scripts and description files for the framework. After, it is presented an application used to validate the framework. The application is a simulation tool for project management.

2. HUMAN PERSONALITY

Among the various paradigms (psychoanalytic, behaviorist, humanistic, biology and evolutionary, and personality traits), personality trait is the most widely accepted theory. This approach structures the human personality by a hierarchical model of traits. The behavior of an individual might be "interpreted" as a combination of the different levels of intensity of the different traits. The next sections present three different approaches for modeling personality traits. They will be used to validate the framework proposed in this paper.

2.1 The Big-Five Model

Among the several ways of organizing personality traits, the "Big-Five" dimensions (used by the Five-Factor Model [3]) have a large acceptation by psychologists, as well as computer scientists modeling personality into human-like artificial characters (virtual humans) (see, for instance [6]). It describes the human characteristics according to five main factors: extraversion, agreeableness, consciousness, openness to experience and neuroticism.

The combination of those five factors reflects the several aspects of human personality. For instance, an authoritarian person would be high on conscientiousness, but low on agreeableness and openness to experience [1] (example cited in [4]). Other combinations might be possible, resulting in a large variety of personality styles.

Although it is widely accepted, the Big-Five Model presents a constraint for us, who are modeling artificial behaviors (and not studying people). Whereas the Big-Five Model maps a behavior into its five factors, one cannot derive a behavior from the level of the factors. In other words, it is not possible to predict if someone is authoritarian only by the means of the level of his/her conscientiousness, agreeableness and openness to experience. Individuals with the same level of those factors might behave differently for the same situation. Indeed, a behavior is also a function of the personal experiences from the past and the personal ambitions for the future.

2.2 The Jung Model

Carl Gustav Jung has also proposed a theory that attempted to describe the differences of personality using categories of introversion and extroversion and also using cognitive and emotional functions. These aspects can be seen as the basic dimensions of the individual differences. According to Jung, these dimensions offer a basic typology needed to classify individuals [7].

In order to identify a psychological type, it is necessary to determine the fundamental attitudes of an individual. In other words, to define whether a person is preferentially guided through its internal world (introversion) or through its external reality (extroversion). Once it is defined, an individual can be classified according its dominant function, also known as *Main Function*, which describes the preferential behavior of the individuals. There are four possible psychological functions that can be chosen as dominant, which are the following: Thought, Felling, Sensation and Intuition.

Jung divided the four aforementioned functions into two classes, rational and irrational. The rational functions (thought and feeling) help to make judgments and decisions, and the irrational functions (sensation and intuition) supply information in which it can be based on those judgments. The fundamentals attitudes (introversion and extroversion) may be combined with the fours functions (thought, felling, sensation or intuition), which can lead to eight different combinations, constituting eight psychological types to represent the dominant function of an individual. In order to provide a more complete description of an individual, a classification of the least preferred behavior can also be defined. It uses the same function classification of the dominant function and it is called auxiliary function. It defines the function that an individual utilizes to cope with the direction least preferred situations.

According to Jung, all the aforementioned functions are present in every individual, in higher or smaller degree, but just one is predominant, which is defined in the *Main Function* [3]. The auxiliary function is identified as the function that someone uses to work with less favorite directions and it is usually the opposite of the main function. For example, if the main function is one of the rational functions (thought or feeling), the marginal function it will be one of the irrational functions (sensation or intuition) or vice-versa [3].

2.3 The Millon Model

The theoretical model of personality proposed by Millon takes into account the importance of biological factors such as genetics [11]. These factors are described in one specific matrix for each individual, which represents a fundamental role in the formation of personality [13]. This matrix is also composed of environmental perceptions and actions taken during conflict situations. During the development of an individual, personality has influence from biological and psychological factors, that interact in an endless spiral, in which each circle of this spiral is constructed over previous interactions, creating, in this way, new bases for the next interactions [14].

Millon proposed a measurement to express personality, which is based on the theoretical comprehension of the actions taken to reach the goals that an individual has in his/her life. Also, it takes into account the way to process information received from the environment. In this sense, it has been elaborated a tool to verify a dynamic configuration of interactions, representing three large areas: motivational aims, cognitive styles and inter-personal relations.

According to Millon, personality can be viewed as a result of twelve bipolar[1] attributes. The area Motivational aims includes three bipolarities, linking the ecological and evolutionist theories. It is based on the conceptual antecessors of theses two theories through three main formulations: the existence of organisms, its adaptation to the environment and the answers provided in this relation. Based on this theoretical model, Millon presented the following polarities: openness versus preservation, modification versus accommodation and individualism versus protection. In these groups of attributes, it can be observed the orientation of an individual in relation to the role of the environment as a motivation of the actions of an individual [14].

The cognitive styles can be found in the evolutionist perspective as well as in contribution of authors such as Jung and Myers. They are related to how individuals are oriented when interacting to the environment. Their main aim is to evaluate the way an individual can process information, along with three main models proposed by Jung (extroversion versus introversion, feeling versus thinking and sensation versus intuition). Based on this and in [15], the bipolarities can be defined as extroversion versus introversion, feeling versus thinking, reflection versus affectivity and systematization versus innovation.

Millon proposed to use the inter-personal component to evaluate the style of the relationship of an individual with the others. All bipolarities of this area are based on a bio-psychosocial theory of reinforcement learning as well as active and passive strategies. The bipolarities are: shyness versus communicability, doubt versus security, discrepancy versus conformity, control versus submission and satisfaction versus dissatisfaction.

3. REPRESENTING PERSONALITY INTO AGENTS

One of the issues of representing personality into agents is the transposition of the concepts coming from Psychology, in general difficult to generalize (each individual is unique), to the computational form, which functioning tries to systematize and, as a consequence, to generalize behavior. However, although personality is defined as "*an individual's characteristics pattern of though, emotion, and behavior*" [4], the main issue of representing human personality into agents is the use of the aforementioned theories on the inverse way in which there were conceived. Indeed, human-personality theories tries to generalize the behavior of an individual within a pattern group, but this does not necessarily means that knowing the group an individual bellow one can predict the individual behavior for every situation.

[1] *Bipolarity can be seen as a complementary measure between two concepts. For instance, if an individual has 30% of modification, it automatically has 70% of accommodation.*

In other words, from an observed behavior pattern, it is possible to categorize an individual (according to a theory), however it is hard to generalize a behavior from a categorized individual.

The models used by psychologists to acquire personality traits of an individual aim to "classify" the individual behavior pattern according to the different dimensions (for instance, the five-factor model deals with five characteristics). This classification is then a function of the observed behavior pattern (or from a pattern acquired from filling a questionnaire), mapping the behavior to a classification. However, the mapping function is not bi-directional for every situation. An individual classified as introverted does not necessarily acts introverted for all situations, but at least in those ones observed or asked in her/his questionnaire.

In order to simplify this issue, without losing generality, the function that maps personality to behavior patterns can have an error parameter for representing possible variations on the behavior pattern. The model proposed by Jung attempted to solve this issue through the Auxiliary Function, which represents the behavior pattern used when the Main Function is not trigged (Jung worked deeply on the subject of unconsciousness). In the same way, an agent architecture dealing with personality must handle unpredictable behaviors of the agents. The agent behavior is then a function (Eq. 1) of the situation it is facing (*s)*, its personality characteristics (*p*), and the level of error (*e*), which model the variation of the personality attributes in the agent behavior.

$$b = f(s, p, e) \qquad \textbf{Eq.1}$$

The way in which the personality is represented and how it is handled, providing a general framework for personality-based agent systems, is presented in the next section.

4. A GENERAL FRAMEWORK

As previously mentioned, autonomy is an attribute intrinsically related to social systems. In the context of agent paradigm, this is related to Multi-agent systems, where agents are organized in one or more groups. Agents also perform tasks and, eventually contribute to each other through coordination mechanisms. The proposed framework takes those characteristics into account, conceiving a way to structure the agents inside its multi-agent system as well as how they coordinate their actions. The source of inspiration for defining the framework was the way in which individuals work within of a human organization. Several pieces of code were also adapted from a previous framework developed by one the author of this paper; the framework MAVIS [17]. The MAVIS framework was developed for ecosystem simulations based on reactive agents. The simulation structure of the framework (simulation kernel, event treatment, etc.) was conserved, but the agent-side of the framework was remodeled for taking into account the agent architecture presented in the paper.

In a working environment, individuals are grouped and structured in a hierarchical way, composing the configuration of organization. Individuals may also be member of several groups. This approach is similar to organizational model AALAADIN [15] used by the MadKit platform [16]. Usually, in working environment, there is also a process guiding the actions of individuals. The process indicates which activities must be performed for the first place, which are the dependencies of each activity, and so on. This idea was also translated to the framework proposed in this paper.

Then, the framework takes into account three different aspects of a multi-agent system:

- **The organizational structure**: that defines the static component of the system. In this model, the hierarchical distribution of agents in groups and the roles they will play inside their groups are defined. The roles are useful for determining the actions an agent is responsible for;
- **The organizational process**: that defines the dynamic component of the system. It structures how the agents integrate their actions to achieve a common goal. The organizational process is organized in activities, where each activity is an set of actions; and;
- **The behavioral component**: that defines how the agents perform the tasks described in the organizational process. It is in this component that personality attributes plays an important role. Each action that an agent performs will suffer influence on its personality attributes.

In order to provide flexibility for these three aspects, the framework takes in charge three description files, each one for a different aspect. In this sense, the organizational structure, process and behavioral component can be customized according to the application without changing the framework code.

4.1 The agent architecture

Each agent in the framework represents an individual within a multi-agent organization. The interactions, rules, structures and interaction protocols that happen within the organization are dependent to the way the agents process information. Then, it is necessary to model the internal architecture of the agents in order to structure the framework. Moreover, the agent architecture must also take into account personality attributes when enabling, disabling or activating an agent action.

The architecture for the agent is shown in Figure 1. As it can be observed, the defined architecture contains not only modules commonly used in a common agent architecture, such as perception, planning, goal definition and action, but also it is composed of a base, defining the artificial psychological profile of the agent.

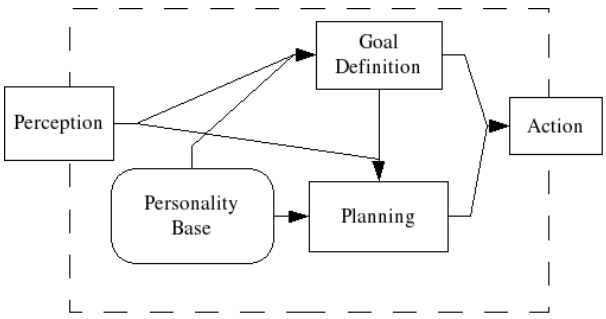


Figure 1 – The agent architecture.

As personality can be seen as the way in which an individual interacts with the environment, it is represented as an information base, or personality base. Depending on the personality model used by the framework in a specific application, the attributes used in that base might differ. For instance, for the model proposed by Millon, it is composed of the three aforementioned topics of the human personality (motivational aims, cognitive modes and inter-personal relations), each area having its respective bipolar attributes.

The personality base has strong influence in the definition of the goals of an agent (goal definition module) as well as in the choice of the actions to perform the chosen goal (planning module). In other words, when the module of Goal Definition is activated, it can request information from the personality base in order to define the agent goal.

In considering the personality base as a common knowledge base, this agent can be seen as a common BDI (Beliefs, desires and intentions), in which based on its beliefs, an agent can choose its goals as well as the needed actions to reach this goal. The use of personality will help to transform the modeling in a more natural one, in which the personality of an individual should and will be taken into account.

As it can be observed in Figure 1, there are some arrows indicating the influence of the personality base over the modules of Goal Definition and Planning. The arrows represent mapping functions from the personality to the respective modules. This mapping is provided by behavioral scripts, explained more in detail in next section.

In order to model the complex set of behaviors that an agent might perform within an organization, it is necessary a huge amount of rules associated to the agent. Depending on the role the agent plays inside the organization (or inside the groups of the organization), the agent must organize its activities (or tasks) according to a priority criterion. This priority order of the activities that an agent is responsible for is modeled by a horizontal layered architecture, similar to the reactive architecture defined by Brooks (Subsumption). This architecture was however extended to provide the following needs:

- **The ability to deal with two level of action abstraction:** The first level is composed of activities, which represents the tasks the agent are responsible for. For instance, an agent that is member of two groups might play different roles at the same time, having, as a consequence, to execute different activities. The second level is composed of actions. In fact, an activity is defined as a set of actions that might be performed in a specific sequence or not;
- **The ability to dynamically define priorities for activities:** In the agent architecture, the layers define the level of priority of an activity. However, this priority might be changed dynamically as a result of an action performed by the agent. Whenever the priority of an activity is changed, the respective set of actions assigned to the agent will also change.
- **The actions of the activity with the greater priority compete between themselves to be activated:** If an agent is executing an activity composed of several actions that one does not depends on the other, the independent action will compete between themselves to be activated, according to a probability degree. This degree is defined as a function that maps the attributes specified in the personality base and the corresponding action. This approach is suitable for representing the "personal" choice of the action, since there is no dependence of the competing actions.

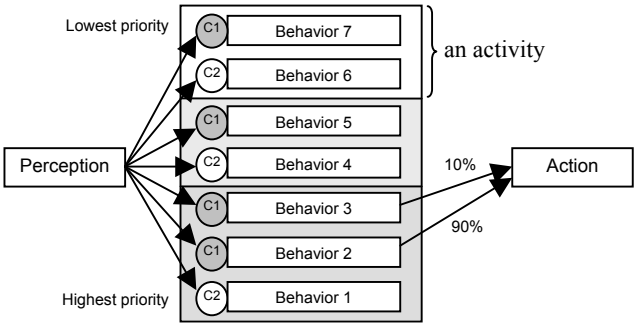


Figure 2 – Functional schema for the agent architecture.

The Figure 2 illustrates the functioning of the layered architecture. Each behavior rule, in order to be activated, needs that its antecessors (represented in the figure as C1, C2, etc.) be validated. Among the behavior rules of the activity with the highest priority (in the figure, represented by the Behaviors 1, 2 and 3), just one will be activated. Supposing that two of them have their corresponding conditions validated (C1), they will compete according to the "personal" interested of the agent. The system will then define the degree of probability for activating the behaviors (in the case illustrated by Figure 2, it is the behaviors 2 and 3) according to the personality attributes saved in the personality base, resulting in the most probable action the agent will take.

The way in which the agent architecture maps the personality to the actions is through a function, scripted when implementing a behavior action. The function itself is based on empirical feedback given by an expert (psychologist). There is no good or bad match between personality and actions, however it normally requires several simulation executions before the simulation model (the script function mapping personality and actions) be considered calibrated by an expert. The next subsection describes the simulation model more in detail.

4.2 Scripting Personality and Behavior

The personality model in the proposed architecture influences the agent behavior. Since, as previously mentioned, there are several ways to represent personality, it is necessary to adapt one of them in a specific application. In order to do this adaptation in easy way, we used a rapid prototyping strategy. We moved the behavior part that relates the actions execution and the personality models into a bean shell script. Therefore, in order to update the agent behavior, it is just necessary to alter the script file.

An action is a Java class that encapsulates an action script. The action accesses the script interpreter and, then, calls its methods separately. Each action method is related to a script method. The main methods that the script must implement in order to integrate with the framework are:

- **The evaluating method**: `validate()`: it is used for calculate the probability of the action to be executed, according to the personality parameters;
- **The quality method**: `getQuality()`: it calculates the amount of work the agent will do in that action. The higher the quality is, the better the action execution, and;
- **The execution method**: `execute()`: it performs the sequence of steps for an specific action.

Besides the methods, the script has the personality type data. The simulation uses that data to link the agent personality to the action ideal characteristics, influencing, in this way, the evaluation and quality methods. In other words, in order to have a better chance to execute an action, the agent must have a personality that is close to the ideal profile described in that action.

As presented in section 3 (Eq.1), the behavior of an agent also takes into account a level error, representing unpredicted behavior. This error is a randomly generated value in [-0.1;01] that is added to the personality attributes when the `validate()` and `getQuality()` methods are requested. Then, a level of error is introduced for simulating unpredicted behavior.

In order to facilitate the end-users of the framework, a tool for generating the scripts files was developed. As previously mentioned in the framework description, the framework deals with three different aspects of a multi-agent system. The tool attempts to facilitate the handling of theses aspects by generating three different files, all of them specified in XML format:

- The organization structure file.
- The work process file.
- Several files containing the personality and scripts.

The file containing the personality descriptors is written for each personality model. It consists on three main tags:

- The `script` tag containing pieces of bean shell script;
- The `personality_descriptors` tag describing the personality stereotypes for each model;
- And the `action_descriptors` tag describing the personality ideal profile to execute the action;

Figure 3 illustrates description of a personality following the Jung approach of personality. Bean shell pieces of code constitute the `script` tag in the XML file. These pieces of code compose the actions files. They have a method-like format to enable the action script generation. This kind of format is necessary to get the action to execute correctly for each model. Since the Jung, Big Five and Millon models have different approaches, it is not possible to use the same evaluation model for them.

```
<model theory="JUNG">
    <script>
        <method name="execute">
int p =
entity.getActivityPriority(activity.getName());
runningTime++;
aux = getQuality(entity);
...
        </method>
        <method name="quality">
personality = entity.getPersonality().getName();
a =  ideal{personality};
...
return new Double(apt);
        </method>
    </script>
    <personality_descriptors>
          <personality name="ES">
            ...
          </personality>
    </personality_descriptors>
    <action_descriptors>
          <action name="code" size="700">
              <property name="ES" value="0.7"/>
          </action>
    </action_descriptors>
</model>
```

Figure 3 –XML file illustrating the personality description.

The `personality_descriptors` tag consists of personality stereotypes. Each stereotype is a personality pre-defined description. They were conceived by a specialist to give a theoretical basis to the use of human personality in the simulation.

Finally, the `action_descriptors` is composed of personality descriptors. This part of the XML file links the action to the personality models. This part of the file contains information about the ideal personality profile to execute the action, this data are used to generate the script file as well.

An agent action is composed of a sequence of several steps. The agent might read its messages in the mailbox update its internal states and/or execute its behavior pattern. Part of agent cognitive mechanism stay in the behavior. The behavior evaluates the list of activities in which the agent is involved. When an activity is chosen (by its priority), the agent create a list of actions. In next step, the agent validates its list accessing the method "validate" of action. This method answer if the action can be executed (it is the condition explained in Figure 2). When more that one action is valid, the agent begins the decision process. On that moment the behavior access the `evaluate()` method of the action. This method determine the probability of the agent execute an action. Then, the personality descriptor of the agent is compared to the personality descriptors of the action. The comparison result is used for provide the degree of probability that the action will be executed by the agent. Finally, when an action is chosen, it starts the its execution. To execute an action the method `execute()` of action is called. During the action execution, it is necessary to access the method `quality()`. This method, as well as, method `evaluate()` compare personality descriptor, but for a different purpose. The method `quality()` use the information of personality to determine the agent performance when execute the action. The greater is the quality of action performed by an agent better is the result from that execution. An example of personality profile used in the framework is illustrated in Figure 4.

```
<personality name="competitive">
   <!-- motivation targets -->
   <property name="openness" value="0.3"/>
   <property name="modification" value="0.7"/>
   <property name="protection" value="0.5"/>
   <!-- cognition mode -->
   <property name="extroversion" value="0.7"/>
   <property name="sensation" value="0.7"/>
   <property name="reflexion" value="0.5"/>
   <property name="innovation" value="0.1"/>
   <!-- interpersonal relationship -->
   <property name="communication" value="0.5"/>
   <property name="selfconfidence" value="0.5"/>
   <property name="discrepancy" value="0.3"/>
   <property name="submission" value="0.5"/>
   <property name="concordance" value="0.5"/>
</personality>
```

Figure 4 – Example stereotype for competitive agents.

The XML files can be written manually or using the graphical tool showed in Figure 5. The graphical tool can read, edit and write the XML files used by the framework. It is a friendly user interface, that it abstracts the user of the XML complexity.

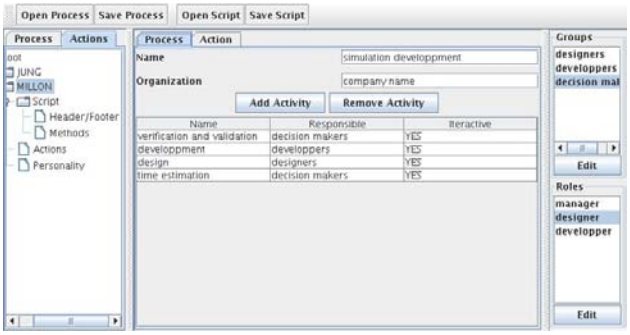


Figure 5 – Initial display of the XML generator tool.

5. APPLICATION TO WORKING ENVIRONMENT SIMULATIONS

The proposed framework was developed to solve the flexibility issue faced during the development of a simulation application, named SimOrg. This application aims to reproduce the organizational dynamics found in working environment and can be used for studying individual behavior within an organizational as well for education and training purposes.

As the original version of the SimOrg application does not give flexibility for representing personality-based agents on different models, a new version of the simulator was proposed in order to generalize the application. The adaptation of SimOrg to the framework provided the ability to accept a larger number of agent personality models such as the Millon, Jung and Big Five models.

5.1 The SimOrg Scenario

In order to analyze the influence of personality in the performance of the SimOrg agents, it has been defined a scenario for a working organization, where agents could not only interact and act according to their own personality, but also follow a working process (organizational rules). The scenario was based on the experience of the authors in developing software. In this sense, the fictitious organization is a software development company.

In the scenario, the chosen development process is an iterative one based on agile methodologies [18], which was simplified for testing purposes. Agile methodologies are ideal for testing the proposed architecture as individuals and their informal communication are very important for them. In other words, the success of a project is not only a matter of passing formal documents through staff., but also individual interactions that can represent a successful project.

The scenario was then represented as having formal, informal interactions and task actions, like *manager, design*, *estimate time*, *code*, *test code* and *validate design*. Those actions were organized into four activities, each one having an individual or group responsible for.

The communication acts follows a small subset of an ACL structure, composed of *performative*, *sender*, *receiver* and *contents*. Although this is enough for representing communication between agents, it was observed that human communication is not necessarily binary (to send or not to send a message). In order to model working human organizations, it is also necessary to represent how good (or bad) an individual performs an action, including communication acts, and the importance that an individual gives for those actions. For instance, suppose that a very introspective individual can not perform a task as desirable. Although there is reluctance in

asking for help, s/he will end up asking for a colleague help. The way in which s/he performs such an action (*quality*) and the need for doing that (*importance*) will certainly be influenced by his/her personality (very introspective). As a consequence, we have introduced in our ACL subset two other components: *quality* and *importance*, whose values are functions of the individual personality attributes (openness, introversion etc.). In this way, an agent is also able to react to a received message according to its quality and importance.

5.2 Scripting the scenario

The structure of the software company described in the scenario is composed of managers, designers and developers; these are meant as roles in the organization and on the simulation as well. The workers (agents) are allocated on groups that are semantically separated according to the roles of their agents. An agent performs one or many of those roles in the organization depending on the groups in which the agent is. In other words, for each group that the agent A belongs, a role is associated to A in the group. A group must always have roles agents that will play them. In this XML script, this is represented by the organization description file. The Figure 6 shows a piece of the XML file that describes the agents and its roles in a group.

```
<group name="X" type="developpers">
   <agent-role agent="C" role="manager"/>
   <agent-role agent="E" role="developper"/>
   <agent-role agent="F" role="developper"/>
</group>
```

Figure 6 – A piece of code representing a group and the roles and agents associated to it.

Once the static part of the organization is defined, the agents will have to perform some actions. The organization sets the actions that each role will execute, so the agent will perform all the actions associated to its roles like in real organizations. The set of possible roles is: Manager, Designer and Developer. And the set of actions is: Manager, Estimate Time, Design, Code, Validate and Verify.

The organization, in order to accomplish the software product manufacturing, must complete some activities in certain order. An activity is a part of the organization's process; it consists of an actions sequence, so when all the actions in one activities are done, the activity itself will be over. Activities can depend on other activities in a certain percentage, that is, suppose that the activity B depends on the activity B in 30%, then when 30% of A is done, B can begin. There are four activities in the organization process: time estimation, design, development and verification and validation. The design activity depends on time estimation, the development activity depends on 30% of the design and verification and validation activity depends on 30% of the development activity.

The actions were conceived in order to clearly distinguish the agent structure: its motivation, fatigue, roles; the agent personality and the action itself. For each action is defined in its script an ideal suitable personality to perform that action, the closer the agent personality is to that ideal, the more efficient the agent will be executing that action. Each agent has an attribute called motivation, this feature also defines the quality of the agent work on the action, if an agent has got a low motivation, this will affect negatively in the action performed by that agent. Another attribute is the fatigue; it influences the agent motivation, so indirectly influences the quality of the agent work.

All those settings for the organization and the process are made in the XML file. The user can share the scenario file that he has build in the application through the Web and then other users can construct their simulation scenario based on that one.

5.3 SimOrg Application

At this stage of the implementation, a stable version of the Simulator is ready and on use. Some tests have already been done in the current version and it is now being validated by possible users. Also, an interactive graphic interface has been built in order to fulfill the requirements of an interactive simulation, as the SimOrg simulation is. The main graphic elements of the simulator are the following: the different data visions of the fictitious organization and the actions that the users could use to interact with the simulations.

Figure 7 illustrates a screenshot of the initial screen of the simulation system. On the top left-hand part of the figure, shown by arrow 1, it is possible to have a vision of the organization, described in a tree-structure. It shows the structure of all groups hierarchically. In Figure 7, for instance, the simulated organization has three groups, decision makers, designers and developers.

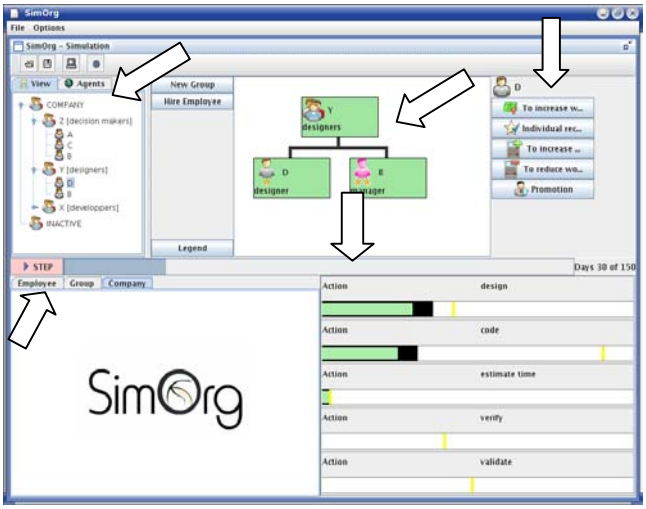


Figure 7 – Screenshot of the initial screen of the simulation.

On the top center part of the figure, shown by arrow 2, it is possible to see the organization chart, which describes the distribution of the agents into the groups. Also, it describes information about the state of the agents, such as the role of the agents in the group. The organizational chart can represent the whole company or parts of it. In Figure 7, for instance, it is shown the organizational chart of the designer group, which is composed of one designer (D) and a manager (E). In the organization chart and in the tree-structure, a user has the permission to change the structure of the organization, changing some agents to different groups and roles. Also, a user can perform user actions for the selected agent, pressing the left bottom of the mouse.

On the top right-hand part of the figure, shown by arrow 3, all the possible user actions are shown and, at any time during the simulation process, a user can choose to apply a user action. As already mentioned, the user actions can be done in order to improve the motivation level of an agent, aiming to increase the productivity of the organization.

On the bottom left-hand side, shown by arrow 4, there is a panel which contains information about agents, groups and the simulated organization. In this part, information about the agent history, such as: the roles that the agent performs within the group and activities of the groups, activities already performed by the agent, are described.

On the bottom right-hand side of Figure 2, shown by arrow 5, action graphics are described. These graphics represent the progress of all active actions of the system in relation to time. The main aim of these graphics is to show how early or late the agents are in the execution of the actions. The graphics show the estimated time of an action (yellow line) as well as the amount of time already used by the agents. In order to let the understanding of the graphic more intuitive, there is a variation of colors to represent the progress of the actions. The green color means that there is no delay of the action until the current iteration. On the other hand, the red color represents a delay in the progress of the action Also, the black part of an activity means the optimal execution time. In using the real and optimal execution time, it is possible to analyze how late an activity is.

In addition to that, as an activity can be performed by a group of agents, in analyzing the agent history (bottom left-hand side), it is possible to detect which agent is having the most negative contribution to the delay in the execution of the activity. Also, for each activity performed by the agents, it is possible to visualize a graphic in which the contribution of each agent in the execution of the activity is described. It is shown when pressing the left button of the mouse in the activity required.

6. CONCLUSION

This paper presented a flexible framework for representing agents in a multi-agent system taking into account personality attributes for deriving the agent behavior. The flexibility refers to the ability to represent human personality into agents by the means of different models. This framework is based on the concept of working environment and reflects the static and dynamic structures and processes usually found in working organizations. As proof of concept, an existing simulation application, SimOrg, was adapted to the framework, having now the ability to represent personality in several ways.

The gain of the flexibility provided by the framework in this specific application is the possibility to easily adapt SimOrg to different companies. Indeed, as there is no consensus among psychologists about human personality, the Human Resource Department of company might work in a differently from another one. The simulation training tool SimOrg must then reflect the concept used by the company.

The ability to quickly change behaviors and personality traits has also been important tool for psychologists, since they can now analyze some important behavioral aspects of groups by prototyping by themselves their own experiments. These analyses will help the psychologists to define which aspects will have influence, for instance, in the productivity of a group and, then, to define which solutions could be done in order to improve the productivity of a working organization.

7. ACKNOWLEDGMENTS

This work has the financial support of CNPq (Brazilian Research Council), under process number 552431/2002-8.

8. REFERENCES

[1] Church, A., and Waclawski, J. The relationship between individual personality orientation and executive leadership behaviour. *Journal of Occupational and Organizational Psychology*. 71(2):99-126. June 1998.

[2] Cooper, M., and Sheldon, M. Seventy years of research on personality and close relationships: Substantive and methodological trends over time. *Journal of Personality*. 70(6):783, December 2002.

[3] Digman, J. Personality Structure: Emergence of the Five-Factor Model. *Annual Review of Psychology*, 41:417-440, 1990.

[4] Funder, D. *The Personality Puzzle*. Norton Ed. New York, US. 1997.

[5] Geoergeff, M., Pell, B., Pollack, M., Tambe, M., and Wooldridge M. The Belief-Desire-Intention model of agengy", *Proceedings of the 5th International Workshop on Intelligent Agent. Agent Theories, Architectures and Languages*, 1998.

[6] Gilbert, N., and Troitzsch, K. *Simulation for the Social Scientist*. Open University Press. 1999.

[7] Goldenstein, S., Menelaos K., Dimitris M., Guibas L., Aaron, E., and Goswami, A. Scalable Nonlinear Dynamical Systems for Agent Steering and Crowd Simulation. *Computers and Graphics*, 25(6):983-998. December 2001.

[8] Hayes-Roth, B., and Rousseau D. A social-psychological model for synthetic actors. Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98), Minneapolis, US, pp.165-172, 1998.

[9] Jung, C. *Tipos Psicológicos*. (translation Orth, Lúcia) Ed. Vozes, Petrópolis, Brazil, 1991.

[10] Loyall, B., Reilly, S., Bates, J., Weyhrauch, P. System for authoring highly interactive, personality-rich interactive characters. *Proceedings of the SIGGRAPH/EG Symposium on Computer Animation*, pp.59-68. 2004.

[11] Millon, T. Reflections on psychosynergy: a model for integrating science, theory, classification, assessment, and therapy. *Journal of Personality Assessment*, 72(3):437-456, 1999.

[12] Prada, R., and Paiva, A. Believable groups of synthetic characters. *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*. Utrecht , NL, pp.37-43, July 2005.

[13] Revelle, W. Personality Processes. *Annual Review of Psychology* (also available at: http://pmc.psych.nwu.edu/revelle/publications/AR.html), 1995.

[14] Salgado, J. Predicting job performance using FFM and non-FFM personality measures. *Journal of Occupational and Organizational Psychology*. 76(3):323-347. September 2003.

[15] Ferber, J., and Gutknecht, O. A meta-model for the analysis and design of organizations in multi-agent systems. Proceedings of the 3rd International Conference on Mult-Agent Systems (ICMAS98), pp. 128-135, Paris, France, 1998.

[16] Gutknecht, O., and Ferber, J. The MADKIT Agent Platform Architecture. *Proceedings of the 1st Workshop on Infrastructure for Scalable Multi-Agent Systems*, pp. 48-55, June 2000.

[17] Campos, A. and Hill, D., An Agent-Based Framework for Visual-Interactive Ecosystem Simulations. *Transaction of SCS*, 15(4):139-152, December, 1998.

[18] Highsmith, J., *Agile Software Development Ecosystem*, The Agile Software Development Series, Addison Wesley Ed., 2002.

Emotion-based Norm Enforcement and Maintenance in Multi-Agent Systems: Foundations and Petri Net Modeling

Julia Fix
University of Hamburg
Vogt-Kölln-Str. 30
D-22527 Hamburg, Germany
+49-40-42883-2225
fix@informatik.uni-hamburg.de

Christian von Scheve
University of Hamburg
Allende-Platz 1
D-20146 Hamburg, Germany
+49-30-91607790
xscheve@informatik.uni-hamburg.de

Daniel Moldt
University of Hamburg
Vogt-Kölln-Str. 30
D-22527 Hamburg, Germany
+49-40-42883-2247
moldt@informatik.uni-hamburg.de

ABSTRACT
A review of recent theories of emotion indicates close interconnections between emotion and social norms in human societies. We consider the possibility of implementing these mutual influences in a multi-agent system in order to establish dynamic and flexible control structures. According to some theories, emotion plays a key role in establishing and maintaining these structures by fostering the internalization of and compliance with social norms. Here we introduce a Petri Net based approach to modeling the emergence and maintenance of social norms in multi-agent systems.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence - *Intelligent agents*, J.4 [**Computer Applications**]: Social and Behavioral Sciences - *Psychology, Sociology*.

General Terms
Design, Human Factors.

Keywords
Emotions, social norms, MAS, socionics, Petri Net modeling.

1. INTRODUCTION
Controllability of multi-agent systems (MAS) is one of the main challenges for current Distributed Artificial Intelligence (DAI) research: the desire for systems' autonomy, flexibility, and proactivity may conflict with imperative issues of global system control [13]. Achieving system control without abandoning core strengths of DAI systems might be accomplished by using the social mechanisms providing coordination and control in human societies. Applying these mechanisms and their functions to artificial societies is the main goal of this paper.

One feasible solution to the problem just outlined is the implementation of a system of shared norms which is not coerced by the designer, but instead emerges from mutual interactions of agents [15]. To realize such an approach, it might be beneficial to adapt to the computational context the mechanisms of norm maintenance and compliance in human social systems. Proper theorizing and empirical evidence both indicate that emotion is essential in sustaining social norms and enforcing sanctions in cases of non-compliance [6, 7, 9]. This regulatory function of emotion in societal affairs was recently corroborated by empirical findings suggesting that emotions reinforce norm-compliant behavior and punishment of deviant behavior [4, 5].

In support of the plea to integrate theories of emotion into the computational study of social norms [16], we aim to show how emotion theories of different disciplines can jointly specify the interrelation between social norms and emotion. Furthermore, we argue that emotion may provide a basis for establishing solid control mechanisms within MAS by promoting norm-compliant behavior, provided that social norms are at all implemented in the system. First, we offer some initial theoretical foundations and second, we represent selected aspects in a Petri Net based model.

2. THEORETICAL APPROACHES
In this section we will briefly review some social science theories suggesting a crucial function of emotion in sustaining common behavioral norms. Basically, the theories claim that some emotions sustain social norms whereas others (e.g., contempt, disdain, disgust) are incentives to punish violators [6, 8]. "Self-conscious" emotions such as shame, embarrassment, and guilt are, inter alia, evoked by violating a social norm. They are negative payoffs and the desire to avoid them contributes to sustaining social norms within an actor, ancillary to any third party punishment. Thus, the anticipation of certain emotions promotes the internalization and, subsequently, the enforcement and maintenance of social norms.

In addressing the "foundational theoretical problem of the social sciences – the possibility of unconscious, unplanned emergent forms of cooperation, organization and intelligence among intentional, planning agents" [2, p.6], we have argued that the potential of social norms in explaining social structural dynamics cannot be investigated properly without considering the role of emotion [14]. Conte and Castelfranchi view a norm as a mental object, i.e. a hybrid configuration of beliefs and goals [3]. Endorsing this approach, the social, temporal, and spatial distribution of norms makes them instances of a macro system; at the same time, however, their definition as configurations of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

beliefs and goals renders them an instance of the micro level and accordingly a primary subject matter of different kinds of reasoning processes. Previously, we have shown that emotion may be a loophole in this respect [14] and that Elster's [8] conceptualization thereof is well suited to serve as an amendment to existing positions.

Elster proposes a definition of certain qualities of social norms instead of the concept as such [8]. Accordingly, norms are "social" insofar as they are sustained through sanctions by a third party which is by definition unaffected by the norm transgression. Consequently, the presence (or representation) of other agents is necessary to enforce and maintain a social norm. Elster argues that imposing sanctions on the norm violator is driven by emotions such as contempt or disgust, whereas "sanction" may just mean a subtle expression of such an emotion, e.g. a gesture, a facial expression, denial of communication, etc. Even if the norm violator would not suffer any material loss, the effectiveness of sanctions is still ensured through the emotion of shame, which a norm violator will suffer when perceiving the sanction "as a vehicle for the emotions of contempt or disgust" [8, p.146]. Also, the mechanisms described by Elster have in part been validated empirically [9, 12, 5, 1].

Damasio provides evidence for the utility of social emotions in view of the internalization of social norms [4]. His somatic-marker hypothesis explains the role of social emotions in categorizing social situations experienced in individual socialization and ontogeny, in which social and personal experiences are to a particularly high extent emotional. Actors experience a wide range of (social) emotions (e.g., sympathy, pride, contempt, shame) which might be induced by punishment or reward. Gradually, experienced situations are categorized according to their intrinsic structure, components, and their significance in terms of prevailing goals and beliefs. Hence, the resulting conceptual categories are connected with the mechanisms responsible for emotion generation. The intrinsic structure of a social situation, options for action and probable future outcomes are more and more associated with corresponding emotions. The subsequent occurrence of a situation of the same category then more the less automatically induces the respective emotional state [4]. "This arrangement allows us to connect categories of social knowledge – whether acquired or refined through individual experience – with the innate, gene-given apparatus of social emotions" [4, p.147]. Moreover, it is demonstrated how cooperation – as an epitome of the norm of reciprocity – can be stimulated and in fact be reinforced by positive emotions [4, p.156].

In view of the interaction of emotion, norms, and normative behavior we can sum up that shame and contempt in particular serve as vehicles for maintaining norms by promoting normative behavior and avoidance of adverse consequences. Compliance with social norms therefore not necessarily arises as a consequence of anticipating a loss of material resources through sanctions. Rather, it is (also) the result of internalized strategies to prevent emotion-based sanctions (e.g., contempt, disdain, detestation or disgust in the punisher) that would entail negative emotions (e.g., shame, guilt, embarrassment) in the violator. Similarly, emotion-based rewards (by way of admiration, appreciation or approval expressed by observers of normative behavior) reliably cause positive emotions (e.g., contentment, satisfaction, pride) in the conforming actor. In consequence, the anticipation of positive emotions facilitates the internalization of social norms. In what follows, we give a sketch of possible formalization of these views and represent them with a Petri Net model of emotion-based norm enforcement and maintenance.

3. PETRI NETS MODELING

In this section we follow a socionic approach [12] to represent the main arguments in a Petri Net based model. Figure 1 shows a Petri Net representation of sanctioning non-complying behavior by means of social emotions, as suggested in [7] and [8]. Rectangles (transitions) represent actions, whereas circles (places) denote available or unavailable resources or conditions that might be met. Arcs directed from transitions to places represent preconditions for action, whereas arcs directed from places to transitions represent an action's outcome. A firing transition (i.e., an action that is implemented) will remove resources or conditions (tokens) from places and insert them into some other place.

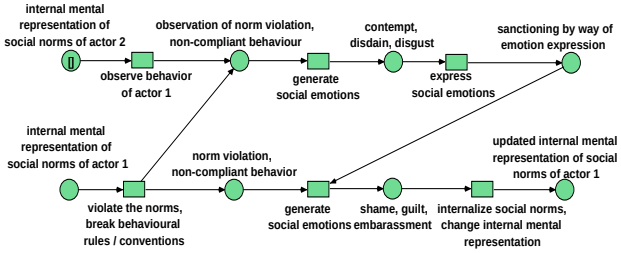


Fig. 1. Internalization [4, 8]

The bottom thread of the net represents the violator's (1) course of action, the upper thread that of an observer / punisher (2). Actor 2 observes the behaviour of actor 1 and perceives a norm transgression (transition "violate the norms, break behavioral rules / conventions"). Social emotions of contempt, disdain or disgust are elicited (transition "generate social emotions") and their expression (transition "express social emotions") constitutes the sanctioning of a norm violator [6, 8] in giving rise (place "sanctioning by way of emotion expression") to negative emotions in the violator (transition "generate social emotions") and induce states of shame, guilt or embarrassment. To integrate the concept of emotion-driven internalization of social norms [4] (s.a.), we extend this model with an additional transition "internalize social norms, change internal mental representation", indicating that certain negative emotions (s.a.) become associated with a situation of norm violation. Occurrence of a similar event would then automatically induce the associated emotions [4]. Associations between negative emotion and norm transgression thus foster the (re-)internalization and reinforcement of social norms and at the same time also update the corresponding internal representation of the norm ("updated internal mental representation of social norms").

Reinforcement of norm-compliant behavior through positive emotions [4, 7, 8] can be modeled in analogy: situations of norm-compliance are associated and internalized along with positive emotions, motivating an actor to seek out situations in which compliance with the (updated) internal representation of a norm leads to intrinsically rewarding positive emotions. Referring to the problem of agent cooperation, this mechanism can be exemplified as cooperative behavior based on adopted social norms within a multi-agent society. In this case, cooperation is mutually rewarded

with different positive emotions, i.e. acknowledgement, gratefulness, or admiration on one side, and pride or satisfaction on the other side. Incentives to cooperate are reinforced on the micro level within a single agent, consequently also reinforcing the norm of cooperation on the macro level of an entire MAS.

Figure 2 integrates processes of sanctioning and reinforcement in a general Petri Net model of emotion-driven norm enforcement and maintenance on the macro level by means of internalization of norms and emotions on the micro level. Here, a powerful variant of Petri Nets – a reference net [17] is used. A notable property of a reference net is the possibility that tokens located on a place in a net (the system net) may again be a reference net (an object net) (or some arbitrary Java object) [17]. Further details on reference nets modeling are omitted here (see [11] for details).

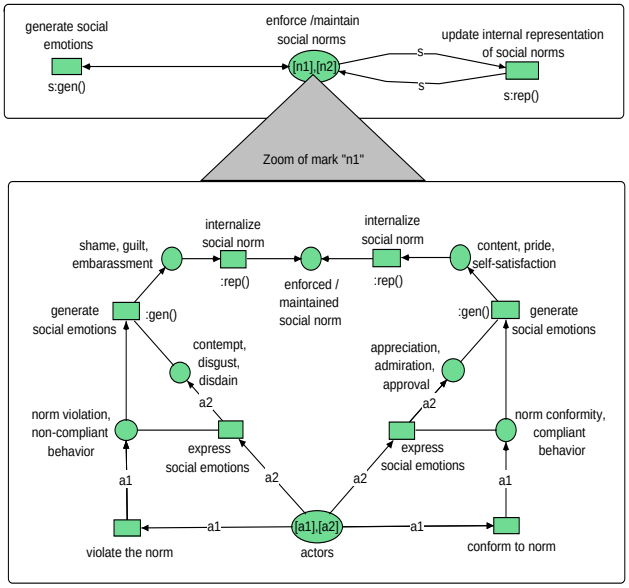


Fig. 2. Emotion-based enforcement and maintenance of norms

4. DISCUSSION

Modeling the social functions of emotion is supposed to improve MAS, e.g. in view of alternative coordination solutions and structuration. Our approach allows modeling the interrelation of social norms and emotion at different levels of abstraction. On the one hand, norms can be modeled on the micro-level, reflecting implications of the internal representation of norms for agents' de facto behavior. On the other hand, norms can also be modeled on the macro-level, thereby focusing on the social mechanisms of norm enforcement and maintenance. A detailed analysis of these approaches and their implications for the dynamics of MAS are clearly beyond the scope of this paper. We basically aimed to show the general role of emotions in enforcing norms in MAS and to evaluate the Petri Net modeling formalisms in view of the interrelation of norms and emotion. The quantitative representation and functions of emotion in agent architectures are expressly *not* addressed herein. Embedding the models into our general multi-agent framework MULAN is the subject of future work.

5. REFERENCES

[1] Berthoz, S., Armony, J.L., Blair, R.J., Dolan R. J. An fMRI study of intentional and unintentional (embarrassing) violations of social norms. *Brain*, 125(8), 2002, 1696-1708.

[2] Castelfranchi, C. The Theory of Social Functions: Challenges for Computational Social Science and Multi-Agent Learning. *Cognitive Systems Research*, 2001, 2(1), 5-38.

[3] Conte, R., Castelfranchi, C. Norms as mental objects. In *Proc. of the 5th Europ. Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW '93)*. Berlin: Springer, 1995, 186-196.

[4] Damasio, A. R. Looking for Spinoza. New York: Harvest, 2003.

[5] Eisenberger, N.I., Lieberman, M.D. Why rejection hurts: a common neural alarm system for physical and social pain. *Trends in Cognitive Sciences*, 8(7), 2004, 294-300.

[6] Elias, N. The Civilizing Process. Oxford: Blackwell, 1978.

[7] Elster, J. Rationality and the Emotions. *The Economic Journal*, 106(438), 1996, 1386-1397.

[8] Elster, J. *Alchemies of the Mind.* Cambridge: Cambridge University Press, 1999.

[9] Fehr, E., Fischbacher, U. Social Norms and Human Cooperation. *Trends in Cognitive Science*, 8(4), 2004, 185-190.

[10] Koehler, M., Moldt, D., Roelke, H. A Discussion of Social Norms with Respect to the Micro-Macro Link. In *Proc. of the 2nd Int. Workshop on Regulated Agent-Based Social Systems (RASTA'03)*, 2003.

[11] von Luede, R., Moldt, D., Valk, R. (Eds.). *Sozionik: Modellierung soziologischer Theorie*. Muenster: Lit, 2003.

[12] de Quervain, D.J.-F., Fischbacher, U., Treyer, V., Schellhammer, M., Schnyder, U., Buck, A., Fehr A. The Neural Basis of Altruistic Punishment. *Science*, 305, 2004, 1254-1258.

[13] von Scheve, C., Moldt, D. Emotion: Theoretical Investigations and Implications for Artificial Social Aggregates. In: G. Lindeman, D. Moldt and P. Paolucci (Eds.) *Regulated Agent-Based Social Systems*. Berlin: Springer, 2004, 189-209.

[14] von Scheve, C., Moldt, D., Fix, J., von Luede, R. My Agents Love to Conform: Norms and Emotion in the Micro-Macro Link. *Computational & Mathematical Organization Theory*. [in press].

[15] Shoham, Y., Tennenholtz, M. On the emergence of social conventions: modeling, analysis, and simulations. *Artifcial Intelligence*, 94(1-2), 1997, 139-166.

[16] Staller, A., Petta, P. Introducing Emotions into the Computational Study of Social Norms: A First Evaluation. *Journal of Artificial Societies and Social Simulation*, 4(1), 2001. http://soc.surrey.ac.uk/JASSS/4/1/2.html.

[17] Valk, R. Petri Nets as Token Objects. An Introduction to Elementary Object Nets. In *Proc. of Application and Theory of Petri Nets*. Berlin: Springer, 1998, 1-25.

An Agent-Based Simulation of Pedestrian Dynamics: from Lane Formation to Auditorium Evacuation

Marcelo C. Toyama, Ana L. C. Bazzan* and Roberto da Silva
Instituto de Informática, UFRGS
Caixa Postal 15064 CEP 91.501-970 Porto Alegre, Brazil
{mctoyama,bazzan,rdasilva}@inf.ufrgs.br

ABSTRACT

In this paper we present an agent-based simulation of pedestrian dynamics based on cellular automata models. Differently from the cellular automata, our model represents different pedestrian characteristics: gender, speed, room geometry knowledge, and herding and obstacle avoidance behavior. We study how different room geometries, different pedestrian groups sizes and characteristics influence the pedestrian dynamics and the macroscopic behavior of the system. With this agent-based approach we expect to obtain more realistic results than the cases where the pedestrians are uniformly modeled. Our analysis indicates that pedestrian groups with different features contribute in different ways to the macroscopic behavior.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence
Intelligent agents, Multiagent systems

General Terms

Human Factors

Keywords

Agent-Based Modelling, Pedestrian Dynamics Simulation

1. INTRODUCTION

Modeling and simulating pedestrian dynamics is more complex than simulating vehicles. Computer simulations of pedestrian dynamics are able to show a high number of characteristics that exist in the real world and contribute for a better understanding of basic pedestrian traffic principles. This way, researchers can explore the effects of new traffic rules, changes in rooms geometry and exit paths avoiding dangerous and expensive experiments.

*Author supported by CNPq

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

In classical models such as continuous and cellular automata all pedestrians are modeled uniformly, disregarding notable pedestrian differences. In this paper we propose an agent-based modelling that represents different pedestrian characteristics: gender, speed, room geometry knowledge, herding behavior, obstacle avoidance behavior, etc.

2. RELATED WORK

The three classes of models for pedestrian dynamics – continuous, agent-based, and cellular automata based – represent important pedestrian dynamics features and each has its own characteristics, as well as advantages and disadvantages. In **continuous models** [2], pedestrians behavior is represented by differential equations describing their movement and pedestrian are treated as particles affected by repulsion and attraction forces. This approach is successful when simulating pedestrian dynamics and space-time self-organized patterns, such as lane formations and oscillations on bottlenecks. Regarding **cellular automata models** (CA), in [1] pedestrians have different speeds and are represented by an automata aiming to maximize their frontal movement. In the model by Schadschneider and coworkers (see chapters of Burstedde et all. and Schadschneider in [5]), pedestrians move influenced by a pheromone trail. In **agent-based models** [4] pedestrians are simulated as agents that seek to minimize their mental stress improving their comfort while walking. This last model is the closest to ours in the sense that it is also agent based. However it is less focused on dynamics and more on stress and comfort of the agents, an issue particular to their scenario.

Next we briefly review features such as lane formation, attractive and repulsive effects and make a comment on implementation issues of those models.

Pedestrian flow is more efficient when pedestrians are organized in lanes. Besides, **lane formation** does not arise from the pedestrian initial spatial configuration but is a consequence of pedestrian interactions. Pedestrian dynamics space-time patterns appear due to the non-linear interactions among them. These interactions are caused by model's **attractive and repulsive effects**. Therefore these effects are important to pedestrian dynamics simulation. Attractive and repulsive effects are present in [2]. Pedestrians are attracted to their destination, other pedestrians, or objects. They maintain a certain distance from other pedestrians and objects to avoid collisions. This repulsive effect depends on the pedestrian density and desired speed. The model proposed in [4] has attractive and repulsive effects, represented in the agent's objective of minimizing the *destination*

stress (*D-stress*), that is the mental stress perceived when the agent cannot go through the shortest way to his destination. Repulsive effect is represented in the agent's objective of minimizing the *pedestrian stress* (*P-stress*), that is the mental stress perceived by the agent from others pedestrians while walking. The model proposed by Adler and Blue represents in the CA rules only attractive effects, but no repulsive effect is simulated. The CA model from Schadschneider and coworkers represents in the *dynamic floor field* and in the *static floor field* attractive effects, however no repulsive effect is modeled.

Regarding **performance**, in continuous models, each pedestrian must be compared with all others in order to perform the attractive and repulsive forces calculation, representing a large effort. In CA models, as they are discrete in time, space, and state variables, plus pedestrian interactions only occur in the local neighborhood, this makes them good for high performance computer simulations such as evacuations of stadiums with thousands of pedestrians. In the model by Osaragi, each agent cannot be too complex, because if it takes too much time to deliberate. Thus, in a simulation with hundreds or thousands of pedestrians the simulation time would be excessively long.

3. MODEL DEFINITION AND DYNAMICS

Our model is based on a CA but represents different pedestrian characteristics. Several underlying concepts are: Preferences Matrix, Floor Field, Sensitivity Coefficients, Speed, Hindrance Coefficient, Transition Probabilities and Collision Resolution.

Each pedestrian Ω has a **preference matrix** M^{Ω} which represents the pedestrian probability of movement in all directions: $\{M^{\Omega}_{i-1,j}; M^{\Omega}_{i,j-1}; M^{\Omega}_{i,j}; M^{\Omega}_{i,j+1}; M^{\Omega}_{i+1,j}\} \in [0,1]$. $M^{\Omega}_{i,j}$ represents pedestrian probability of remaining in his current position. The preferences matrix may be calculated, for example, by a route selection algorithm.

Attractive and repulsive effects are important to reproduce self-organized collective effects in pedestrian dynamics. We avoid explicitly modeling attractive long ranged interactions. Instead, we use the *floor field* concept. Through the *dynamic*, *static* and *repulsion floor fields*, we can simulate pedestrian interactions and attractive geometry effects in a simple and unified way.

The **dynamic floor field** represents the virtual path created by pedestrians while they walk. It models the long ranged attractive interaction among pedestrians. The dynamic field ($D_{ij} = 0$) is zero for all cells (i,j) in the grid at the initial time ($t = 0$). Whenever a pedestrian moves from one cell (i,j) to one of the neighboring cells, the dynamic floor field at cell (i,j) is increased by one: $D_{ij} = D_{ij} + 1$. Besides, the dynamic floor field is time dependent. It has diffusion parameter $\alpha \in [0,1]$ and decay parameter $\delta \in [0,1]$.

The **static field** models space regions that have a higher attractive effect, such as exit doors. It does not change over time, even with pedestrian movement.

The values for the static floor field $S_{i,j}$ used in this paper are calculated as in [3]. The static field has the maximum value at the door cells. It decreases with distance and is zero for the cell farthest away from the door.

The explicit values of $S_{i,j}$ at cell (i,j) is $S_{i,j} = MD - DIST_{i,j}$ with MD being the maximum distance from any cell to a door and $DIST_{i,j}$ being the minimum distance from cell (i,j) to the nearest door.

A pedestrian feels uncomfortable when s/he walks in a strange person's direction. This repulsive effect keeps a pedestrian from getting too close to another person. The original model from Schadschneider and coworkers does not take such repulsive effects into account. The **repulsion field** is our contribution to model the repulsive short ranged interaction among pedestrians. Each pedestrian creates a individual repulsion field. The repulsion field has its maximum value at $E1$ and obeys the relation $E1 > E2$. The resulting repulsion field at position R_{ij} is the sum from all individuals repulsion fields all over the grid and has the following domain: $R_{ij} \in [0, \infty[$.

Each individual in a crowd has a different reaction to long ranged interactions (dynamic floor field and static floor field) and short ranged interactions (repulsive floor field). The model represents each pedestrian in a different way using **sensitivity coefficients**. For each floor field, a sensitivity coefficient is created: $K_s \in [0, \infty[$ is a static floor field sensitivity coefficient; $K_d \in [0, \infty[$ is a dynamic floor field sensitivity coefficient; and $K_r \in [0, \infty[$ is a repulsion field sensitivity coefficient.

Through the sensitivity coefficients, it is possible to model different characteristics such as herding behavior, knowledge about room geometry and obstacle avoidance behavior. For example, a large K_s implies a motion toward the exit via the shortest path. For decreasing K_s a pedestrian performs a random walk and just find the door by chance. This is important for simulations on a dark or smoke-filled room where pedestrians do not have full knowledge about the exit location. The dynamic floor field coefficient K_d controls the herding behavior. A large K_d implies a strong herding behavior, such as the one observed in the panic situation. A full explanation about K_s and K_d may be found in [3]. The repulsion field coefficient K_r controls the obstacle avoidance behavior. A large K_r implies a pedestrian strongly avoidance other pedestrians.

Pedestrians can also have different **speeds**. Differently from [1], the speed values and speed probability distribution are based on real empiric observations about men's and women's speeds. We first generate a Gaussian distribution (mean 1.62 and 1.38 m/s and variance 0.22 and 0.17 for men and women respectively).

During a panic situation, objects that usually hinder the way may be jumped. Chairs, auditorium seats, and other minor objects are common examples. Other models do not take this effect in account during panic simulations. We introduce a **hindrance coefficient** $T_{ij} \in [0,1]$ to represent the possibility of a pedestrian jumping over an obstacle. An obstacle is fully not passable if $T_{ij} = 1$. For decreasing values of T_{ij} it is easier to jump the obstacle.

The **transition probability** P_{ij} to (i,j) for the pedestrian Ω depends on all other concepts presented. With these factors we propose a modified transition probability which is different for distinct situations: normal (Equation 1) and under panic (Equation 2). These formulas are modifications of those proposed by Schadschneider and colleagues and include the repulsion field $exp(K_r \cdot R_{ij})$ to represent the repulsive short ranged interaction among pedestrians, and the hindrance coefficient T_{ij} for panic scenarios.

$$P_{ij} = \frac{N \cdot M_{ij} \cdot (1 - n_{ij}) \cdot exp(K_d \cdot D_{ij}) \cdot exp(K_s \cdot S_{ij})}{exp(K_r \cdot R_{ij})} \quad (1)$$

$$P_{ij} = \frac{N \cdot M_{ij} \cdot (1 - n_{ij}) \cdot exp(K_d \cdot D_{ij}) \cdot exp(K_s \cdot S_{ij}) \cdot (1 - T_{ij})}{exp(K_r \cdot R_{ij})} \quad (2)$$

where N is the normalization factor to ensure that the sum over all the matrix cells is 1 ($\sum_i \sum_j P_{ij} = 1$).

A **collision** occurs when two or more pedestrian choose to move to the same cell. The conflict is solved according to the relative probabilities which each pedestrian chooses the target cell (i, j).

4. SIMULATIONS

Due to lack of space, here we do not show details of all simulations performed (lane formation in a corridor, variable door size evacuation simulation, room evacuation, and auditorium evacuation under panic). Instead, we concentrate on the parameters employed and the results achieved. Mostly, each cell represents 40 x 40 cm^2 and each iteration step is 1 second.

First, one experiment by Schadschneider and colleagues was reproduced for the sake of verification of the correctness of our model. It is the evacuation of a room of 25.2 m^2 or $63X63$ cells, with an exit of one cell in the middle of one wall. All 1116 pedestrians are initially distributed randomly and try to leave the room. It was verified that our results are close to the ones found by Kirchner.

Next, we have performed the simulation regarding lane formation on a corridor (30 X 15 cells) with 65 pedestrians (50% men and 50% women). The dynamical floor field diffusion is $\alpha = 0.05$ and the decay is $\delta = 0.15$. With these parameters, the dynamical field lasts longer and has little diffusion. The repulsion field is: $E1 = 0.6$, $E2 = 0.2$. All pedestrians have herding behavior ($K_d = 1$) and sensitivity to the repulsion field ($K_r = 1$). The kind of experiment can be assessed from the qualitative point of view: our model is able to achieve lane formation (Fig. 1) and pedestrian flow is more organized than a random pedestrian distribution.

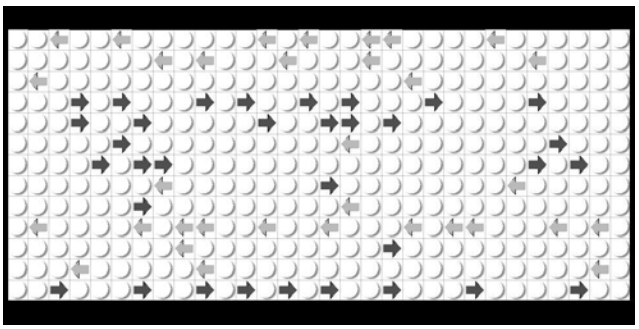


Figure 1: Lane formation

We have then simulated the evacuation of a room, 84.64 m^2 or 23 X 23 cells, varying the door size from 1 to 21 cells. The parameters are $\alpha = 0.05$, $\delta = 0.15$, $E1 = 0.6$, and $E2 = 0.2$. All 132 pedestrians (50% men and 50% women) are assumed to be in the *optimal regime* (they have a combination of knowledge about room geometry ($K_s = 1.3$) and herding behavior ($K_d = 0.5$) that minimizes the evacuation time). The repulsion floor field sensitivity coefficient is $K_r = 1$. We have measured the average pedestrian flow, the evacuation time, and the escape probability. The former increases with the door size (notice that this is not obvious: there are reports showing that pedestrian flow decreases for door size greater than 1 and smaller than 4, when there are conflicts near near the exit door).

In the last experiment, we have simulated the evacuation of an auditorium, 27 X 43 cells, in a panic scenario. The auditorium's chairs have hindrance coefficient $T_{i,j} = 0.9$ and the transition probability in panic scenario is as described in Equation 2. At $t = 0$ there are 237 individuals (50% men and 50% women) distributed in the auditorium chairs. Individuals are in the *disordered regime* after the announcement of a need to evacuate the auditorium quickly. According to [3], pedestrians in this regime are under panic and have a strong herding behavior ($K_s = 0.8$, $K_d = 2$ and $K_r = 1$). The dynamical floor field diffusion is $\alpha = 0.05$ and the decay is $\delta = 0.15$. Pedestrian flow increases at a maximum near time step 50 and begin to decrease near time step 125.

5. CONCLUSIONS AND FUTURE WORK

In continuous and cellular automata models all pedestrians are modeled uniformly, disregarding pedestrian differences. Differently from these, we proposed an agent-based model that represents different pedestrian characteristics: gender, speed, room geometry knowledge, herding behavior, and obstacle avoidance behavior. The motivation to use agent-based approaches is to achieve more realistic results than those from cellular automata and continuous models where the pedestrians are uniformly modeled. Our results indicate that when the pedestrian population is heterogeneous, the macroscopic behavior is different from a uniformly modeled population of pedestrians. Analysis indicate that pedestrian groups with different features (such as the group in the *optimal regime* and the group in the *normal regime*) have different escape probabilities and contribute in different ways to the macroscopic behavior. For future work, we intend to perform simulations with more than two pedestrian groups to investigate how the pedestrian dynamics behavior is affected by mixed classes of pedestrian.

6. REFERENCES

[1] J. L. Adler and V. J. Blue. Cellular automata model of emergent collective bi-directional pedestrian dynamics. In J. S. McCaskill, N. H. Packard, and S. Rasmussen, editors, *Artificial Life VII*, volume VII, pages 437–445. MIT Press, 2000.

[2] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51(5):4282–4286, 1995.

[3] A. Kirchner and A. Schadschneider. Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. *Physica A: Statistical Mechanics and its Applications*, 312:260–276, September 2002.

[4] T. Osaragi. Modeling of pedestrian behavior and its applications to spatial evaluation. In N. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 836–843, 2004.

[5] M. Schreckenberg and S. D. Sharma. *Pedestrian and Evacuation Dynamics*. Springer-Verlag New York, Germany, 2002.

Modeling Uncertain Domains with Polyagents

H. Van Dyke Parunak, Sven Brueckner
Altarum Institute
3520 Green Court
Suite 300
Ann Arbor, MI 48105 USA
+1 734 302 4600

{van.parunak, sven.brueckner}@altarum.org

ABSTRACT

Agents are a powerful tool for systems modeling. Instantiating each domain entity with an agent captures many aspects of system dynamics and interactions that other modeling techniques do not. However, an entity's agent can execute only one trajectory per run, and so does not capture the alternative trajectories accessible in the evolution of any realistic system. Averaging over multiple runs does not capture the range of individual interactions. We address these problems with a new modeling construct, the *polyagent*, which represents each entity with a single persistent *avatar* supported by a swarm of transient *ghosts*. Each ghost interacts with the ghosts of other avatars through digital pheromone fields, capturing a wide range of alternative trajectories in a single run of the system that can proceed faster than real time.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence– *multiagent systems.*

General Terms

Algorithms, Measurement, Experimentation.

Keywords

Multiple futures, uncertainty, BDI, swarm intelligence, digital pheromones, dynamics.

1. INTRODUCTION

The fundamental entity in an agent-based model is the agent, corresponding to a discrete entity in the domain being modeled, and the fundamental operator is the interaction among agent behaviors. The fundamental entity in an equation-based model [9] is some system observable, and the fundamental operator is the evolution of that observable. Agent-based models often map more naturally to the structure of a problem than do equation-based models, have representations that are more naturally constructed and explored, and provide more realistic results [8], but have a shortcoming. The observables in an equation-based model are often averages across many agents, and so implicitly capture the range of agent variation (although at an aggregate level). The software agent representing an entity in an agent-based model can execute only one trajectory in each run of the system, and so does not capture the alternative trajectories accessible to the entity in the evolution of any realistic system. Good modeling practice calls for averaging over multiple runs, but this approach still does not capture the range of individual interactions involved.

A new construct, the *polyagent*, represents each entity with a single persistent *avatar* and a swarm of transient *ghosts*. Each ghost interacts with ghosts of other avatars through digital pheromone fields, capturing a wide range of alternative trajectories in a single run that can proceed faster than real time for many reasonable domains.We have used this approach in a number of systems that we have constructed over the years. The purpose of this paper is to articulate the polyagent as a explicit modeling construct and provide some guidance concerning its use.

Section 2 reviews the challenge of adequate sampling in multi-agent modeling of complex domains. Section 3 proposes a new modeling construct, the polyagent, as an answer to this challenge. Section 4 discusses systems we have constructed that can be understood as instances of polyagents. Section 5 concludes.

2. THE CHALLENGE OF MODELING MULTI-AGENT INTERACTIONS

Imagine $n + 1$ entities in discrete time. At each step, each entity interacts with one of the other n entities. Thus at time t its interaction history $h(t) \in n^t$. Its behavior is a function of $h(t)$. This toy model generalizes many interesting domains, including predator-prey systems, combat models, innovation and diffusion of ideas, and disease propagation.

An equation-based model typically reasons at the level of aggregate or average observables across the agent population (e.g., in the predator-prey example, predator population, prey population, average predator energy level, or average prey energy level, all as functions of time). No attempt is made to model the trajectory of an individual entity.

An agent-based model must explicitly describe the trajectory of each agent. In our general model, during a run of length τ, each entity will experience one out of n^τ possible histories. The population of $n + 1$ entities will sample $n + 1$ of these possible histories. The length of a run is often orders of magnitude larger than the number of modeled entities ($\tau >> n$).

Good practice recommends that such models be run many times, with different random seeds, to explore the range of possible outcomes. Each such run only samples one set of possible interactions. For large populations and scenarios that permit multiple agent interactions, the number of runs needed to sample the pos-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

sible alternative interactions thoroughly can quickly become prohibitive, and the portion of the space of possible entity histories actually sampled by a single run is vanishingly small. We would need a number of runs on the order of τ to generate a meaningful sample. Polyagents can capture the outcome of multiple possible interactions in a single run.

3. THE POLYAGENT CONSTRUCT

A polyagent represents each domain entity with a single *avatar* that manages its correspondence with the domain, and a swarm of *ghosts* that explore its alternative behaviors.

The *avatar* persists as long as its entity is active, and maintains state information reflecting its entity's state. Its computational mechanisms may range from simple stigmergic coordination to sophisticated BDI reasoning. Each avatar generates a stream of *ghosts*. Ghosts have limited lifetime, dying off after a fixed period of time or after some defined event to make room for more ghosts. The avatar controls the rate of generation of its ghosts, but typically has several ghosts concurrently active.

Ghosts explore alternative possible behaviors for the avatar that generated them. They are computationally very simple, and interact with one another through a digital pheromone field, a vector of scalar values that is a function of both location and time. That is, each ghost chooses its actions stochastically based on a weighted function of the strengths of the various pheromones in its immediate vicinity, and deposits its own pheromone to record its presence. A ghost's "program" consists of the vector of weights defining its sensitivity to various pheromones.

Multiple ghosts multiply the number of interactions that a single run of the system can explore. Instead of one trajectory for each avatar, we now have one trajectory for each ghost. If each avatar has k concurrent ghosts, we explore k trajectories concurrently. But the multiplication is in fact greater than this.

The environment maintains the digital pheromone field, supporting three functions [1]: It *aggregates* deposits from individual agents, fusing information across multiple agents and through time. Aggregation of pheromones enables each ghost to interact with multiple other ghosts at the same time through their pheromone field, which summarizes their individual behaviors. It *evaporates* pheromones over time, an innovative alternative to traditional truth maintenance in artificial intelligence. Traditionally, knowledge bases remember everything they are told unless they have a reason to forget something. Detecting inconsistencies is NP-complete. Pheromones constantly forget everything they learn, unless it is continually reinforced, so inconsistencies automatically remove themselves. It *propagates* pheromones to nearby places, disseminating information for access by nearby agents.

Propagation enables each ghost to sense the presence of multiple other agents. If n avatars deposit pheromones, each ghost's actions are influenced by up to n other agents (depending on the propagation radius), so that we are exploring in effect nk interactions for each entity, or n^2k interactions overall. If individual ghosts deposit pheromones, the number of interactions being explored is even greater, on the order of k^n. The interactions are not played out in the detail they would be in a conventional multi-agent model. But empirically, they are reflected with a fidelity that is adequate for the problems we have addressed.

The avatar can *activate* its ghosts when it wants to explore alternative possible futures, modulating their generation rate to determine the number of alternatives explored and initializing the ghosts' weight vectors to reflect the breadth of alternatives it wishes to explore. It can *evolve* its ghosts to learn the best parameters for a given situation, monitoring the performance of existing ghosts against some fitness parameter, and then breeding the most successful to determine the parameters of the next generation. It can *review* its ghosts' behavior to estimate the direction and likely variation in how its own future behavior.

The polyagent bears comparison with functionally decomposed multi-agent architectures, population-based search, fictitious play, and two previous uses of the term "polyagent" (**Error! Reference source not found.**).

Our polyagent is distinct from the common use of multiple agents to model different functions of a single domain entity. In a polyagent, each ghost has the same function: to explore one possible behavior of the domain entity. The plurality among ghosts provides, not functional decomposition, but a range of estimates of alternative likely behaviors.

Many forms of evolutionary computation [3] allow multiple representatives of a single entity to execute concurrently, to evaluate their fitness. In these systems, each agent samples only one possible series of interactions with other entities in the domain. Pheromone-based coordination in the polyagent construct permits each ghost to adjust its behavior based on many possible alternative behaviors of other entities in the domain.

Similarly, the multiple behaviors contemplated in fictitious play [5] take place against a static model of the rest of the world.

Like the polyagent, ant-colony optimization (ACO) [2] uses pheromones to integrate the experiences of parallel searchers. The novel contribution of the polyagent is the avatar as a single point of contact for the searchers representing a single domain entity.

"Polyagent" is a neologism to capture the concept of several software agents that collectively represent a domain entity and its alternative behaviors. The term is used in two other contexts that should not lead to any confusion. In medicine, "polyagent therapy" uses multiple treatment agents (notably, multiple drugs combined in chemotherapy). K. Kijima [4] uses the term in analyzing

Table 1: The Polyagent and Other Technologies	Multiple agents / entity	Avatar/Ghost dualism	Parallel search of alternate behaviors	Parallel search of multiple interactions
Polyagent	X	X	X	X
Functional multiple agents	X			
Evolution	X		X	
Fictitious play	X		X	
ACO	X		X	X
Kijima's polyagents				
Polyagent therapies				

Table 2: Applications of Polyagents

Domain	Avatar	Ghosts sample ...
Factory Scheduling [1]	Workpiece	Alternative routings
Path Planning for Robotic Vehicles [6]	Vehicle	Alternative paths through battlefield
Characterizing & Predicting Agent Behavior [7]	Warrior	Alternative personalities and interactions with the environment

multiple decision-makers. In Kijima's approach, the term "polyagent" makes sense only as a description of a system, and does not describe a single agent. In our approach, it makes sense to talk about a single modeling construct as "a polyagent."

4. EXAMPLES OF POLYAGENTS

We discovered the value of polyagents as a modeling construct by reflecting on several applications that we have constructed and observing their common features. The full paper discusses these applications, which are summarized in Table 2.

Our current use of polyagents involves a fair amount of art. There is much room for theoretical work to make the technique more rigorous. Perhaps the most challenging question has to do with the legitimacy of merging pheromones of multiple ghosts representing agent A into a single field that then guides the behavior of agent B. This process enables B to explore concurrently its possible interactions with multiple alternative realizations of A, but the formal legitimacy of this process is open to question.

Pheromone field strength depends on the frequency with which agents visit various locations. It may be viewed as a probability field describing (for instance) the likelihood of finding an agent of a given type at a given location. If those agents are ghosts representing alternative futures of an entity's trajectory, the probability field represents the likelihood of different future states. An analogous situation arises in quantum mechanics (Table 3), which we are exploring as an intellectual and formal model for engineering polyagent systems and interpreting their behavior.

5. CONCLUSION

One strength of agent-based over equation-based modeling is that it captures the details of the trajectory experienced by each entity. In uncertain domains, this strength is also a weakness, because any single set of entity trajectories is only a sample from a space of possible trajectories that is much too large to explore thoroughly by repeated experiments. Polyagents can sample multiple interactions in a single simulation run. An avatar mediates between the entity being modeled and a swarm of ghosts that sample alternative possible trajectories. The avatar may employ sophisticated cognitive reasoning, but the ghosts interact entirely through digital pheromone fields that they deposit and sense in their shared environment. The avatar modulates the continual generation of ghosts, and interprets their aggregate behavior to provide an estimate of its entity's likely behavior.

Table 3: Parallels between Quantum Physics and Polyagents

Quantum Physics	Polyagents
Duality between (single, localized) particle and (distributed) wave function	Duality between (single, localized) avatar and (distributed) swarm of ghosts
Interactions among wave functions' amplitude fields model interactions among particles	Ghosts' pheromone fields can be interpreted as probability densities that model interactions of agents
Wave function captures a range of possible behaviors	Swarm of ghosts captures a range of possible behaviors
Observation collapses the wave function to a single behavior	Avatar interprets the aggregate behavior of the ghosts and yields a single prediction of behavior

We have applied polyagents to several problems. Empirically, the construct functions very well, but it opens a wide range of questions about the relation between multiple ghosts interacting with a pheromone field that represents multiple alternative realizations of other entities. The system exhibits several parallels with quantum physics that suggest the latter discipline may be a useful basis for a more formal model of polyagents.

The full version of this paper is available at http://www.altarum.net/~vparunak/AAMAS06Polyagents.pdf .

6. ACKNOWLEDGEMENTS

This material is based in part upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract Nos. F3062-99-C-0202 and NBCHC040153. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DARPA or the Department of Interior-National Business Center (DOI-NBC). Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

7. REFERENCES

[1] Brueckner, S. *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. Thesis at Humboldt University Berlin, Department of Computer Science, 2000.

[2] Dorigo, M. and Stuetzle, T. *Ant Colony Optimization*. Cambridge, MA, MIT Press, 2004.

[3] Jacob, C. *Illustrating Evolutionary Computation With Mathematica*. San Francisco, Morgan Kaufmann, 2001.

[4] Kijima, K. Why Stratification of Networks Emerges in Innovative Society: Intelligent Poly-Agent Systems Approach. *Computational and Mathematical Organization Theory*, *7*, 1 (June): 2001, 45-62.

[5] Lambert, T. J., III, Epelman, M. A., and Smith, R. L. A Fictitious Play Approach to Large-Scale Optimization. *Operations Research*, *53*, 3 (May-June): 2005.

[6] Parunak, H. V. D., Brueckner, S., and Sauter, J. Digital Pheromones for Coordination of Unmanned Vehicles. In *Proceedings of Workshop on Environments for Multi-Agent Systems (E4MAS 2004)*, Springer, 2004, 246-263.

[7] Parunak, H. V. D. and Brueckner, S. A. Extrapolation of the Opponent's Past Behaviors. In A. Kott and W. McEneany, Editors, *Adversarial Reasoning: Computational Approaches to Reading the Opponent's Mind*, Chapman and Hall/CRC, 2006.

[8] Parunak, H. V. D., Savit, R., and Riolo, R. L. Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and Users' Guide. In *Proceedings of Multi-agent systems and Agent-based Simulation (MABS'98)*, Springer, 1998, 10-25.

[9] Sterman, J. *Business Dynamics*. New York, NY, McGraw-Hill, 2000.

TEAKS: Simulation of Human Performance at Work to Support Team Configuration

J. Martínez-Miranda
Dept. of Computer Eng. and Mathematics
Universitat Rovira I Virgili
Tarragona, SPAIN
(+34) 977 55 85 16
juan.martinez@urv.net[1]

A. Aldea
Department of Computing
Oxford Brookes University
Oxford, UK
(+44) 0 1865 484 516
aaldea@brookes.ac.uk

R. Bañares-Alcántara
Dept. of Eng. Science
The Oxford University
Oxford, UK
(+44) 0 1865 273 122
rene.banares@eng.ox.ac.uk

M. Alvarado
Mexican Institute of Petroleum
MEXICO, DF.
(+52) 55 917 65755
matiasa@imp.mx

ABSTRACT
The management of a complex engineering project is a difficult task that initially involves the division of the project into tasks; the selection of the right people; and the correct allocation of those tasks for the selected people. Team configuration process is typically performed by a manager based on his/her past experience and the available (though frequently scarce, uncertain and dynamic) information about the cognitive, personal and social characteristics of the potential team members. To support this decision-making process we propose TEAKS, a knowledge-based tool that given an initial team configuration and a set of tasks, simulates the most possible team performance. Its formal bases are Fuzzy Set Theory and Fuzzy Logic and it is implemented using Multi-Agent Systems technology in Java, JADE, JESS and FuzzyJESS. This tool was validated with an industrial project involving 23 team members and 23 tasks (ranging from task assignment to SAP administration, SQL programming and equipment connectivity).

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *coherence and coordination, multiagent systems.*

General Terms
Management, Performance, Experimentation, Human Factors.

Keywords
Agents and cognitive models, agent-based simulation and modelling, applications of autonomous agents and multi-agent systems, artificial social systems.

1. INTRODUCTION
When a new complex project starts, a project manager is put in charge of partitioning this project into tasks and selecting the people who will perform them. The success of a project is greatly due to the personal responsibility of each member, but also to an adequate communication, collaboration and co-operation between them [2]. In addition to social factors, the emotional state of a person plays a critical role in rational decision-making, perception, human interaction, and human intelligence [11]. The computational simulation of human intelligent behaviour has been one of the main research topics in Artificial Intelligence, and therefore, some of these techniques can be employed to simulate social human behaviour as well, particularly human social interaction and behaviour in a team. In this paper we present TEAKS (TEAm Knowledge-based Structuring), a fuzzy agent-based tool that can be used to simulate team-working performance.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8–12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005…$5.00.

2. RELATED WORK
The simulation of human behaviour has been one of the most interesting areas of research and application since early 70's [6]. More recently, the computational simulation of work groups to predict group performance has started to emerge. For instance, each worker can be modelled as a self-running process that controls a worker entity through data and rules embedded in each process as is proposed in [1].

Given the characteristics of the Multi-Agent Systems [13], they can be seen as a very useful tool for modelling human behaviour and in particular social behaviour. The agent-based approach enhanced the potentialities of computer simulation as a tool for theorising about social scientific issues, since it facilitates the modelling of artificial societies of autonomous intelligent agents [5]. In [10] an agent-based model to represent human behaviour is presented where the agents can be characterised by a set of attributes and their behaviour can be described by a set of simple decision heuristics. Additionally, in [12] an application-independent agent architecture called PECS (Physics, Emotion, Cognition, Social Status) to simulate human behaviour within groups is proposed.

Fuzzy logic can also be considered to represent some human characteristics. In [7] a fuzzy emotional agent for decision-making in a mobile robot is described. Another interesting work is presented in [9] where fuzzy agents with dynamic personalities are proposed. Our TEAKS model is based on fuzzy agents; fuzzy parameters represent human characteristics and fuzzy rules model

[1] The first author is a Ph.D. student of the Artificial Intelligence Program at the Technical University of Catalonia (SPAIN).

the human performance within a work team. The agent architecture is based on the PECS model.

3. TEAKS PARAMETERS TO SIMULATE HUMAN PERFORMANCE

To simulate the human behaviour at work, those relevant human characteristics that affect a person performance at work should be identified. In this research we have decided to incorporate the following characteristics to our team members: personality trends, a set of basic emotions, social characteristics and cognitive capabilities.

3.1 Personality Trends

There are several studies that show the importance of the personality traits in the job performance [4]. We have taken into account the CLEAVER technique to identify behaviour styles in front of different work situations [3]. The CLEAVER technique is applied to the candidates through several psychometric questions. The result of this questionnaire is a numeric value between 1 and 99 for each of the following personality trend parameters (DISC): **Drive, Influence, Steadiness** and **Compliance:** capability to develop the work following establish rules or procedures.

3.2 Emotional State

Emotions have been proven to be an important factor at work place [8]. Given the importance of the emotions in the human work behaviour, the critical decision is to select the set of basic emotions to model them into the agents. From the extended psychology classification of the basic emotions, we select a little set of four basic emotions to model the agents' emotional state at work. Two of them are positive emotions and the other two have a negative influence over the people's performance: **Positive emotions**: ***desire*** and ***interest*** of the person to develop a specific task in a given moment. **Negative emotions**: ***disgust*** and ***anxiety*** generated by a given task in specific moment.

3.3 Social Characteristics

The characteristics of human relations in groups and teams (such as competence, trust, co-operation, among others) are the main topic of research in areas such as social psychology, social sciences and organisational behaviour. Modelling all these characteristics is out of the scope of this research, but nevertheless, we considered a small set of social characteristics: **introverted / extroverted,** and **prefers to work alone / in team:** when a person is working, he/she may prefers to work alone to develop a specific task(s) even though this person can be extroverted, and vice versa.

3.4 Cognitive Capabilities

Human cognitive capabilities involve several brain processes such as learning and memory among others. Modelling these brain processes and their interactions to generate an intelligent behaviour has been one of the main goals of Artificial Intelligence and it is, as well as human social characteristics, out of the scope of this research. The cognitive capabilities of a person were taken into account in our model by considering the person degree of expertise in a particular domain: **Project Manager:** a person in charge of managing the project and assigning tasks to the other team-members. **Co-ordinator:** person in charge of specialised tasks, re-configuration of tasks and re-location of resources. **Specialist:** person in charge of complex and specialised tasks. **Technician:** Person who can deal with technical and no-specialised tasks. **Assistant:** Person in charge of not complex, routine and repetitive tasks. In addition of each person's role, every team-member has two other independent-role parameters to represent his/her cognitive capabilities: **experience level** and **creativity level**.

3.5 Performance Parameters

The resulting team members' performance is calculated by taking into account its internal characteristics, the interaction with its assigned task and the interaction with the rest of team-members. We have proposed six parameters to evaluate the expected final performance of each team member: **goals achievement**, **timeliness** and **quality** of the performed task, **team collaboration level**, **individual contribution level** and **required supervision level**. These six parameters had been taken from the existent formats used by project leaders of the Mexican Petroleum Institute (IMP), (the institution which provides us the case study described in section 7).

3.6 The Project

In the TEAKS model a project is represented by its division into tasks, every task is then represented by the following parameters: **number of participants** in the task, **estimated duration** (measured in days), **sequence** (two or more tasks could be executed sequentially or in parallel), **priority within the project**, **deadline**, **cost**, **quality**, **application domain**, **task description**, **level of difficulty** and **type** (required specialisation level).

4. TEAKS FUZZY AGENTS

The behaviour of a person, i.e. the simulation of his/her performance within the project individually and as a team-member, is generated by the combination of all the previously mentioned characteristics. It is difficult to accept that a numerical value could be used to measure the intensity of either human emotion, experience or creativity level. However, we frequently use qualitative attributes to describe properties of human moods, features and emotions. Following this fact, we propose that it is more natural and intuitive to associate qualitative values to the person's internal parameters.

4.1 Fuzzy Rules

Once the fuzzy sets for the agent and task parameters have been defined, the next step is to build the fuzzy rules that are required to simulate the agent behaviour. Fuzzy rules are used to update the values in the internal state of the agents and the agents' performance.

An example of the fuzzy rules used in this process is shown below (where T1 represents a task and A1 an agent):

IF T1 presents a *high_delay* AND A1 has a *driver* personality with *high_intensity* THEN

The *desire* emotion will have a *high_increase*
The *interest* emotion will have a *high_increase*
The *disgust* emotion wills *stay_equal*
The *anxiety* emotion will have a *low_increase*

5. TEAKS VALIDATION

TEAKS was implemented over the JADE Multi-Agent framework (see http://jade.tilab.com/). JESS (see http://herzberg.ca.sandia.gov/jess/) and FuzzyJess (http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJToolkit2.html) rule engines were used to implement the fuzzy sets and the reasoning mechanism required to simulate the agent performance. TEAKS initial test and validation was performed at the Mexican Institute of Petroleum (IMP, www.imp.mx), a research and technological development centre. At the IMP, like in other organisations, the configuration of the project team is fundamental to success in large and complex projects.

An Information Technology project was selected for validation due to the availability of the project manager, the accessibility to confidential documents that evaluate the performance of some of the people involved in the project; and the recent finalisation of that project. In particular, this case study was a project on the development and implementation of a Geographical Information System (GIS) for internal use at the IMP. This project was divided into 23 tasks and assigned (coincidentally) to 23 team-members (1 project manager, 1 coordinator, 15 specialists and 6 technicians). The results generated by our simulation tool were compared with the evaluation reports provided by the project manager.

6. DISCUSSION

The people (and agents) performance is strongly affected by the values of the CLEAVER parameters. Low values for these parameters generate low values in most of the other evaluation parameters. On the other hand, high values of CLEAVER parameters and high values in the parameters related to the social characteristics give rise to a better performance. The difficult of the task parameter influences the team member behaviour. Agent 11 was in charge of 2 tasks (without team-mates in both tasks), and one of these tasks was of medium difficulty. The result of its behaviour for each of the six parameters was better for the *low difficulty task* than for the *medium difficulty task*.

It can be argued that most of the results depend on the rules configuration, despite that, in most cases, the simulation and the GIS/IMP project results were similar. Nevertheless, we must take into account the following:

- We could not access the information for all the team members involved in the project. Nevertheless, we had the opportunity to talk with the project manager and some of the people who worked in the project. Their comments were that, given the software limitations (absence of some of the policies of the institution, such as involvement of the client and manager services among others), the initial predictions can be considered useful and resemble a part of the reality.
- The project manager filled the evaluation documents where the results of the team-members performance are recorded. This evaluation is according to the project manager perception and, in many cases, he/she did not work directly with some of the team members.
- The personality type parameters for each agent were set according to the results of the application of the CLEAVER technique. Nevertheless, the values of parameters such as creativity and experience were configured according to the perception and experience of one of the project co-ordinators regarding his/her interaction with the other team members.
- We have compared the results for only one project. As a result of these initial results, the managers of IMP have shown a huge interest in applying this model for more projects. Once the new information becomes available, we plan to use it for further comparisons and validation (IMP plans to finish this job in the next months).

Even with this set of encouraging results of the initial validation, more tests with other case studies are obviously needed to enhance TEAKS and improve its validation.

7. REFERENCES

[1] Bernhardt W. and Schilling A. *Simulation of Group Work Processes in Manufacturing*. In Proceedings of the 1997 Winter Simulation Conference, pp 888-891.

[2] Biegler, L. T., Grossmann I. E. and Westerberg, A. *Systematic Methods of Chemical Process Design.* Prentice Hall International Series. In The Physical and Chemical Engineering Sciences. Chapt. I pp 1-21, 1997.

[3] Cleaver, P. *Tres Técnicas de Personalidad (Cleaver).* (in Spanish) Editor: Sociedad de Psicología Aplicada, Lugar: México, D.F., 2000

[4] Code, S. and Langdan-Fox, J. *Motivation, cognitions and traits: predicting occupational health, well-being and performance.* In Stress and Health Journal Vol. 17, Issue 3, pp. 159-174, 2001.

[5] Conte, R., Gilbert, N. and Sichman, J. S. *MAS and Social Simulation: A Suitable Commitment*. In Lecture Notes in Computer Science Vol. 1534, 1-9 1998.

[6] Dutton, J. M. and Starbuck, W. H. *Computer Simulation of Human Behaviour.* John Willey (ed.), New York, NY. 1971.

[7] El-Nasr, M. S. and Skubic, M. *A Fuzzy Emotional Agent for Decision-Making in a Mobile Robot*. In Proceedings of the 1998 IEEE International Conference on Fuzzy Systems, May, 1998.

[8] Fisher, C. D. *Mood and emotions while working: missing pieces of job satisfaction?* In Journal of Organisational Behaviour. Vol. 21, Issue 2, March 2000, pp. 185-202.

[9] Ghasem-Aghaee, N. and Ören, T. I. *Towards Fuzzy Agents with Dynamic Personality for Human Behaviour Simulation*. Proc. of Summer Computer Simulation Conference, 2003, pp. 3-10.

[10] Pahl-Wost, C. and Ebenhöh, E. *Heuristics to characterise human behaviour in agent based models.* In Pahl-Wostl, C., Schmidt, S. and Jakeman, T. (eds) iEMSs 2004, June 2004.

[11] Picard, R.W. *Affective Computing* MIT. Press: Cambridge, MA. 1997.

[12] Urban, Ch. and Schmidt, B. *Agent-Based Modelling of Human Behaviour*. In Emotional and Intelligent II - The Tangled Knot of Social Cognition, AAAI Fall Symposium Series, 2000.

[13] Wooldridge M. and Jennings N. R. *Intelligent Agents: Theory and Practice.* The Knowledge Engineering Review, Vol. 10, No. 2, pp 115-152. 1995.

Modelling Dynamics of Cognitive Agents by Higher-Order Potentialities

Tibor Bosse[1]
[1]Vrije Universiteit Amsterdam
Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam
The Netherlands Tel. +31 20 598 77{50,63}
http://www.cs.vu.nl/~{tbosse, treur}

Jan Treur[1,2]
[2]Utrecht University
Department of Philosophy
Heidelberglaan 8, 3584 CS Utrecht
The Netherlands Tel. +31 6 15028916
{tbosse, treur}@cs.vu.nl

ABSTRACT
In the development of disciplines addressing dynamics, such as Mathematics and Physics, a major role was played by the assumption that processes can be modelled by introducing certain state properties (also called potentialities) that anticipate in which respect a next state will be different. The current paper is a first exploration of this perspective to analyse and model dynamics. Potentiality-based modelling subsumes quantitative, numerical modelling approaches, such as Dynamical Systems Theory (DST), and qualitative or symbolic modelling approaches to dynamics, such as BDI-modelling, and is applicable to model dynamics in a wide variety of (cognitive and noncognitive) disciplines. Thus, the modelling of dynamics of cognitive agents can be fully integrated with the modelling of other phenomena in Nature.

1. INTRODUCTION
Dynamics of the world shows itself by the occurrence of different world states, i.e., states at different points in time that differ in some of their state properties. In recent years, within Cognitive Science dynamics has been recognised and emphasised as a central issue in describing cognitive processes [9]. Van Gelder and Port [6] propose the Dynamical Systems Theory (DST) as a new paradigm that is better suited to the dynamic aspects of cognition than symbolic modelling approaches. However, as DST commits to the use of quantitative methods (differential and difference equations), it is often considered less suitable to model higher cognitive processes such as reasoning and language processing. DST is based on the notion of a state-determined system: a system in which properties of a given state fully determine the properties of future states; cf. [3], p. 25; [6], p. 6. This means that for state properties that are different in a future state, state properties in the given state can be found that somehow indicate or anticipate these differing (changed) properties in the future state. This idea closely relates to the concept of *potentiality* that goes back to Zeno and Aristotle [1]: if a potentiality *p* for a state property *a* occurs in a given state, then in a next state, property *a* will occur. A potentiality can be considered a kind of anticipatory state property: a state property anticipating the different state property (in the changed state).

The current paper contributes a first exploration of the notion of potentiality with respect to its use as a basis for modelling the dynamics of cognitive agents. A methodological perspective is described to analyse and model these dynamics in terms of potentialities. This modelling perspective subsumes both quantitative approaches (such as DST) and qualitative or symbolic approaches to modelling of dynamics. Given that the perspective subsumes DST, which since long has proved its value for quantifiable areas within a wide variety of disciplines, the scope of applicability of the proposed method includes disciplines such as Physics, Chemistry, Biology, and Economics.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

2. POTENTIALITIES
Given a particular state that just changed with respect to some of its state properties, it is natural to ask for an explanation of why these new state properties occurred. In a state-based approach, as a source for such an explanation, state properties found in the previous state form a primary candidate. A main question becomes how to determine for a certain state that it is going to change to a different state, and, more specifically, how to determine (on the basis of some of the state properties in the given state) those state properties for which the new state will differ from the given one. This poses the challenge to identify state properties occurring in a given state that in some way or the other indicate which of the (other) occurring state properties will be different in a subsequent state; by having these properties, the state anticipates the next state: anticipatory state properties. If such state properties are given, anticipation to change is somehow encoded in a state. The existence of such properties is the crucial factor for the validity of the assumptions underlying the Dynamical Systems Theory [3, 6]. It indicates the possibility to include concepts in the state ontology to conceptualise state properties that are useful to describe properties of changed states. Aristotle did introduce such a type of concept; he called it *potentiality* (to move), or movable. For example, the difference between an arrow at rest at position *P* and the snapshot of a moving arrow at time *t* at position *P* is that the former has no potentiality to be at some different position *P'*, whereas the latter has. This explains why at a next instant *t'* the former arrow is still where it was, at *P*, while the latter arrow is at *P'*.

3. HIGHER-ORDER POTENTIALITIES
The effect of a potentiality on a future state can be described by relating its occurrence in the present state to the occurrence of a certain state property in the future state. This specification can be viewed as the definition of what it is a potentiality for. A more complicated question is how to specify when (under which past and present circumstances) a potentiality itself will occur. For the case of empty space, where an object is assumed to have no interaction with other objects, a potentiality to change position is present because it was present at an earlier point in time and persisted until *t* (inertia of motion). However, if the potentiality in a new state is different from the earlier one, a question becomes why this is so. This leads to the question addressed in this section of how a *changed potentiality* can be explained. An answer to this question is the use of *higher-order potentialities*. The idea behind this is simple: to

obtain an explanation of changed state properties over time, potentialities were introduced. Potentialities are also changing over time. As they are considered state properties themselves, it would be reasonable to treat them just like any other state property that changes over time. This means that for a potentiality $p^{(1)}$ a so-called *second-order potentiality* $p^{(2)}$ is introduced to explain why $p^{(1)}$ may become changed over time. And of course this process can be repeated for $p^{(2)}$, and so on. This leads to an infinite sequence of *higher-order* potentialities, $p^{(1)}$, $p^{(2)}$, $p^{(3)}$, $p^{(4)}$, …, where for each natural number n the potentiality $p^{(n)}$ is called an *n-th-order potentiality*. Using such higher-order potentialities, the idea is the following:

- for a certain point in time *t0* the occurrence of a state property *a* is determined by the state at a previous time point *t1* < *t0*, in particular, by the occurrence of the first-order potentiality $p^{(1)}$ for *a* at that time point *t1*.
- the occurrence of the first-order potentiality $p^{(1)}$ at *t1* can be determined by the state at a time point *t2* < *t1*, in particular by the occurrence of its own potentiality which is the second-order potentiality $p^{(2)}$ for *a* at *t2*.
- and so on.

This process can be visualised as depicted in Figure 1. In this figure, the *x* at the lower-left can be ignored for the moment.

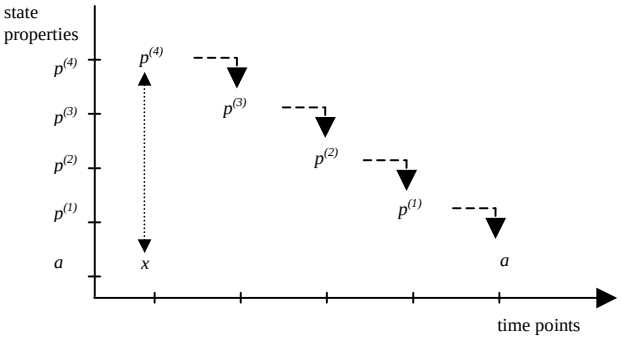


Figure 1. Dynamics based on higher order potentialities

This shows how the concept of potentiality to explain change of a certain basic state property *a* can take the form of a single entity, for example one number, to indicate what a changed property in an immediate subsequent state will be. Moreover, this can be extended by a large number of other (higher-order) entities, the occurrence of which can explain changed basic state properties *a* in future states.

Strange as the idea of an infinite number of higher-order potentialities may seem at first sight, in mathematical context (in particular in calculus) this has been worked out well (using infinite summations). For the discrete case, the idea of difference tables for functions has been developed. These differences play the role of relative potentialities: they indicate the next value not in an absolute sense, but in comparison to the current value. For the continuous case, higher-order potentialities have been formalised in the form of (higher-order) derivatives of a function. The (first-order) derivative of a function at a time point t gives an estimated measure of how the function will change its value in a next time point. The well-known Taylor series [12] for sufficiently smooth functions (at least infinitely often differentiable) shows how changes of the value from *t* to *t'* (within some given neighbourhood of *t*) depend on all (higher-order) derivatives:

$$f(t') = f(t) + \Sigma_k \; f^{(k)}(t)(t' - t)^k / k!$$

This expression shows how the (relative) potentiality at *t*, defined by the combination of all (infinitely many) higher-order potentialities, determines the changed state at the future time points *t'*.

In later times, successors of Aristotle have addressed the question how to further develop the phenomenon of dynamics (or change), in particular within Physics. They developed classical mechanics based on certain types of potentialities. For example, Descartes [4] took the product of mass and velocity of an object as an appropriate foundation for its potentiality to be in a changed position, or 'quantity of motion'. In modern physics this 'quantity of motion' concept is called *(linear) momentum*. This is one way in which a concept 'potentiality' (for change of position) was formalised in physics, thus providing one of the cornerstones of classical mechanics. Leibniz [7] formalised the notion of potentiality for changed position in a different manner by the product of mass and the square of velocity, what later got the form of the notion of kinetic energy, another cornerstone in Physics.

Also second-order potentialities played a crucial role in the development of Physics, for example, to obtain the concept 'force'. Newton uses the term 'impressed motive force' to express the change of motion in his second law: 'The change of motion is proportional to the motive force impressed; and is made in the direction of the right line in which that force is impressed' [8]. This law expresses that the concept of force used by Newton directly relates to change of motion. For (quantity of) motion he gives the same definition as Descartes, i.e., momentum. For an impressed force the following definition is given: 'An impressed force is an action exerted upon a body, in order to change its state, either of rest, or of uniform motion in a right line'. These definitions show that the concept 'force' used by Newton as an addition to state ontology can be given a definitional relationship to 'motion generated in a given time'. This 'motion generated in a given time' can be considered a second-order potentiality for the first-order potentiality 'motion' or momentum. So, within classical mechanics, after the concepts 'momentum' and 'kinetic energy' which were added to the state ontology as specific types of (first-order) potentiality, the concept 'force' can be considered a third property added to the state ontology, this time as a second-order potentiality. Using this state ontology, Newton (and also Leibniz) developed mathematical techniques of calculus, such as differentiation and integration.

4. REDUCERS

As indicated in the previous sections, the idea of explaining changed states in terms of higher-order potentialities can be very useful. However, when each potentiality can only be explained by introducing another (higher-order) potentiality, the approach risks ending up with an infinite chain of explanations. Then infinitary mathematical techniques such as infinite summations are needed, as shown in the Taylor series. To avoid this, one of the higher-order potentialities often can be connected somehow to lower level potentialities and/or basic state properties of the world. To this end, the notion of *reducer* of a higher-order potentiality is introduced: a combination of lower level potentialities and/or basic world state properties that is equivalent to this higher-order potentiality.

An example in the domain of Physics illustrates this notion. Analysing the motion of planets around the sun, Newton found out that they can only follow their orbit if a second-order potentiality (i.e., a force) is assumed, in the direction of the sun [8]. Newton calculated (using his calculus under development) in detail that this force was proportional to 1 divided by the square of the distance. For example, for an object in space with mass *m* approaching earth (with mass *M*), Newton's law of

gravitation for the motive force on the object is as follows (here x is the distance between the object and the earth, and c is a constant): $F = c\ mM/x^2$. Such a relation between the second-order potentiality force and basic state properties such as mass and distance shows how a higher-order potentiality can be reduced: in this case $c\ mM/x^2$ is a reducer for F.

To explain the idea of reduction, consider Figure 1 again. This time, pay attention to the additional (world) state property x, which is the reducer of $p^{(4)}$. For simplicity of the presentation we take an example where reduction only involves a basic state property, and no lower level potentialities.

5. POTENTIALITY-BASED MODELLING

Based on the idea of reduction, a generic method can be formulated to describe dynamics in terms of higher-order potentialities. The method involves a number of steps (also see Figure 1), roughly expressed as follows:

1. pick a basic world state property a of which the change over time needs to be modelled.
2. introduce a potentiality $p^{(1)}$ that explains the change over time of a.
3. can a reducer be found for the potentiality $p^{(n)}$ in the model that has the highest order?
   - 3a. *yes* -> introduce a reducer x that connects $p^{(n)}$ to a lower-level state property: in terms of world state properties and lower level potentialities; then end.
   - 3b. *no* -> introduce a potentiality $p^{(n+1)}$ that explains the change over time of $p^{(n)}$; then go to step 3.

To illustrate the idea of the method, suppose one wants to explain the fact that an object approaching the earth changes position over time (due to gravitation). In that case the basic world state property a would be something like 'object O is in position P' (step 1). Then, to explain the change over time of this state property, for the first-order potentiality $p^{(1)}$ a concept 'momentum' can be introduced (step 2). However, since no reducer of this potentiality can be found, a second-order potentiality $p^{(2)}$ is introduced to explain the change over time of a momentum, i.e. a concept 'force' (step 3b). Finally, a reducer is introduced for this second-order potentiality force: the expression $c*mM/x^2$ (step 3a). To summarise, this example shows how the change of a world state property in the domain of Physics can be explained and modelled in terms of higher-order potentialities.

6. DISCUSSION

In this paper, the perspective to analyse and model dynamics in terms of (first- and higher-order) potentialities has been explored. This potentiality-based perspective played a crucial role in the development of Physics. These examples refer to motion of non-living objects. Another type of motion to be explained is motion of a living being. Often used explanations of human (or animal) actions refer to internal mental states: e.g., an explanation of human behaviour from internal mental state properties such as desires. The desire to be at P' plays a role similar to that of the potentiality for being at P'. Indeed, this similarity can be traced back in history, for example, to Aristotle: 'Now we see that the living creature is moved by intellect, imagination, purpose, wish, and appetite. And all these are reducible to mind and desire.' [2], Part 6. Here properties of 'mind and desire' are mentioned as the source of motion of a living being. He shows how the occurrence of certain internal (mental) state properties (e.g., desires) within the living being entails or causes the occurrence of an action in the external world. This indicates hoe the potentiality-based perspective can be used to model BDI-agents. Thus, the use of potentialities in a sense unifies different types of approaches (both symbolic approaches, such as BDI-modelling [10], and nonsymbolic approaches, such as DST [9]) to cognitive modelling that are often seen as mutually exclusive. For every (basic) state property that is changing over time, a potentiality can be added to anticipate its changes. As this (first-order) potentiality may also change, a second-order potentiality may be introduced to anticipate these changes. In principle, this can be continued indefinitely. However, in many cases there is no need to go this far. Often, after a few steps a reduction can be made in the sense that a higher-order potentiality turns out to be equivalent to a combination of basic state properties and/or lower level potentialities. Within DST this is where a differential equation comes in, reducing an nth-order derivative to lower level properties: the assumption of reduction is another assumption underlying DST. All in all, the proposed modelling perspective places the modelling of cognitive phenomena not in an isolated position, but fully integrates it with the modelling of other phenomena in Nature.

Analysing dynamics of a process in terms of basic world state properties and higher-order potentialities opens up the possibility to create an executable *formal specification* (and subsequently, *simulation*) of the process. This is the subject of current research. In addition, an interesting challenge is to apply the presented method to affective agent states (such as emotions and moods). It is claimed by many authors (e.g., [11]) that affective states have the potential to control agent behaviour; see [5] for an example application. Therefore, an obvious next step is to model affective states as potentialities as well, thereby providing more formal evidence of the above claims. This option will be explored in future work.

REFERENCES

[1] Aristotle (350 BC). *Physica* (translated by R.P. Hardie and R.K. Gaye).

[2] Aristotle (350 BC). *De Motu Animalium* On the Motion of Animals (translated by A. S. L. Farquharson).

[3] Ashby, R. (1952). *Design for a Brain*. Chapman & Hall, London. Revised second edition, 1960.

[4] Descartes, R. (1633). The World or Treatise on Light. Withdrawn from publication, 1633. Also in: Descartes, R. (1998), Descartes: The World and Other Writings (edited by Stephen Gaukroger). Cambrige University Press. (translated by M.S. Mahoney).

[5] Gebhard, P. (2005). ALMA - A Layered Model of Affect. In: *Proc. of the 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'05*, pp. 29-36.

[6] Gelder, T.J. van, and Port, R.F. (1995). It's About Time: An Overview of the Dynamical Approach to Cognition. In: (Port and van Gelder, 1995), pp. 1-43.

[7] Leibniz, G.W. von (1956). *Philosophical Papers and Letters*, LeRoy E. Loemker, ed., Chicago: University of Chicago Press, 1956.

[8] Newton, I. (1729). The Mathematical Principles of Natural Philosophy; Newton's Principles of Natural Philosophy, Dawsons of Pall Mall, 1968.

[9] Port, R.F., and Gelder, T. van (eds.) (1995). Mind as Motion: Explorations in the Dynamics of Cognition. MIT Press, Cambridge, Mass

[10] Rao, A.S., Georgeff, M.P. (1991). Modeling rational agents within a BDI architecture. In: R. Fikes and E. Sandewall (eds.), *Proc. of the Second Conference on Knowledge Representation and Reasoning*, Morgan Kaufman, pp. 473-484.

[11] Scheutz, M., and Sloman, A. (2001). Affect and Agent Control: Experiments with Simple Affective States. In: *Proceedings of IAT 2001*, World Scientific Publishers.

[12] Taylor, B. (1715). Methodus Incrementorum Directa et Inversa.

Approaches for Resolving the Dilemma between Model Structure Refinement and Parameter Calibration in Agent-Based Simulations

Manuel Fehler , Franziska Klügl and Frank Puppe
Institute for Artificial Intelligence and Applied Computer Science
University of Würzburg, Germany
{fehler, kluegl, puppe}@informatik.uni-wuerzburg.de

ABSTRACT

Agent-based simulations form a valuable tool for learning about real world societies and global behaviors of systems emerging from microscopic relationships. Calibration of model parameters for detailed agent-based models is a big problem for standard calibration techniques, due to the large parameter search spaces, long simulation run times, uncertainties in the structural model design and different observation levels upon which the model needs to be calibrated. In this paper we propose several methods to improve the calibration process of agent-based simulations.

Categories and Subject Descriptors

I.6 [**Simulation and Modelling**]: Model Development, Model Validation and Analysis

General Terms

Experimentation, Algorithms, Measurement

Keywords

Agent-based Simulation, Simulation Calibration

1. INTRODUCTION

Agent-based simulations (ABS) form a valuable tool for studying real world societies and global behaviors of multi-agent systems (MAS) emerging from microscopic relationships. In ABS agents and societies of a real world system are represented explicitly in a simulation model. Autonomous actors are reproduced by agents with their autonomy, individual goals etc. This allows a close mapping between real and modeled system and leads to advantages in relation to macroscopic simulation techniques and also to other micro modeling paradigms like object-oriented simulation, cellular automatons or petri networks.
To derive valid results and knowledge from an ABS great care needs to be put in the model development process. The main problem does not relate to particular specification and implementation languages, but more generally to the (missing) link between macroscopic aggregate behavior and microscopic agent-level model. One major advantage of ABS is that the observation element "individual" equals the modeling element "agent". However it is not clear what properties and behavior, i.e. modeled structure of the real-world agent, actually lead to the measurable aggregate values. This leads to a rather high uncertainty about a valid model structure and consequently about a valid parameter setting. However, parameter calibration can only be done with an appropriate model structure as starting point. On the other hand, whether a model structure is valid can only be assessed with parameter settings. This dilemma is particular to ABS and cannot be addressed by parameter calibration or optimization that take the model and its structure as a black box. Therefore, new methods and techniques have to be developed that combine parameter calibration with model structure refinement for ABS.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. RELATED WORK

In 1 we described the need for treating parameter calibration and structural refinement of ABS in a combined fashion. As a result the related work is described in three parts:
Black box calibration techniques: In general parameter calibration of simulation models is treated as an optimization problem [4]. The simulation models are adopted as black boxes computing a predefined objective function which measures the overall simulation validity. The problem for direct application of these techniques in ABS is the potentially large level of detail in such simulation models resulting in huge parameter search spaces.
Structural model refinement of ABS: ABM can be designed in a goal-oriented top-down fashion, i.e. starting with an abstract model with the desired macro behavior and adding model detail as required to answer the simulation question, or bottom-up based on the design of local agent relationships. In both cases the step of tuning and refining model structures in case of invalid simulation results is generally handled rather arbitrarily [2].
Systematic Calibration: The calibration issue is not tackled in most agent-oriented engineering methods. Two examples from ABM engineering dealing with calibration are the SADDE methodology [7] which treats the simulation model as a black box employing genetic algorithms (GA) for parameter estimation and the approach described by Calvez

et al. [1] which also uses black box GA.

3. EXAMPLE MODEL: SHOPSIM

Our application example is an ABS of grocery shopping-behavior in Northern Sweden. In the model stores are defined by their position and seven characterizing attributes like price level, assortment size, quality etc. The shopping agents perceive the store attributes and their individual distance to the stores using some perception functions. They combine the different perceptions into a personal store ranking. Based on this ranking the agents spend their yearly grocery shopping budget. The global objective function of the model measures how precisely the empirical turnovers of the stores are reproduced by the model.

The empirical data for the model validation consists of original statistics of store turnovers and estimations of shopping budgets for the shopping agents. The data used to design and calibrate the target model is fuzzy or simply missing. Many stores attributes are based on subjective expert estimations. Further it is unknown which store attributes are actually important for shopping decisions, i.e. whether all or only a subset of the store attributes need to be perceived by the agents. Finally, the potentially over 100,000 agents, evaluating 132 stores each, greatly limits the number of possible parameter configuration evaluations during calibration.

4. MODEL DESIGN AND CALIBRATION

We call our approach a white box calibration approach as opposed to standard black box search (BBS) methods. Our goal is to enhance the efficiency of applied BBS methods by exploiting available expert knowledge and by using calibration methods that explicitly deal with ABM structure.

We suggest a top-down goal oriented analysis of the model and calibration problem. Starting with the conceptual model design, the simulation question(s) to be answered and the necessary level of detail are defined. These form the first two observation levels (macro and micro) upon which we need to ensure model validity. From the simulation question we specify the global objective function for calibration. Based on the analysis of the calibration problem we apply decomposition and abstraction techniques, analyze simplified models and successively refine the model until the desired level of detail has been reached. We do this to overcome arbitrary model changes on local levels.

4.1 Model Abstraction and Decomposition

Based on an analysis of the complexity of the calibration problem one can identify possibilities for increasing the efficiency of applying BBS techniques. The goal is to reduce computational model complexity, to be able to apply individual objective functions for decomposed submodels and to study model relationships on different abstraction levels.

Proposed Proceeding We employ a set of abstraction and decomposition techniques to explore simplified models, create computationally less complex models and analyze local relationships. Abstractions can be aggregations of model units, metamodeling, reductions of heterogeneity or analysis of local interactions. Decompositions can be goal-oriented and on functional, behavioral or temporal levels. A detailed description can be found in Fehler et al.[3].

4.2 Structure and parameter refinement

Several structural model aspects and combinations of these can be responsible for a certain system behavior. We require a technique to efficiently search through possible structural configurations with combined parameter estimation. An example is the calibration of agent perception functions. Agents can potentially perceive several aspects of their environment and act according to reasoning based on these perceptions. However, it may not be known which environmental properties need to be perceived by the agents and if some perception combinations have increased influence on agent behavior. In SHOPSIM we have to identify relevant store attributes and calibrate the associated perception functions.

Proposed Proceeding Our goal is to explore as few perception function subsets (PFS) as possible while identifying the most valid perception model. To search the space of possible PFS we propose to use a combination of a beam search strategy in PFS space, a simulated annealing (SA) search in perception function parameter space and effect analysis using Design of Experiments (DoE) techniques [5]. We start with a subset containing all possible perception functions and optimize its parameters using SA. The next step is to identify the perception functions that are actually responsible for agent behavior. We do this using factorial experiments to determine major and combinational effects up to a preestimated degree using a minimal number of required simulations. Once promising PFS have been determined the experimenter may choose a set of PFS for further investigation following the beam search strategy. The search in structure space may be cut off at any time, if effect analysis suggests no further improvements for certain PFS. Following this procedure in SHOPSIM we explored only 12 instead of potential 127 perception models in order to identify the most valid candidates.

4.3 Validity on multiple observation levels

We want to enforce model validity not only on a global but on several observation levels. Parameter calibration on a single observation level is generally done using objective functions describing degrees of validity on this level.

Proposed Proceeding To ensure model validity on several observation levels we employ behavioral constraints on simulation behaviors during model calibration. Standard constrained parameter optimization uses constraints on the model input parameters, which are tested before and after each parameter configuration evaluation, to guide the search process [6]. We employ behavioral constraints during individual simulation runs. This way we can simply cancel invalid simulation runs, which decreases total calibration time.

4.4 Reasonable introduction of heterogeneity

To keep the parameter search space as small as possible we want to determine small sets of model properties that can be used to describe relevant heterogenous model aspects, like groups of agents or resources. Additionally we want to determine the error inherent in the available fuzzy empirical data. This is important as we derive information of how good we can actually get during parameter calibration in respect to an objective function comparing simulation output to fuzzy empirical data values.

Proposed Proceeding We explore equivalence classes in model properties to determine relevant heterogeneous model aspects which require additional detail, i.e. parameters. In

SHOPSIM this means that all stores with the same attribute values for a given subset of all possible attributes form a single equivalence class. We seek classes which correctly group stores with similar turnovers. If we measure the deviation of individual store turnovers within valid equivalence classes from the mean class turnover, we can derive information of the calibration error inherent in the fuzzy empirical data. In SHOPSIM we see that for the most appropriate equivalence classes the error inherent in the data leads to a quality of only 60% in global turnover reproduction. This helps us in calibrating the actual model parameters as we now know that we will not be able to reduce the error in turnover reproduction to less than 40% during parameter optimization.

4.5 Reverse engineering approach

Suppose we have a bottom-up designed ABM including expert specified structural aspects and parameters. The simulation results are not satisfying. We need to determine required changes in the simulation model without arbitrary changes in the ABM structure.

Proposed Proceeding We propose to invert the calibration process. Instead of fitting agent model parameters to a given environmental model we reverse engineer an environmental model with properties that fit the predefined agent model. On the one hand the resulting model can be analyzed by the domain expert to identify anomalies and thus required model changes. On the other the reverse engineering approach can be used to abstract certain heterogeneous model relationships allowing faster model calibration.

In SHOPSIM we may have expert designed distance perception functions. The parameters of the remaining perception functions need to be calibrated to complement the heterogenous distance perception. Here, we abstract the different possible store attributes into a general store "attractiveness" attribute. The idea is to reverse engineer an attractiveness for each store that fits the agents distance perception. Section 4.6 describes an efficient way to do this.

The first gain is that the resulting store attractivenesses can by analyzed for correlation with original store properties. This allows a more directed adaption of model structures and identification of bad empirical data about store attributes. Suppose we want to analyze the micro validity of agent shopping trips based on different bottom up designed distance perception functions. We do this by analyzing them in environmental models theoretically fitted to the agent perception model using reverse engineering. This means we compute what the attractiveness of each store would have to be in order to correctly reproduce store turnovers, if we assume that the local agent model is correct. This allows to focus local behavior validity analysis on the designed distance perception function without having to deal with possible distortions of agent behavior resulting from behavior based on fuzzy data on the non-distance store attributes.

The second gain is that, by reverse engineering, we abstracted the heterogenous distance attribute of the stores into the store attractiveness attribute which is then perceived homogenously by the agents. To calibrate the original model, the original non-distance store attributes need to be perceived in such a way that the engineered attractiveness is produced. However, we can now use a simple linear model instead of the computationally expensive ABS to calibrate the parameters of the non-distance perception functions. This allows a much faster evaluation of possible parameter configurations and thus the application of more exact estimation techniques like gradient based search.

4.6 Calibration of many local parameters

Suppose we want to calibrate numerous local (agent) parameters as in 4.5. The number of parameters exceeds the size that can be sensibly treated using global BBS. We have an estimation of domain expert knowledge how the local environment of an agent influences its parameter values.

Proposed Proceeding We design local update functions which change the agents parameters based on deviation from desired local properties or behaviors. The functions are based on knowledge about local relationships from the domain expert. We update the parameters in an iterative fashion until the values converge. In the SHOPSIM reverse engineering example from the last section we have to calibrate local attractiveness for 132 stores, i.e. 132 parameters, based on a heterogenous distance agent perception. We defined update functions for each store attractiveness. Depending on the deviation of simulated from the target turnovers the store attractiveness values were adapted. Depending on the chosen perception model 2-5 update iterations were sufficient to calibrate the 132 attractiveness parameters so that the original store turnovers could be perfectly reproduced for given heterogeneous agent distance perception.

5. SUMMARY AND OUTLOOK

In this paper we proposed a framework and associated methods for calibrating ABS designed for prediction and exploration of real world MAS. A main focus was put on the sensible introduction of heterogeneity into the model and the analysis of bottom-up model designs, to overcome the often infeasible trial and error step of parameter and micro level structure refinement. In the future we will further focus on the problem of the introduction of heterogeneity into ABM as this poses a very hard problem to parameter calibration due to the significantly increasing search space size.

6. REFERENCES

[1] Benoit Calvez and Guillaume Hutzler. Automatic tuning of agent-based models using genetic algorithms. In Luis Antunes and Jaime Semao Sichman, editors, *Proceedings of the 6th International Workshop on Multi-Agent-Based Simulation 2005*, 2005.

[2] Jim Doran. *Agent-Based Modelling of Ecosystems for Sustainable Resource Management*, pages 383–403. Springer-Verlag Berlin Heidelberg, 2001.

[3] Manuel Fehler, Franziska Klügl, and Frank Puppe. Techniques for Analysis and Calibration of Multi-agent Simulations. In M.-P. Gleizes, A. Omicini, and Franco Zambonelli, editors, *ESAW 04, LNAI 3451*, pages 305–321. Springer-Verlag Berlin Heidelberg 2005, 2005.

[4] Michael C. Fu. Optimization for Simulation: Theory vs. Practice (Feature Article). *INFORMS Journal on Computing, Vol.14, No.3*, pages 192–215, 2002.

[5] Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, Inc., 1997.

[6] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Verlag New York, Inc., 2000.

[7] Carles Sierra, Jordi Sabater, Jaume Agust-Cullell, and Pere Garcia. Evolutionary programming in SADDE. In *AAMAS 2002*, pages 1270–1271. 2003.

Multiagent Simulation Meets the Real World

Toru Ishida
Department of Social Informatics, Kyoto University
Kyoto 606-8501, Japan
ishida@i.kyoto-u.ac.jp

ABSTRACT

To realize large scale socially embedded systems, this paper proposes a multiagent-based participatory design that consists of steps called 1) *participatory simulation*, where scenario-guided agents and human-controlled avatars coexist in virtual space and jointly perform simulations, and 2) *augmented experiment*, where an experiment is performed in real space by human subjects and scenario-guided extras. In this methodology, we use production rules to describe *agent models* for approximating users, and multiagent scenarios to describe *interaction models* among services and their users. To learn agent and interaction models incrementally from simulations and experiments, we establish the *participatory design loop* with deductive machine learning technologies.

1. INTRODUCTION

Ubiquitous computing systems in the public space often interact with anonymous users and form socially embedded systems. We propose to apply multiagent technologies to the participatory design of ubiquitous computing systems that are to be embedded in society. To this end, we separate agent models from interaction models: the former covers beliefs, desires, intentions and emotions of human users of ubiquitous computing systems, and the latter covers protocol, method, rule or law that guides users when interacting with the ubiquitous computing systems. We view interaction models as behavioral guidelines of human users playing with socially embedded systems; users keep autonomy within the given guidelines. The question is whether or not users will follow the guidelines in an actual ubiquitous environment. The effectiveness of guidelines depends on agent models, which include the personalities of the users (deliberative, reactive, etc.).

We call the descriptions of interaction models *scenarios*. We have developed a scenario description language *Q* and its processor, which can simultaneously interpret scenarios for a large number of agents [4]. *Q* has been already connected to Microsoft agents for Web navigation, FreeWalk [8] for digital city experience [3], Cormas for natural resource simulation [1], and Caribbean for a massively multiagent simulation [11]. We have used *Q* and FreeWalk to conduct crisis management field studies [5,6,7].

In this paper, we pursue *multiagent-based participatory design methodologies* (*participatory design* hereafter) to confirm the adequateness of socially embedded systems; the following steps outline our approach.

1. Describe *interaction models* among ubiquitous computing systems and their users so as to define the expected behaviors of socially embedded systems. We use multiagent scenarios for this purpose, while using production rules to describe *agent models* for approximating users.
2. Perform a *multiagent-based simulation* by modeling users under the given interaction protocols. The simulation takes place in virtual space and involves simulated users, i.e. scenario-guided agents, whose internal models are described in production rules. Results of the simulation can predict how the entire system would work in society.
3. Replace some of the simulated users by human-controlled avatars to perform a *multiagent-based participatory simulation* (*participatory simulation* hereafter). The simulation is performed in virtual space, and the avatars are controlled by the humans sitting in front of their desktop computers. From the results of the participatory simulation, we can improve the models of users, as well as the interaction protocols: i.e. both agent and interaction models.
4. Perform experiments in real space to try out the entire system with human subjects. Since the number of human subjects is often limited, the experiment should be augmented by scenario-guided agents in virtual space and scenario-guided human extras in real space. We called this the *multiagent-based augmented experiment* (*augmented experiment* hereafter); note that most participants will be virtual constructs.

2. PARTICIPATORY SIMULATION

In our participatory design methodology, we first conduct multiagent-based simulations. Figure 1(a) illustrates how a multiagent-based simulation is realized. The scenario processor interprets interaction protocols and requests agents in virtual space to perform sensing and acting functions. Note that, since agents are inherently autonomous, though agents are guided by scenarios, there is no guarantee that they will behave as requested.

We can easily extend multiagent-based simulations to yield *participatory simulations* by replacing some of scenario-guided agents with human-controlled avatars. Figure 1(b) illustrates how humans and agents can cooperatively perform a simulation. Just as video games, human subjects can join simulation by controlling avatars by joy sticks, mice or other input devices. To analyze simulation results, we monitor the entire process of the simulation by visualizing the virtual space. In addition of video taping the virtual space, we record how the human subjects control their avatars. Recording human behavior is often useful for analyzing simulation results.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

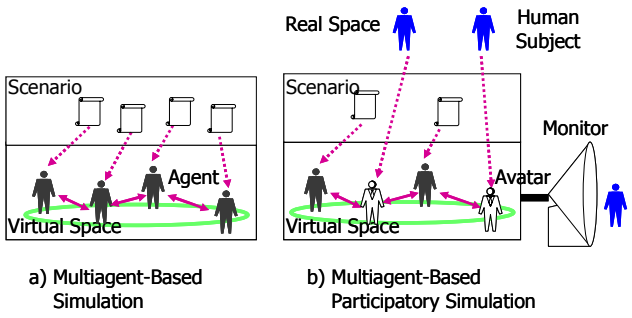


Fig. 1. Participatory Simulation

We conducted a multiagent-based simulation in the evacuation domain. We simulated the controlled experiments conducted by Sugiman [9]. He established a simple environment with human subjects to determine the effectiveness of two evacuation methods: the *Follow-direction method* and the *Follow-me method*. In the former, the leader shouts out evacuation instructions and eventually moves toward the exit. In the latter, the leader tells a few of the nearest evacuees to follow him and actually proceeds to the exit without verbalizing the direction of the exit. Sugiman used university students as evacuees and monitored the progress of the evacuations with different number of leaders. The experiment was held in a basement that was roughly ten meters wide and nine meters long; there were two exits, one of which was not obvious to the evacuees.

Our first evacuation simulation used scenarios with simulated leaders and evacuees in a two-dimensional virtual space. Since the simulation results closely parallel those recorded in Sugiman's controlled experiment, we could conclude that the agents were reasonably well modeled [5]. We then examined the same scenario in the three-dimensional virtual space FreeWalk. One difference between the two-dimensional and three-dimensional simulations is that the latter allowed humans to join the simulation: human subjects can sense their environment and control their avatars in the virtual space.

A participatory simulation was then performed with twelve human subjects: we replaced twelve out of twenty scenario-guided agents by human-controlled avatars. The virtual space is designed so that the human subjects could not distinguish agents from avatars. During the participatory simulation, we found humans behaved differently from their model used in the multiagent-based simulation. We thus adopted machine learning technology to develop agent models from the logs of the participatory simulation. By analyzing the logs and the videos of the simulation, we leaned that a) evacuees use their eyes, not their ears, to identify the leaders, b) upon finding a leader, evacuees tend to approach the leader, even if the Follow-direction method is performed, and c) if contact with the leader is lost, they follow other evacuees that are moving in a group.

Once an accurate model is acquired from participatory simulations, it becomes possible to simulate experiments that cannot be conducted in real space. For example, Sugiman's experiment was performed using two or four leaders, but we can vary this number to clarify the relation between the number of leaders and the time required for evacuation. Moreover, we can explore an effective combination of the Follow-me and Follow-direction methods. Given accurate agent models, participatory simulations are also useful in educating or training people. People can be provided with an experiential learning environment wherein evacuations or other drills can be experienced.

3. AUGMENTED EXPERIMENT

To understand how people behave with socially embedded systems, real-world experiments are often conducted. In the case of video phones, for example, since user behavior in everyday life is not easy to simulate, it is essential to observe how users utilize video phones in real space. In ubiquitous computing, however, because a large number of electronic devices are embedded in public space like railway stations and are continuously in service, it is often impossible to conduct experiments with large number of human subjects. *Augmented experiments* offer ability to perform experiments with a small number of human subjects through augmentation of multiagent-based simulations.

Figure 2 illustrates how augmented experiments are realized. Figure 2(a) represents a real world experiment, where human subjects communicate and perform an experiment in real space. Figure 2(b) shows how we introduce virtual space into a real world experiment. Using sensors in real space, behavior of human subjects are reproduced in virtual space. The sensors can be cameras, RFIDs or GPS depending on the environment. By using virtual space, we can monitor the entire experiment from various viewpoints. Furthermore, we can communicate with human subjects in real space by selecting their avatars in virtual space. *Transcendent communication* is a new monitoring interface, where a visually simulated public space provides a more flexible and consistent view than regular surveillance systems [7].

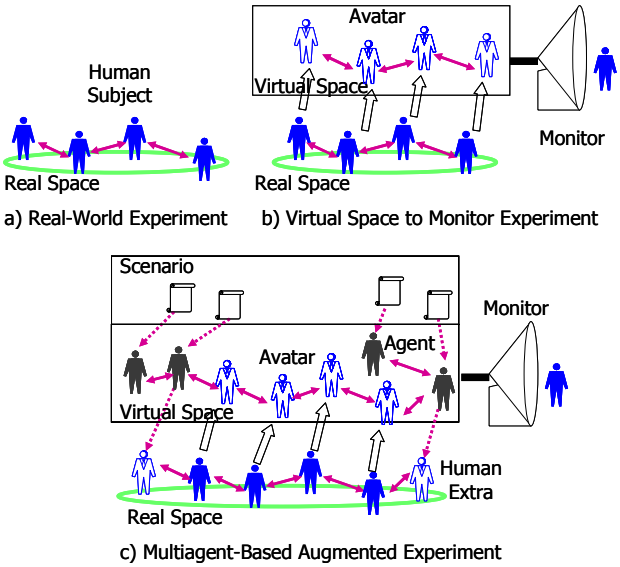


Fig. 2. Augmented Experiment

Figure 2(c) illustrates an augmented experiment. In parallel with a real world experiment, a large scale multiagent-based simulation is conducted in virtual space: the experiment is augmented by the simulation. To combine real and virtual spaces, sensors are used to project human behavior into the virtual space as shown in Figure 2(b). Furthermore, to provide social reality to the human

subjects, scenario-guided extras are placed around the subjects. In contrast to participatory simulations, the human extras do not control avatars: the human extras in the real space are controlled by the scenario-guided agents. For example, we may use human extras acting bus drivers to conduct a demand bus experiment. Conducting an augmented experiment is possible if the real space is equipped with enough sensors. Though we can monitor the experiment via virtual space, video capture of the human subjects in real space is also indispensable for analyzing the entire experiment.

Augmented experiments are needed for testing evacuation methods in railway stations, because there is no other way to conduct experiments with enough reality. To conduct an augmented experiment, we placed twenty eight cameras in Kyoto station, and captured the movements of passengers in real time. Those cameras are specially designed to detect passengers' behavior but not personal features.

We then reproduced the passengers' behavior in virtual space. The evacuees on the station platform are projected into virtual space. The bird's-eye view of real space is reproduced on the screen of the control center so that a leader in the center can easily monitor the experiment [7]. Furthermore the leader can point at particular passengers on the screen, and talk to them through their mobile phones. When the monitor detects pointing operations, a wireless connection is immediately activated between the control center and the indicated passengers. Dragging operations indicating a rectangular area would enable the leader to broadcast an announcement to a group of passengers.

The difficulty with augmented experiments is to provide the human subjects with a sufficient level of reality. To guide extras by scenarios in a timely fashion, the agents must receive sensing information from the corresponding extras. Learning issues are similar to those in participatory simulations, but the evaluation is more difficult, since the behaviors of human subjects in real space are harder to analyze than those in participatory simulations.

4. Future Work

By analyzing the logs of participatory simulations or augmented experiments, we can refine agent and interaction models. The learning process of multiagent-based participatory design consists of two phases: the *normal design loop* and the *participatory design loop*. At first, scenario writers create agent and interaction models and verify their design by multiagent-based simulations. We call this process the normal design loop and have already proposed its detailed steps [5]. After verifying the agent and interaction models, participatory simulations or augmented experiments are performed. The results of which may be different from those of multiagent-based simulations, because human behaviors are often more variable than those of agents modeled by scenario writers. If it is the case, we first refine the agent models to better express the variation seen in human behavior. We then perform the multiagent-based simulations again with the new agent models. If the new simulations fail to produce the expected effects, we then refine the interaction models: modify protocols, methods, rules, and/or laws. After the improved interaction model is confirmed by multiagent-based simulations, participatory simulation or augmented experiment takes place again. This loop is called the participatory design loop.

At first glance, inductive machine learning technologies can be applied to refine agent models and interaction models. However, it is extremely rare to have a large enough number of human subjects to gather enough data, for inductive learning: deductive methods with domain knowledge are more feasible. To refine agent models, given that the training examples will be limited, explanation-based learning is a reasonable approach. Though variations exist in human behavioral rules such as "escape from crowds" and "follow crowds," observing the behavior of a human-controlled avatar can create a plausible explanation that selects one of the two rules. By combining machine learning technologies with interviews of human subjects, we have successfully refined agent models in several domains [6, 10]. To refine interaction models, if we have enough candidate protocols, we can try them, and select the one that best fits. If we need to invent a new model, however, there is no available technology that can create such an interaction model, while the visualization of simulation processes can support scenario writers in inventing new models.

REFERENCES

[1] F. Bousquet, I. Bakam, H. Proton and C. Le Page. Cormas: Common-Pool Resources and Multiagent Systems. *LNAI* 1416, pp. 826-838, 1998.

[2] A. Drogoul, D. Vanbergue and T. Meurisse. Multiagent-Based Simulation: Where are the Agents? *LNAI* 2581, pp. 1-15, 2002.

[3] T. Ishida. Digital City Kyoto: Social Information Infrastructure for Everyday Life. *Communications of the ACM (CACM)*, Vol. 45, No. 7, pp. 76-81, 2002.

[4] T. Ishida. *Q*: A Scenario Description Language for Interactive Agents. *IEEE Computer*, Vol. 35, No. 11, pp. 54-59, 2002.

[5] Y. Murakami, T. Ishida, T. Kawasoe and R. Hishiyama. Scenario Description for Multi-Agent Simulation. *AAMAS-03,* pp. 369-376, 2003.

[6] Y. Murakami, Y. Sugimoto and T. Ishida. Modeling Human Behavior for Virtual Training Systems. *AAAI-05*, pp. 127-132, 2005.

[7] H. Nakanishi, S. Koizumi, T. Ishida and H. Ito. Transcendent Communication: Location-Based Guidance for Large-Scale Public Spaces. *CHI-04*, pp. 655-662, 2004.

[8] H. Nakanishi and T. Ishida. FreeWalk/Q: Social Interaction Platform in Virtual Space. *VRST-04*, pp. 97-104, 2004.

[9] T. Sugiman and J. Misumi. Development of a New Evacuation Method for Emergencies. *Journal of Applied Psychology*, Vol. 73, No. 1, pp. 3-10, 1988.

[10] D. Torii, T. Ishida and F. Bousquet. Modeling Agents and Interactions in Agricultural Economics. *AAMAS-06,* 2006.

[11] G. Yamamoto and Y. Nakamura. Architecture and Performance Evaluation of a Massive Multi-Agent System. *Agents-99*, pp. 319-325, 1999.

Congestion Tolls as Utility Alignment Between Agent and System Optimum

Ana L. C. Bazzan* and Robert Junges
Instituto de Informática, UFRGS
Caixa Postal 15064
91.501-970 Porto Alegre, RS, Brazil
{bazzan,junges}@inf.ufrgs.br

ABSTRACT

In multi-agent systems, greedly agents can harm the performance of the overall system. This is the case of traffic commuting scenarios: drivers repete their actions trying to adapt to daily changes. In this domain, there are several proposals to achieve the traffic network equilibrium. Recently, the focus has shifted to information provision in several forms as a way to balance the load. Most of these works make strong assumptions such as the traffic authority and/or drivers having perfect information. In reality, the information the central control provides to drivers contains estimation errors. The goal of this paper is to propose a socially efficient load balance by internalizing social costs caused by agents' actions. Two issues are addressed: the model of information provision accounts for information imperfectness, and the equilibrium which emerges out of drivers route choices is close to the system optimum due to mechanisms of road pricing. The model can then be used for traffic authorities to simulate the effects of information provision and toll charging.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence
Multiagent systems, Coherence and coordination

General Terms

Economics

1. INTRODUCTION

The increasing demand for mobility in our society causes frequent jams in highways. Control measures have been proposed to better control the utilization of the available infra-structure; among these we find *driver information and guidance systems* (DIGS). Normally, information provision and route guidance strategies aim at achieving the *system* optimum (minimization or maximization of some *global* objective criterion). Note that this does not imply the *user* optimum and vice-versa. User optimum means equal costs for all alternative routes connecting two points in the network, for each road user. This may lead to system' suboptimality. If the focus is on global optimum, then the route guidance system may eventually recommend a route which is more costly (for a single user) than it would be the case if the user optimum were to be recommended. In general, traffic control authorities are interested in the system optimum, while the user seeks its own optimum.

In a more general way, this question relates to emergence of coordination in multiagent systems, in cases where the agents do not communicate and thus have to achieve coordination in a somehow greedy way. This normally causes the system to perform suboptimally. Therefore, it is important to align the user and the system optimum. Previous works have been proposed but mostly they assume some degree of cooperation and communication among the agents. Since drivers are self-interested, this alignment cannot be imposed by the system as it has been proposed in other works.

*Author supported by CNPq

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. RELATED WORK: TRAFFIC, INFORMATION, AND COLLECTIVES

Modeling traffic scenarios with multi-agent systems techniques is not new. However, the focus has been mainly on coarse-grained level regarding traffic problems as traffic agents monitoring problem areas. On the other hand, our long term work focuses on a fine-grained level or rather on traffic flow control.

Our previous work with a scenario of iterated route choice provides a good opportunity to test learning and emergence of coordination. The main problem is to achieve system optimum or at least acceptable patterns of traffic distribution out of user's own performance strategies, i.e. users trying to adapt to the traffic patterns they experiment every day. There are different means to achieve a certain alignment of system and user optimum without relegating important issues such as lack of knowledge of non-commuter users, traffic information and forecast, etc.

The case of simple user adaptation in a binary route choice scenario, with and without traffic forecast was tackled in [3]. A scenario was simulated where N drivers had to decide for one out of the available routes, in every round. At the end of the round, every driver gets a reward that is computed based on the number of drivers who selected the same route,

in a kind of coordination game. In [4] different forms of traffic information – with different associated costs – were analyzed which may be used to assess the future traffic situation for the selection of one route. Finally, information recommendation and manipulation was tested in [2] with a traffic control center giving manipulated information for drivers in the scenario of the Braess Paradox, in a tentative of diverting drivers to less congested routes.

In all these works, different means of utility alignment were tried. Some were more successful than others regarding the performance of the global metrics. However, in the present paper we want to drop the "perfect information" assumption (both the traffic control center and all individuals having knowledge of all alternatives).

From the perspective of the economics of traffic, road pricing has been proposed as a way to realize efficient road utilization i.e. to achieve a distribution of traffic volume as close as possible to the system optimum. Congestion toll is one of the road pricing methods: considering the system optimum, a toll is computed which is the difference between the marginal social cost and the marginal private cost. Notice that this difference can be negative meaning that drivers actually get a reimbursement.

Regarding the future traffic situation as well as the *effects* of the toll system (and other measures to control the traffic), most of the work published has assumed that the control center has perfect information, meaning that it will know precisely the states of near future traffic conditions. This is a hard assumption given that acurate weather and accident forecasts are not possible. A state-dependent toll pricing system is discussed in [5]. Additionally, it is investigated which are the impacts of two alternatives toll schemas: toll is charged before or after drivers select a route. This distinction is important because congestion tolls, when announced before the drivers make their decisions, carry some meaningful information.

Still related to the quest of user vs. system optimum, Tumer and Wolpert [6] have investigated the use of a multiagent system to control routing of packages in a computer network, as well as the sensitivity of the network to the Braess paradox using the Collective Intelligence (COIN) formalism. In COIN, the world utility, $G(\zeta)$, is an arbitrary function of the state of all agents across all times. The utility for an agent is given by the difference between the total cost accrued by all agents in the network and the cost accrued by agents when all agents sharing the same destination are "erased". The problem with the use of this approach in a road network is that we cannot expect the agents (drivers) to have any idea about the global goal or cost. This is possible in the computer network scenario because the agents are router nodes which have knowledge about the throughput (packages to be routed to different destinations).

3. MODEL AND RESULTS

We have developed a simple model for the adaptive route choice. Since an agent has only qualitative information about the routes, and none about the other agents, each agent needs an expectation of the costs s/he will have if s/he selects a certain route. This can be understood as the probability according to which a driver selects one route. For instance, if it is 1 for route r then the driver always takes route r. With a certain periodicity, driver d updates this heuristic according to the rewards s/he has obtained on the routes s/he has taken so far. The update of the heuristic is done according to the following formula: $heuristics(d, r_i) = \frac{\sum_t utility_{r_i}(d,t)}{\sum_i \sum_t utility_{r_i}(d,t)}$.

The variable $utility_{r_i}(d,t)$ is the reward agent d has accumulated on route r_i up to time t. Therefore an important factor is how often and in which intervals the heuristic is updated. This is especially relevant because the reward depends on the other agents. Using this simple model, we have performed experiments varying the frequency of heuristic adaptation. To prevent that all agents update their heuristics during the same round, each agent adapts with a given probability.

In order to model the imperfectness of information, we follow [1] who represent the traffic conditions as L discrete states. We also consider K information types. For each combination of state and information type, there is a cost function which also depend on the traffic volume. Thus, the dynamics on the routes are simulated according to these abstract cost functions which nonetheless can reproduce the macroscopic behavior *of the system* given that it includes a stochastic component regarding the traffic states.

The cost functions we use were adapted from [5], from which we also use much of the nomenclature, cost functions, and some scenarios, althought these were slightly modified to include the drivers's adaptation to route choice. We also assume R alternative routes between two points in the network. Both have a traffic capacity of M vehicles. Traffic can be in one of L states (e.g. if $L = 2$ we can have congested/non-congested states only). In the model, each state occurs with probability q^l. The control center predicts traffic based on imperfect information and provides information to the drivers which is also imperfect. There can be K types of information. At a given time, one information k is given with probability p^k. There can be any correspondence between K and L but mostly it is assumed that $L \geq K$, $K \geq 2$, and $L \geq 2$.

The probability of a state l to occur after information k is given by $\pi^{k,l}$. If $\pi^{k,l} = 1/L$, the information conveys basically no meaning i.e. it is tantamount to no information. When $L = K$ (i.e. for each k there is an exclusive l) we can have perfect information provided one of the π's is one and the others are zero. For example, if $K = L = 2$, when $\pi^{k,l_1} = 1$ and $\pi^{k,l_2} = 0$ ($l_1 \neq l_2$), this is a situation of *perfect information provision* because it is known for sure which state is expected to occur after the information is provided.

3.1 Parameters and Settings

In the examples discussed here, $R = K = L = 2$, and the travel cost functions used are linear of type $c_i^l = \zeta_i^l + v_i^l \times x_i^k$, where x is the number of drivers. In particular we use the following functions [5]:

$c_1^1 = 1.0 + 0.6 \times 0.001 \times x_1^k \quad c_1^2 = 1.5 + 1.6 \times 0.001 \times x_2^k$
$c_1^2 = 1.0 + 0.4 \times 0.001 \times x_1^k \quad c_2^2 = 1.5 + 0.1 \times 0.001 \times x_1^k$

The meaning of these functions is that the marginal costs (v_i^l) of each route r_i differ. Therefore we expect the number of drivers to be smaller in route 2 when $l = 1$. In [5] the system equilibrium (without any toll etc.) is computed for $M = 3000$ drivers: 2391.5 (route 1) and 608.5 (route 2) for $k = 1$; 1927.3 (route 1) and 1072.7 (route 2) for $k = 2$.

We show here that this equilibrium is never achieved when drivers use greedy strategies such as selection of route based

on average of past travel time. However, when the selection is based on an utility function which includes the toll paid or the amount reimbursed, then traffic volumes are close to the system equilibrium.

3.2 Scenarios

In **scenario I** we are interested in reaching the equilibrium by only allowing users to apply their probabilities to select route r_i, given the information provision k. In our case this is done via adaptation of these probabilities (denotated as $\rho^d_{r,k}$) given the past utilities which are function of the costs. The basic mechanism for each driver d is given by: $\rho^d_{r,k} = \frac{\overline{U^d_{r,k}}}{\sum_{i=1}^{R} \overline{U^d_{i,k}}}, 1 \leq i \leq R$, where $\overline{U^d_{r,k}}$ is the average of past utility for selecting route r_i when the information provided was k.

In **scenario II** a toll is computed by the traffic control center and paid by the driver. This computation depends on the information k so that drivers have to make a relation to their previous choice for the last situation where k was provided. The toll value for each driver on route r_i is calculated as $\tau^k_{r_i} = \frac{x^k_r - x^{k^*}_r}{x^k_r}$ where: $x^{k^*}_{r_i}$ is the number of drivers *in the equilibrium situation* for route r_i given information k, and $x^k_{r_i}$ is the expected number of drivers, estimated by the last time k was provided. Notice that this value can be positive (driver pays) or negative (driver gets a reimbursement).

The reasoning of the driver d is as follows. If information k is provided and the last time this has happened I was reimbursed because I have chosen route r_i, then I better select r_i again with probability $\rho^d_{r_i,k} = 1 - rate_{cur}$. However, if information k is provided and the last time this has happened I had to pay, then I select r_i with probability $\rho^d_{r_i,k} = \tau^d_{r_i,k}$. Here $rate_{cur}$ is the rate of curiosity, i.e. a probability of d experimenting another route (other than r_i) even though r_i was good the last time; and $\tau^d_{r_i,k}$ is the toll paid by driver d when selecting route r_i under information k.

3.3 Results

We have used $M = 3000$ agents in order to compare the results to those in [5]. If we let the drivers select routes only according to the utility they perceive (**scenario I**), then an user stable state is reached, but it is far from the system optimum. Number of drivers in both route (graphic not shown here) are not as mentioned before (Section 3.1). For $k = 2$ the $M = 3000$ drivers select each route with equal probability, so that nearly 1500 end up in each route, at any given time. This happens because the utility of drivers is nearly the same when we substitute $x^k_i = 1500$ in the cost functions. For $k = 1$, the expected system equilibrium is two-thirds in route 1 and one-third in route 2. However, this equilibrium is nerver reached: users are stuck in a suboptimal stable state. Thus, a mechanism is needed which internalizes the externalities caused by the drivers selections.

Our **scenario II** simulates exactly what happens when drivers update their route choice probabilities according to a utility which is based on the past toll. Figure 1 depicts the number of drivers in each route, for each k, when a toll is charged and the curiosity rate is zero, meaning that drivers act only based on the the reward or punishment represented by the toll in the last time information k was provided. As we see in that graph, the user equilibrium is now much closer to the system equilibrium. For $k = 1$ the distribution of drivers is around 2300 to 700 (routes 1 and 2 respectively), and for $k = 2$ this distribution is around 2000 to 1000.

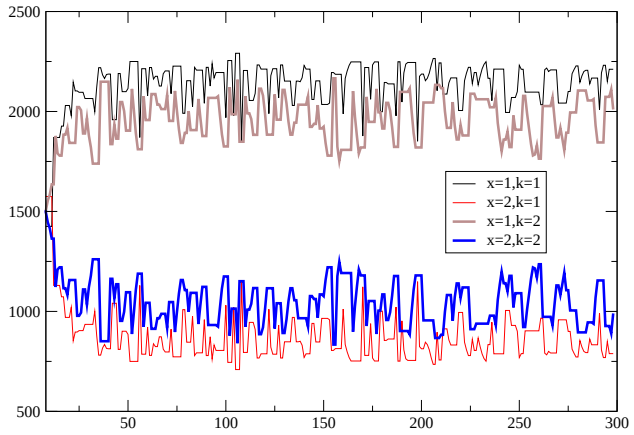


Figure 1: Number of drivers at each route, for each k; with toll and curiosity rate equal 0

Different rates of curiosity of course change these figures. The higher the rate, the more drivers deviate from the system equilibrium.

In summary, this paper shows that congestion tolls are useful to internalize the costs drivers impose to others when they act greedly. Tolls are the equivalent of the utility alignment proposed in the COIN framework, but departing from the assumption that agents are cooperative. For the context of multiagent systems, the present work can contribute to the development of a more general method of mechanism design, specially when agents are self-interested and the system optimum has to be reached without imposing explicit central coordination.

4. REFERENCES

[1] Arnott, R., de Palma, A., and Lindsey, R. Does providing information to drivers reduce traffic congestion? *Transportation Research A 25* (1991), 309–318.

[2] Bazzan, A. L. C., and Klügl, F. Case studies on the Braess paradox: simulating route recommendation and learning in abstract and microscopic models. *Transportation Research C 13*, 4 (August 2005), 299—319.

[3] Klügl, F., and Bazzan, A. L. C. Route decision behaviour in a commuting scenario. *Journal of Artificial Societies and Social Simulation 7*, 1 (2004).

[4] Klügl, F., Bazzan, A. L. C., and Wahle, J. Selection of information types based on personal utility - a testbed for traffic information markets. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)* (Melbourne, Australia, July 2003), ACM Press, pp. 377–384.

[5] Kobayashi, K., and Do, M. The informational impacts of congestion tolls upon route traffic demands. *Transportation Research A 39*, 7–9 (August-November 2005), 651–670.

[6] Tumer, K., and Wolpert, D. Collective intelligence and braess' paradox. In *Proceedings of the AAAI* (2000), AAAI Press, pp. 104–109.

Agent-based modeling of human education data

Yuqing Tang
Department of Computer Science
Graduate Center, City University of New York
365, 5th Avenue
New York, NY 10016, USA
ytang@gc.cuny.edu

Simon Parsons and Elizabeth Sklar
Department of Computer & Information Science
Brooklyn College, City University of New York
2900 Bedford Avenue
Brooklyn, NY 11210, USA
{parsons,sklar}@sci.brooklyn.cuny.edu

ABSTRACT

Agent-based simulation is increasingly used to analyze the performance of complex systems. There are two main ways agent-based models are built — from equation-based models and directly from data. We are building models in both ways, investigating approaches for creating them and for validating them. In this paper we describe results of our work on one specific agent-based model, showing how it can be validated against the equation-based model from which it was derived, and the extent to which it can be used to derive additional results over and above those that the equation-based model can provide.

Categories and Subject Descriptors

I.6.6 [**Simulation and Modeling**]: Model Validation and Analysis

General Terms

Experimentation, Verification

Keywords

Agent-based modeling, simulation, education

1. INTRODUCTION

Agent-based modeling contrasts with traditional approaches to simulation, which typically involve building sets of interrelated differential equations. Such traditional models, commonly called *equation-based models* (EBMs), have been widely applied and generate useful predictions about the behavior of populations. So why use agent-based models? There seem to be four main answers [2]:

- agent-based models are a natural way to describe systems comprised of interacting entities;
- agent-based models are flexible;
- agent-based models capture emergent phenomena; and
- agent-based models provide access to a greater level of useful detail.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profi t or commercial advantage and that copies bear this notice and the full citation on the fi rst page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specifi c permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

In particular, modeling interactions between entities can be much easier in agent-based systems than in EBMs, even when one is comfortable with the concepts of partial differential equations, and can reveal details of the system being modeled that are obscured by EBMs.

In this paper, we describe results of our work on one specific agent-based model, showing how it can be validated against the more traditional model from which it was derived, and highlighting the extent to which it can be used to derive additional results over and above those that the traditional model can provide.

2. A MODEL OF HUMAN CAPITAL

The model that we consider in this paper is drawn from [3], an article that derives a linear model from US census data, and analyzes the aggregate behavior of the model. The model is derived in [3] to identify the effect of the tendency for human societies to stratify by level of education — so-called *human capital*. The model from [3] gives the level of human capital $z_{i,t+1}$ of members of the $t+1$th generation of the ith dynasty as being:

$$z_{i,t+1} = k_{t+1} + \alpha\left(\frac{z_{i,t} + z'_{i,t}}{2}\right) + \beta\left(\frac{\sum_{j=1}^{n} z_{j,t}}{n}\right) + \epsilon_{i,t+1} \quad (1)$$

The first term measures the effect on the level of education ("education" and "human capital" are used more or less interchangeably in this model) of the $t+1$th generation of the education of its parents in the t-th generation. The second term does something similar based upon the neighbors of the parents. The third term is constant across dynasties, but may vary in time to capture exogenous trends in education — for example legislation that requires a certain number of years of additional schooling for given generations. The final term captures a specific "shock" to the human capital in a specific generation of a specific dynasty — for example the early death of a parent, requiring the children to curtail their education.

The notion of "dynasty" and "generation" used here require a little explanation. Each generation of the ith dynasty has two children, one male and one female. Each is assumed to then become the spouse of an opposite sex member of another dynasty, forming a family which in turn produces one male and one female child. One family from a given generation of the ith dynasty remains in the ith dynasty, and one becomes part of another dynasty (the dynasty of the corresponding non-ith partner). Thus there is a constant number of members of each generation, and of each dynasty at each generation.

Based on this model, we have developed an agent-based simulation with a fixed number of agents, n in each generation, with $n/2$ dynasties, and 2 children per family. The basic simulation loop, which executes once for each generation, has three steps:

1. Establish level of z based on:
   (a) Parents
   (b) Neighbors of parents
2. Establish factors that influence z for children
   (a) Spouse
   (b) Neighbors
3. Generate children

Step 1 is fixed by (1), and Step 3 is fixed by the requirement to produce one male and one female child in each generation. Clearly the results are going to depend on the way in which Step 2 is implemented, and our model includes a number of variations described in full in [9].

3. EXPERIMENTS

An agent-based version of the model described in the previous section was implemented in REPAST [5], a Java-based Swarm-like [8] tool developed at the University of Chicago for agent-based modeling in social science applications. We handled the geographic aspects by placing agents on an $n \times n$ grid, where at most one dynasty "lives" in a single grid-square. By varying the size of the grid and number of agents we can create environments of differing population density and have modeled communities of up to 10,000 dynasties.

3.1 Verification

Having constructed an agent-based model of human capital from the equation-based model in [3], we first need to "complete the loop" by performing a statistical analysis of the results from the agent-based model, obtained when using the parameter values assumed in the paper, to show that our agent-based model will achieve the same results as the equation-based model we started with. This verification step is needed in order to justify the further experimental results with the model.

The central result of [3], and the only quantitative result that we can use to check the model against, is the prediction that increasing sorting — which the paper takes to mean increasing the correlation between the human capital values of the parent agents of a generation — will only cause an increase in inequality — which the paper takes to mean that the standard deviation of the human capital distribution grows generation by generation — when the value of α is large. [3] demonstrates this by showing the effect of changing correlation from 0.6 to 0.8 for various values of α. Running experiments on a 50×50 grid — which allows us to deal with a population that is considerably larger than the 1500 individuals analyzed in [3] — we find that our model gives good agreement with the predictions made in [3].

For example, we can plot the effect of parental choice of spouse in terms of the percentage change in inequality (as defined in [3]) against α, the parameter that mediates the effect that parents have on their children's human capital. This gives Figure 1.

3.2 Identifying new features

As we discussed above, one of the advantages that agent-based models have over equation-based models is that one can examine the model in greater detail. Whereas equation-based models can only really be studied in terms of broad statistical features — such as the results from [3] examined above — we can probe agent-based models in considerable detail, discovering what happens to

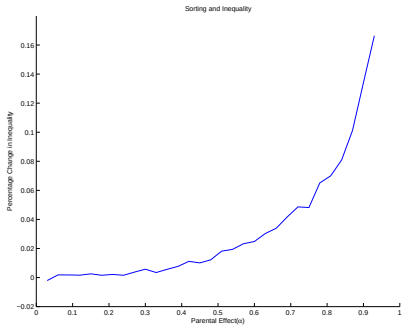


Figure 1: The relationship between the parental effect α and the percentage change in inequality.

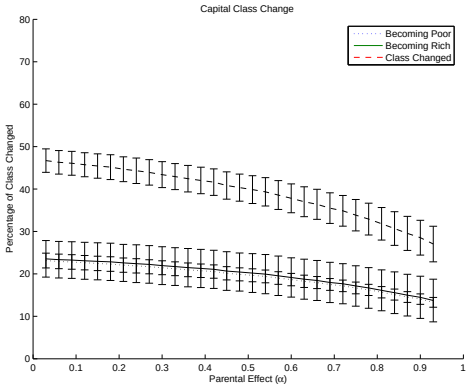


Figure 2: The relationship between the percentage of dynasties that change "class" and α.

individuals as well as to classes of individual. We have carried out such an investigation into the human capital model.

The headline result from [3], replicated by our agent-based model is that *on average* inequality in terms of human capital grows over generations. The widening standard deviation of the human capital distribution suggests that rich dynasties get richer and poor dynasties get poorer. However true this may be at a population level, it is interesting to ask whether it is true for all (or even most) individual dynasties, or whether there is some mobility between dynasties with different levels of human capital.

It turns out that such mobility exists.

We divided our dynasties up into three "classes" — the quotes reminding us that this terminology, while convenient, conflates human capital, basically years of formal schooling, with monetary capital. We call dynasties that fall within one standard deviation above or below the average human capital for the population *middle class*, we call those more than one standard deviation below average *poor*, and those more than one standard deviation above average *rich*. We then examined whether dynasties moved between classes.

The results are given in Figures 2 and 3. These show the way that the number of dynasties that are mobile in this sense changes for two different values of α and β, respectively. When α changes, β is held constant and vice-versa. The graphs plot three values, the total percentage of dynasties that move, the percentage that become richer, and the percentage that become poorer, all data being presented along with standard deviations. The results show that,

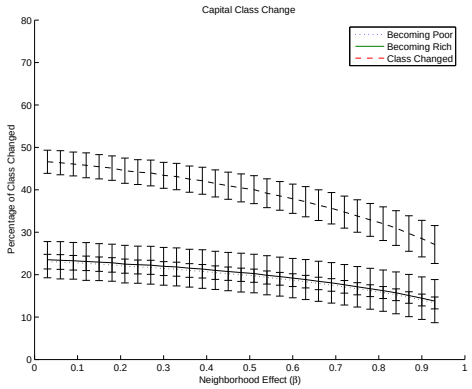


Figure 3: The relationship between the percentage of dynasties that change "class" and β

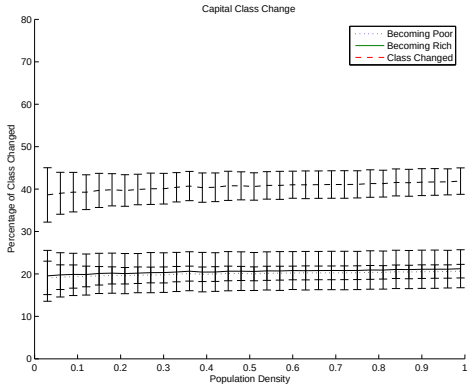


Figure 4: The relationship between the percentage of dynasties that change "class" and population density.

no matter what the value of α and β, there is some mobility (at least 25% of the population, and as much as 45% of the population changes class). Furthermore this change is symmetrical — more or less the same number of dynasties get richer as get poorer.

Note that this effect is separate from the growing inequality — because "middle class" is always defined in terms of the *current* standard deviation, if inequality was the only effect, the percentage of dynasties changing class would be lower than the figure we find. What we see here is the result of mixing, that is, individuals choosing partners or neighbors who are sufficiently far above or below them in human capital terms that their offspring move from one class to another.

We also checked that class mobility was unaffected by other parameters of the model. Since the neighbor effect is based upon a geographic notion of neighborhood, and since neighbors certainly have an effect on class mobility, then one might imagine that changing the density of the population might have some effect on class mobility as well. However, this is not the case. As Figure 4 shows, population density has no systematic effect on class mobility.

For further details of our results, see [9].

4. SUMMARY

This paper set out to show that it was possible to construct an agent-based model from a traditional, equation-based, model, and: (i) verify the agent-based model against the predictions made by the equation-based model; and (ii) use the agent-based model to identify new predictions that cannot be obtained directly from the equation-based model. Both these aims have been achieved.

This work fits into a wider effort that is attempting to model aspects of the education system [7], with the overall aim of being able to identify, and thus establish the impact of, changes in education policy (rather as [1] does for the case of rent control). As described in [7], we have developed a number of models, including a model of interactions in classrooms [6] — which allows us, for example, to model the effects of different pedagogical techniques to overcome absenteeism — and a model of school districts — which, for example, enables us to study the effect of policies like "No child left behind".

Our aim is to tie these models together, and, more ambitiously, to tie them into a comprehensive simulation of the way that education fits into the economy as a whole. One way to think of this is through the model provided in [4]. This describes individuals as making a choice about how much education to receive. This education, combined with their inherent aptitude, equips each agent with a certain level of productivity which, after its education is complete, allows it to pay off its education and purchase units of consumption. Thus education becomes a decision which both enables consumption and constrains it (because education has to be paid for). The kinds of model described in [7] tie in with this approach — the classroom model will affect the way that ability relates to productivity for example — and the human capital model described here provides a means of relating the education decisions made by one generation to those made by the next, giving us a means of projecting the effects of our models over generations.

5. ACKNOWLEDGMENTS

This work was partially supported by funding from NSF #REC-02-19347.

6. REFERENCES

[1] R. N. Bernard. Using adaptive agent-based simulation models to assist planners in policy development: The case of rent control. Working Paper 99-07-052, Sante Fe Institute, 1999.

[2] E. Bonabeau. Agent-based modelling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Science*, 99(3):7280–7287, May 2002.

[3] M. Kremer. How much does sorting increase inequality? *The Quarterly Journal of Economics*, 112(1):115–139, Feb. 1997.

[4] J. Laitner. Earnings within educational groups and overall productivity growth. *The Journal of Political Economy*, 108(4):807–832, August 2000.

[5] `http://repast.sourceforge.net`.

[6] E. Sklar and M. Davies. Multiagent Simulation of Learning Environments. In *Proceedings of the Fourth International Conference on Autonomous Agents and MultiAgent Systems (AAMAS-2005)*, 2005.

[7] E. Sklar, M. Davies, and M. S. T. Co. SimEd: Simulating Education as a MultiAgent System. In *Proceedings of the Third International Conference on Autonomous Agents and MultiAgent Systems (AAMAS-2004)*, pages 998–1005, 2004.

[8] `http://www.swarm.org/`.

[9] Y. Tang, S. Parsons, and E. Sklar. Agent-based modelling of human education data. Technical report, Department of Computer and Information Science, Brooklyn College of the City University of New York, 2005. (the full version of this paper).

Study of Social Consciousness in Stochastic Agent Based Simulations: Application to Supply Chains

Thierry Moyaux
FOR@C & DAMAS
Pav. Pouliot, Dpt. Informatique
Univ. Laval, Quebec, Canada
moyaux@iad.ift.ulaval.ca

Brahim Chaib draa
DAMAS
Pav. Pouliot, Dpt. Informatique
Univ. Laval, Quebec, Canada
chaib@ift.ulaval.ca

Sophie d'Amours
FOR@C & CENTOR
Pav. Pouliot, Dpt. G. Mecanique
Univ. Laval, Quebec, Canada
sophie.damours@gmc.ulaval.ca

ABSTRACT

Empirical game theory allows studying the strategic interactions of agents in simulations. Specifically, traditional game theory describes such interactions by an analytical model, while empirical game theory employs simulations. In this paper, we use empirical game theory to study how the more-or-less selfishness of agents affects their behaviour. To this end, we assume that every agent utility can be split in two parts, a first part representing the direct utility of agents and a second part representing agent social consciousness, i.e., their impact on the rest of the multiagent system. An application to supply chains illustrates this approach. In this application, the collaborative strategy is often used by every company-agent at whatever their same level of social consciousness, which may indicate that every agent is strongly related with one other.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Economics, Experimentation, Performance

Keywords

Supply Chain, Nash equilibrium, Simulation, Collaboration

1. INTRODUCTION

Walsh and his colleagues [5] have introduced the concept of empirical game theory (GT) by using GT to analyze the behaviour of agents in simulations. In other words, they replaced the analytical model representing strategic interactions in GT by simulation outcomes. Another example of the use of this approach is provided by Wellman and his colleagues [6] who used it to design their agent for the Trading Agent Competition/Supply Chain Management 2003. Our previous work [3] is also such an example of approach suitable for the analysis of interactions in agent-based simulations profitting from GT concepts and applied to supply chains (SC). In our application to SC, we found that the highest level of collaboration was often a Nash equilibrium which incurs the minimum SC cost. We now enhance this previous work in two ways in order to study (Enhan. 1) how the obtained results vary when the utility measuring the achievement of the group goal changes with regard to the individual utility of every agent, and (Enhan. 2) how robust the results are with respect to the stochasticity of some parameters. Our approach is presented in Section 2.

Then, we apply our approach to SC again. In this context, Enhan. 1 is performed by measuring how the SC (i.e., group) goal of delivering products to end-customers is achieved, while the individual goal of every company is to minimize its inventory holding cost. Indeed, companies have to make a trade-off between increasing their inventory level to avoid stock-outs and achieve thus the group goal, and decreasing their inventory level to achieve their individual goal. Finally, the stochastic parameter in Enhan. 2 represents end-customers demand of the two markets buying products to the SC. With Enhan. 2, we check if results depend on the considered instance of a single demand distribution. This application to SC management is described in Section 3.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8–12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. APPROACH

Concerning Enhan. 1, every agent $i \in I$ in our previous work [3] had a common fixed way to calculate its utility u^i that was based on its direct/individual utility u^i_{dir} and on its impact on other agents u^i_{soc}, so that $u^i = ((1-\epsilon^i)u^i_{\text{dir}} + \epsilon^i u^i_{\text{soc}})/(1-\epsilon^i)$, with $\epsilon^i \in [0;1)$; we divide by $(1-\epsilon^i)$ because u^i_{dir} represents realistic costs in our case study. On the contrary, we always kept $\epsilon^i = 2/3$ for each of the $|I|$ agents ($|I|$ denotes the cardinal of the set of agents I) in our previous paper, while Enhan. 1 consists in relaxing this constraint. More precisely, we now verify that agents still have incentives to use the collaborative strategy when ϵ^i varies, or conversely, if every agent behaviour depends on the value of ϵ^i. Notice that $\epsilon^i \to 1$ implements a fully benevolent agent because u^i_{dir} becomes negligible in comparison with u^i_{soc}. In this paper, we only address the case in which $\epsilon^1 = \epsilon^2 = \ldots = \epsilon$ with $\epsilon \in \{1/6, 2/6, 3/6, 4/6, 5/6\}$, which means that all agents have the same level of selfishness ϵ, but that this common level may be different between two of the considered situations. As a consequence, we do not consider the case in which selfish agents cohabit with benevolent agents (the prisonner's dilemma cannot occur), although the presented approach allows doing so. This assumption simplifies the analysis of the simulation outcomes, and we will try to relax it in future work.

Enhan. 2 deals with taking probability distribution for some stochastic parameters into account in replacement of a fixed sequence of numbers, in order to study stochastic models. This means that we generate an instance of the considered distribution, and then, we apply our approach with this fixed sequence of numbers. This is the same method as in our previous paper, except that we repeat the process with other instances of the considered distribution in order to next use statistical tools. In the case study, the stochastic parameter is end-customer demands D^{lumber} and D^{paper} representing the two markets our SC sells in.

Then, our approach may be described as three successive steps, viz **optParam**, **simAllConfig**, and finally **anaSimOut**: (i) **optParam** first optimizes some parameters for every value of ϵ when every agent i uses the same strategy $s^i \in S$ (the strategy set S is unique to all agents), i.e., optimization is only carried our for the strategy profile $(s^i, s^i, \ldots, s^i)$ in order to save computation time. In the case study, these optimized parameters are the inital inventory levels $\{Inv^i_{\text{init}}\}_{i=1}^{i=|I|}$. (ii) Then, **simAllConfig** simulates the SC for the $|S|^{|I|}$ combinations of s^i among the $|I|$ agents with the $\{Inv^i_{\text{init}}\}_{i=1}^{i=|I|}$ found by **optParam**. (iii) Finally, **simAllConfig** builds games in the normal form with the simulation outcomes generated by **simAllConfig**, and looks for the Nash equilibria of these games. Gambit [2] is the set of software tools used to analyze these games. Depending on $\{\epsilon^i\}_{i=1}^{i=|I|}$, different games can be built based from each output of **simAllConfig**.

3. CASE STUDY: APPLICATION TO SUPPLY CHAIN MANAGEMENT

3.1 Adaptation of the approach

We illustrate this approach in the context of the management of a wood SC. We profit from Enhan. 1 to study if selfish as well as benevolent company-agents should collaborate by sharing information concerning market consumption. This shared information is of importance to companies when they place orders, because this information may be distorted by a phenomenon called the "bullwhip effect" [1] that causes an increase of their inventory level and a decrease in their customer service level, and information sharing is the most often proposed solution to the bullwhip effect [4].

Concerning Enhan. 1, we take u^i_{dir} as company-agent i inventory holding cost, and u^i_{soc} as backorder cost, i.e., as stockout cost. Specifically, backorders correspond to orders that were not fulfilled in the past and that have to be shipped as soon as possible, i.e., there are no lost sales in this simulation model because clients wait for product availability. Since backorders are measured in our specific case as negative inventory levels, the calculation of u^i_{dir} and u^i_{soc} are similar, and thus, their aggregation into u^i represents a cost. Of course, every agent tries to minimize its cost u^i (no money can be earned). We think backorders can be used as a measure for social consciousness u^i_{soc} because agents are *not obliged* to consider them when they make decisions (while they always have to take the inventory holding cost into account in u^i_{dir}), but the *goal of the entire* SC should be to satisfy end-customers and avoid thus these backorders. In other words, the SC has not achieved its goal if products are not available by retailers, which is a problem obvious only to retailers. In order to make the other companies internalize the SC goal, we need to have them take backorders into account.

We profit from Enhan. 2 to consider a stochastic market consumption pattern. More precisely, we assume that market demand is a uniform distribution M of integer numbers spread over $\{11, 12, \ldots, 17\}$, and we carry out our experiments with $|M| = 10$ particular instances m_A to m_J of M over the fifty weeks of a simulation run. For instance, m_A corresponds to the two series of fifty numbers $D^{\text{lumber}} = \{16, 13, 16, 11, 15, \ldots, 15, 15, 12, 16, 15\}$ and $D^{\text{paper}} = \{15, 15, 14, 15, 14, \ldots, 13, 14, 14, 11, 13\}$, where the wth number represents the market demand in Week w, e.g., $D^{\text{lumber}}_2 = 13$ means that 13 units are requested to the LumberRetailer in Week 2. We only consider $|M| = 10$ instances of the distribution, because $|M| * |S|^{|I|}$ simulations are required to build the corresponding ten games in the normal form. However, **simAllConfig** carries out $|M| = 10$ times these $|S|^{|I|}$ simulations with one of the ten instances of M in order to build a game. In each of the obtained games, **anaSimOut** changes the ratio ϵ between the u^i_{dir} and u^i_{soc} when Gambit looks for Nash equilibria.

Ordering strategy	Level of collaboration	Type of communication
μ	No collaboration	No communication
β	Information sharing with direct neighbours	Point to point
γ	Information centralization	Bulletin board

Table 1: The three ordering strategies.

3.2 Simulation model

In this case study, we use exactly the same simulation model with $|I| = 6$ agents as in [3]: a LumberRetailer buys from a LumberWholesaler, which buys from a Sawmill, which itself is the only company managing both lumber and paper units. In fact, besides lumbers, the Sawmill can also sell paper units to the PulpMill, which then sells to the PaperWholesaler, which finally sells to the PaperRetailer. Each retailer sells in a different market (lumber and paper), and the Sawmill buys from an infinite source.

Next, each of the six companies i makes its ordering decisions by applying one of $|S| = 3$ ordering strategies s^i called α, β and μ. Once set, s^i remains the same for the fifty weeks of a simulation run. Table 1 outlines these three strategies:

- With ordering strategy μ, company-agents make their decision on their own and can only rely on their incoming orders and shipping to decide what order to place. The only information transmitted with μ are orders and no additional information is shared to help other agents place their own orders. In practice, μ is an (s, S) policy, which is classic in Inventory Management, and in which a company orders $(S - InvP)$ items when the level of its inventory position $InvP$ falls below s. We showed in [3] that taking $s = 0$ and S equals to the current incoming order is optimal.

- There is slight collaboration with strategy β, because agents now transmit the market consumption information to their supplier, which allows every agent to place orders in a more accurate way. From a technical viewpoint, every placed orders is a two-dimension vector, and the quantity ordered is equal to the sum of its two elements: the first element corresponds to the market consumption transmitted by each company to its direct supplier, and the second element represents the products to be ordered in more or in less than the market

Value of ϵ	Nash equilibria
$\epsilon = 1/6$	$(\gamma,\mu,\gamma,\mu,\gamma,\gamma)$, $(\gamma,\gamma,\gamma,\mu,\mu,\gamma)$
$\epsilon = 2/6$	$(\gamma,\mu,\gamma,\mu,\mu,\beta)$, $(\gamma,\gamma,\gamma,\mu,\mu,\gamma)$
$\epsilon = 3/6$	$(\gamma,\beta,\gamma,\mu,\mu,\beta)$, $(\gamma,\gamma,\gamma,\mu,\gamma,\gamma)$
$\epsilon = 4/6$	$(\gamma,\gamma,\gamma,\gamma,\gamma,\gamma)$
$\epsilon = 5/6$	$(\beta,\mu,\beta,\mu,\mu,\beta)$, $(\beta,\beta,\beta,\beta,\beta,\beta)$ $(\beta,\gamma,\beta,\gamma,\beta,\gamma)$, $(\beta,\gamma,\beta,\gamma,\gamma,\beta)$ $(\gamma,\beta,\gamma,\beta,\beta,\beta)$, $(\gamma,\gamma,\gamma,\gamma,\beta,\beta)$

Table 2: Results for Instance m_A.

consumption. Notice that the first element indicates the past market consumption, since this information travels the SC as slowly as orders.

- The highest level of collaboration is implemented in strategy γ, in which retailer-agents write the market consumption information on a bulletin board, and the rest of agents can read this information (this is information centralization). In practice, γ works as β, except that companies take the first element of their two-dimension orders equal to the current market consumption (while it is equal to past data with β).

3.3 Results and analysis

Table 2 enumerates all the Nash equilibria obtained with different values of ϵ with the particular instance m_A of the market demand M. For example, for $\epsilon = 5/6$, there are 6 Nash equilibria, and the first one is the *strategy profile* $(s^1, s^2, s^3, s^4, s^5, s^6) = (\beta,\mu,\beta,\mu,\mu,\beta)$; in this equilibrium, the LumberRetailer ($i = 1$), the LumberWholesaler ($i = 3$) and the Sawmill ($i = 6$) use Strategy β, while all other agents use γ. Note that μ occurs 15 times, β 29 times and γ 40 times in overall Table 2, which represents respectively to frequencies of 20%, 29% and 51%. Therefore, the highest level of collaboration γ occurs in half of the found Nash equilibria.

Next, the question addressed in this paper is to determine at which value of ϵ the Nash equilibria switch from a full-collaborating system (that is, most companies collaborate at the highest level, e.g., $(\gamma,\gamma,\gamma,\beta,\gamma,\gamma)$) to a full non-collaborating one (that is, most companies disagree to collaborate, e.g., $(\mu,\beta,\mu,\ \mu,\mu,\mu)$). To answer this question, we check if the Nash equilibria with low ϵ (selfish agents) often apply the non-collaborative μ, and/or if the Nash equilibria with high ϵ (benevolent agents) frequently use the highly collaborative γ. To see this, we calculate the frequency at which every strategy is used depending on the value of ϵ. That is, we calculate the central column about m_A in Table 3 based on the numbers in Table 2. This calculation is achieved in the following way. In line '$\epsilon = 5/6$' in Table 2, there are 6 equilibria that count a total of 3 μ, 20 β and 13 γ. These three numbers are reported in line $\epsilon = 5/6$ in Table 3 as '$3/36 = 8\% \rightarrow \mu$', which means that 3 of the 36 (=3+20+13) strategies used in a Nash equilibrium are μ. Next, the right column of Table 3 contains the same results, but with the nine other instances m_B to m_J of market consumption. To calculate every frequency in the right column, we take the average of the frequencies obtained with the ten market demands, i.e., these frequencies are not weighted by the number of equilibria: to obtain '$7\% \rightarrow \mu$' with $\epsilon = 5/6$ in the right column of Table 3, the 8% obtained with m_A has the same importance as the percentages obtained with $m_B, \ldots, m_J$, even though they do not represent the same quantity of equilibria.

With both center and right columns in Table 3, we can see that μ only has low occurence frequencies, and that these frequencies do not depend on ϵ. This means that the decision of collaborating does not depend on the level of social consciousness ϵ. The interpretation of this may be that every company-agent is so strongly *dependent* on each other, that they should all collaborate.

Value of ϵ	Frequency of μ, β and γ under Instance m_A	Average frequency of μ, β and γ over Instances m_A to m_J
$\epsilon = \frac{1}{6}$	$4/12 = 33\% \rightarrow \mu$ $0/12 = 0\% \rightarrow \beta$ $8/12 = 67\% \rightarrow \gamma$	$27\% \rightarrow \mu$ $9\% \rightarrow \beta$ $64\% \rightarrow \gamma$
$\epsilon = \frac{2}{6}$	$5/12 = 42\% \rightarrow \mu$ $1/12 = 8\% \rightarrow \beta$ $6/12 = 50\% \rightarrow \gamma$	$22\% \rightarrow \mu$ $6\% \rightarrow \beta$ $72\% \rightarrow \gamma$
$\epsilon = \frac{3}{6}$	$3/12 = 25\% \rightarrow \mu$ $2/12 = 17\% \rightarrow \beta$ $7/12 = 58\% \rightarrow \gamma$	$17\% \rightarrow \mu$ $14\% \rightarrow \beta$ $69\% \rightarrow \gamma$
$\epsilon = \frac{4}{6}$	$0/6 = 0\% \rightarrow \mu$ $0/6 = 0\% \rightarrow \beta$ $6/6 = 100\% \rightarrow \gamma$	$9\% \rightarrow \mu$ $16\% \rightarrow \beta$ $75\% \rightarrow \gamma$
$\epsilon = \frac{5}{6}$	$3/36 = 8\% \rightarrow \mu$ $20/36 = 56\% \rightarrow \beta$ $13/36 = 36\% \rightarrow \gamma$	$7\% \rightarrow \mu$ $43\% \rightarrow \beta$ $50\% \rightarrow \gamma$

Table 3: Relative frequency of the occurence of μ, β and γ in the incurred Nash equilibria.

4. CONCLUSION

This paper studied the impact of social consciousness on individual decision making. As an illustration, we studied if companies in a supply chain would collaborate by sharing the demand when they are more or less benevolent. We found that the benevolence level does not have a great impact, perhaps because companies are tightly linked in our simulation.

5. ACKNOWLEDGMENT

We would like to thank FOR@C, the Research Consortium in E-business in the forest products industry (Université Laval, Québec, Canada), for supporting this research. This work was also partially supported by the National Sciences and Engineering Research Council of Canada (NSERC).

6. REFERENCES

[1] H. L. Lee, V. Padmanabhan, and S. Whang. Information distortion in a supply chain: The bullwhip effect. *Management Science*, 43(4):546–558, 1997.

[2] R. D. McKelvey, A. M. McLennan, and T. L. Turocy. Gambit: Software tools for game theory, version 0.97.0.4, 2004. `http://econweb.tamu.edu/gambit/` (accessed 16 October 2004).

[3] T. Moyaux, B. Chaib-draa, and S. D'Amours. Experimental study of incentives for collaboration in the quebec wood supply game. *IEEE Transactions on Engineering Management*, 2006. (submitted).

[4] D. Simchi-Levi, P. Kaminsky, and E. Simchi-Levi. *Designing and Managing the Supply Chain*. McGraw-Hill Higher Education, 2000.

[5] W. E. Walsh, R. Das, G. Tesauro, and J. O. Kephart. Analyzing complex strategic interactions in multiagent systems. In *Proc. workshop on Game Theoretic and Decision Theoretic agents*, 2002.

[6] M. P. Wellman, J. Estelle, S. Singh, Y. Vorobeychik, C. Kiekintveld, and V. Soni. Strategic interactions in a supply chain game. *Computational Intelligence*, 21(1):1–26, 2005.

Logics for agent systems

Complexity and Succinctness of Public Announcement Logic

Carsten Lutz
Institute for Theoretical Computer Science
TU Dresden, Germany
lutz@tcs.inf.tu-dresden.de

ABSTRACT

There is a recent trend of extending epistemic logic (EL) with dynamic operators that allow to express the evolution of knowledge and induced by knowledge-changing actions. The most basic such extension is public announcement logic (PAL), which is obtained from EL by adding an operator for truthful public announcements. In this paper, we consider the computational complexity of PAL and show that it coincides with that of EL. This holds in the single- and multi-agent case, and also in the presence of common knowledge operators. We also prove that there are properties that can be expressed exponentially more succinct in PAL than in EL. This shows that, despite the known fact that PAL and EL have the same expressive power, there is a benefit in adding the public announcement operator to EL: it exponentially increases the succinctness of formulas without having negative effects on computational complexity.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Theory

Keywords

logics for agent systems, computational complexity, dynamic epistemic logic, public announcement

1. INTRODUCTION

One of the most prominent applications of logic in agent-based systems is reasoning about the knowledge and belief of agents. Although traditionally, epistemic logic (EL) is the basic logical tool for this purpose [14], it has always been clear that EL is too simple for many relevant applications in this area. Most strikingly, basic EL does not include any syntactic or semantic means for representing dynamic and evolutionary aspects of knowledge. Since it is crucial for almost all agent-based systems that the knowledge and belief of agents are subject to change [1], such expressive means are often indispensible. In the literature, there are two dominant approaches to adding dynamics to EL: first, EL can be extended with a temporal component that allows to reason about the evolution of knowledge over time [9]. And second, EL can be extended with dynamic operators that allow to describe the ramifications of knowledge-changing actions. The latter approach is a relatively recent development, and the resulting extensions of EL are often called *dynamic epistemic logics (DELs)* [8].

By now, a large number of DELs has been proposed, and the various proposals differ considerably in expressive power [2, 11, 16, 3, 4, 6, 7]. However, there is a dynamic operator that is included in almost all proposed logics: the *public announcement* operator that has first been introduced in [16]. This operator allows to state that, after some announcement that is publicly made by an outsider to all agents simultaneously, some property holds true. Both the announcement and the property may include epistemic statements such as "agent a knows fact F" or "agent a believes that agent b knows fact F". The announcement is assumed to be truthful, i.e., the person making the announcement does not lie. The effect of the announcement being public is that everybody knows the announced fact, everybody knows that everybody knows it, and so forth. It is interesting to note that the announced fact is not necessarily true anymore *after* the announcement. For example, this is the case if the announced fact is "agent a knows fact F, but agent b doesn't know that" (because, after the announcement, agent b knows that agent a knows F).

Existing research about the public announcement operator has mainly concentrated on the expressiveness and axiomatics of the obtained extensions of EL. For example, it is known that EL with the public announcement operator has the same expressive power as EL without it.[1] However, performing *actual reasoning* is of great importance when applying DELs in agent-based systems, and thus, the computational properties of such logics need to be analyzed. While the computational complexity of model checking DELs is easily pinpointed, not much is known about deciding satisfiability and validity (henceforth called *reasoning*). Therefore, the purpose of the current paper is to *analyze the computational complexity of reasoning in epistemic logics ex-*

[1]This should not be taken as an indication that the operator is not worth studying; see below.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

tended with the public announcement operator. Since computational complexity, expressiveness, and succinctness issues turn out to be intimately and subtly related to computational complexity in the considered logics, we include the latter in our analysis.

We start our investigation with *public announcement logic (PAL)*, the extension of basic EL with the public announcement operator. As noted above, the expressive power of PAL is known to be identical to the expressive power of EL: there exists an equivalence-preserving translation from the former to the latter [16, 8]. Computationally, this translation is only moderately useful: it yields decidability of reasoning in PAL, but it does not produce tight complexity bounds due to an exponential blowup in formula size. More precisely, the known translation yields upper complexity bounds for PAL that are identical to those for EL, but raised by one exponential. In contrast, the best known lower bounds are the ones from EL.

In this paper, we show that the existing upper bounds can be improved by one exponential, and thus reasoning in PAL is of the same complexity as reasoning in EL. To this end, we first propose a novel, equivalence-preserving translation from PAL to EL. Like the existing one, this translation induces an exponential blowup in formula size. The advantage of the new translation is that it can easily be modified such that it becomes only satisfiability-preserving, but avoids the exponential blowup in formula size. The modified translation takes formulas of single-agent PAL to formulas of single-agent EL, and formulas of multi-agent PAL to formulas of multi-agent EL extended with an "everybody knows" operator. Thus, it can be used to prove that (i) single-agent PAL is NP-complete, and (ii) multi-agent PAL is PSPACE-complete. We then extend our equivalence-preserving translation and its satisfiability-preserving modification to PAL extended with (two variants of) common knowledge. In this case, the target language of the translation is propositional dynamic logic (PDL), and we obtain EXPTIME-completeness results.

Due to the fact that PAL and EL are equally expressive and of the same computational complexity, one may be tempted to think that the addition of the public announcement operator to EL is only syntactic sugar and not of much interest. However, it has been convincingly argued in [3, 8] that PAL is a much more intuitive and natural formalism for talking about the dynamics of knowledge than EL. In this paper, we add another, more concrete advantage of PAL: we prove that there are properties that can be expressed exponentially more succinct in PAL than in EL. Thus, the public announcement operator contributes to the succinctness of the logic, and this exponential increase in succinctness is not even penalised by an increase in computational complexity. Of course, our succinctness result also implies that one cannot hope to find an *equivalence-preserving* translation from PAL to EL that avoids an exponential blowup. A limitation of our (current) succinctness result is that it applies only to the class of all Kripke structures, and not to the class of epistemic structures.

2. PUBLIC ANNOUNCEMENT LOGIC

Let PL and N be countable sets of *propositional letters* and *agents*. The formulas of *public announcement logic with common knowledge (PALC)* are built according to the following syntax rule:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid K_a\varphi \mid [\varphi]\psi \mid C_A\varphi$$

where p ranges over PL, a ranges over N, and A ranges over 2^N. The operator $[\varphi]\psi$ is the *public announcement operator*: $[\varphi]\psi$ states that, after φ is publically announced, ψ holds true. A detailed introduction to PALC and how it can be used to model the dynamics of knowledge can be found in [3, 8]. Here, we confine ourselves to a simple example: the formula

$$K_a(K_b\mathsf{secret} \wedge [K_b\mathsf{secret}]K_c\mathsf{leaked}_{a,b}),$$

with secret and $\mathsf{leaked}_{a,b}$ propositional letters, states the following: agent a knows that agent b knows the secret, and if this is publically announced, then agent c will know that agent a has leaked the secret to b.

By dropping the *common knowledge operator* $C_A\varphi$ from PALC, we get formulas of *public announcement logic (PAL)*. By dropping the public announcement operator from PAL, we get formulas of *epistemic logic (EL)*. As usual, we use $\varphi \vee \psi$ as an abbreviation for $\neg(\neg\varphi \wedge \neg\psi)$, $\varphi \rightarrow \psi$ for $\neg\varphi \vee \psi$, $\widehat{K}_a\varphi$ for $\neg K_a\neg\varphi$, $\langle\varphi\rangle\psi$ for $\neg[\varphi]\neg\psi$, and $\top$ for $p \vee \neg p$, with p an arbitrary (but fixed) propositional letter. When talking of *single-agent PAL or EL*, we assume that N is a singleton. In this case, we drop the index $\cdot_a$ from $K_a\varphi$ and $\widehat{K}_a\varphi$.

The semantics of PALC is defined by means of Kripke structures. An *epistemic model* (or *model* for short) is a triple $\mathcal{M} = (S, \sim, V)$ with S a non-empty set of states, $\sim = (\sim_a)_{a\in N}$ a family of equivalence relations on S, and V a function assigning a set of states $V(p) \subseteq S$ with each $p \in \mathsf{PL}$. Given a model $\mathcal{M} = (S, \sim, V)$ and an $s \in S$, we define:

$$\begin{array}{rcl}
\mathcal{M}, s \models p & \text{iff} & s \in V(p) \\
\mathcal{M}, s \models \neg\varphi & \text{iff} & \mathcal{M}, s \not\models \varphi \\
\mathcal{M}, s \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, s \models \varphi \text{ and } \mathcal{M}, s \models \psi \\
\mathcal{M}, s \models K_a\varphi & \text{iff} & s \sim_a t \text{ implies } \mathcal{M}, t \models \varphi, \text{ for all } t \in S \\
\mathcal{M}, s \models [\varphi]\psi & \text{iff} & \mathcal{M}, s \models \varphi \text{ implies } \mathcal{M}|\varphi, s \models \psi \\
\mathcal{M}, s \models C_A\varphi & \text{iff} & s \,(\bigcup_{a\in A} \sim_a)^* \, t \text{ implies } \mathcal{M}, t \models \varphi, \\
 & & \qquad\qquad \text{for all } t \in S
\end{array}$$

where $\cdot^*$ denotes the reflexive-transitive closure operator on binary relations, and the model $\mathcal{M}|\psi := (T, \approx, U)$ is defined as follows:

$$\begin{array}{rcl}
T & := & \{t \in S \mid \mathcal{M}, t \models \psi\} \\
\approx_a & := & \sim_a \cap (T \times T) \\
U(p) & := & V(p) \cap T.
\end{array}$$

It is easily checked that the semantics of the derived modalities is as follows:

$$\begin{array}{rcl}
\mathcal{M}, s \models \widehat{K}_a\varphi & \text{iff} & s \sim_a t \text{ and } \mathcal{M}, t \models \varphi, \text{ for some } t \in S \\
\mathcal{M}, s \models \langle\varphi\rangle\psi & \text{iff} & \mathcal{M}, s \models \varphi \text{ and } \mathcal{M}|\varphi, s \models \psi.
\end{array}$$

Given a model $\mathcal{M}$, we use $S(\mathcal{M})$ to denote the set of states of $\mathcal{M}$. It is not too difficult to see that, by fixing the first argument, the public announcement operator can be conceived as a unary normal modal operator (in the sense of, e.g., [5]) that behaves in many ways like a modal operator for a functional accessibility relation. For example, $\langle\varphi\rangle\psi$ implies $[\varphi]\psi$.

$$
\begin{array}{rcl}
[\varphi]p & \rightsquigarrow & \varphi \to p \\
[\varphi](\psi \wedge \chi) & \rightsquigarrow & [\varphi]\psi \wedge [\varphi]\chi \\
[\varphi]\neg\psi & \rightsquigarrow & \varphi \to \neg[\varphi]\psi \\
[\varphi]K_a\psi & \rightsquigarrow & \varphi \to K_a[\varphi]\psi
\end{array}
$$

Figure 1: The standard translation from PAL to EL.

3. EXPRESSIVITY AND SUCCINCTNESS

As first observed by Plaza [16], the basic epistemic logic with public announcements PAL has exactly the same expressive power as EL. Indeed, by exhaustively applying the rewrite rules in Figure 1, one can convert every PAL formula into an equivalent EL formula [8]. For the purposes of this paper, however, it is convenient to work with a translation that is different from this standard one. To introduce the new translation, we need a bit of notation: for φ a PAL formula and $\sigma = \varphi_1 \cdots \varphi_k$ a finite sequence of PAL formulas, we use

- $|\varphi|$ to denote the length of φ, i.e. the number of symbols needed to write down φ, including symbols such as "[" and "]"; likewise, $|\sigma|$ denotes $|\varphi_1| + \cdots + |\varphi_k|$;
- $\mathcal{M}|\sigma$ as an abbreviation for $(((\mathcal{M}|\varphi_1)|\varphi_2)\cdots|\varphi_k)$ with $\mathcal{M}|\varepsilon = \mathcal{M}$;
- $\mathsf{pre}(\sigma)$ to denote the set of all true prefixes of σ including the empty sequence ε;
- for each $\nu \in \mathsf{pre}(\sigma)$, ν/σ to denote the leftmost symbol of σ that is not in ν.

We now define, for every PAL formula φ and every finite sequence of PAL formulas σ, an EL formula φ^σ. The definition of the formulas φ^σ proceeds by induction on $|\varphi| + |\sigma|$ as shown in Figure 2. In the $K_a\varphi$ case, the conjunction collapses to true if $\sigma = \varepsilon$. To see that we really do induction on $|\varphi| + |\sigma|$, note that the symbols "[" and "]" contribute to the size of the left-hand side of the last line.

The most important property of the formulas φ^σ is given by the following lemma. It can be proved by induction on $|\varphi| + |\sigma|$, for details consult the appendix.

Lemma 1. *For all models $\mathcal{M} = (S, \sim, V)$, PAL formulas φ, finite sequences of PAL formulas σ, and states $s \in S(\mathcal{M}|\sigma)$, we have $\mathcal{M}, s \models \varphi^\sigma$ iff $\mathcal{M}|\sigma, s \models \varphi$.*

Lemma 1 clearly implies that each PAL formula φ is equivalent to the EL formula φ^ε, which gives us the new translation. It can easily be seen that this translation usually produces different formulas than the standard translation given in Figure 1. For example, the result of translating $[p_1][p_2]\cdots[p_k]q$ with our translation results in the formula

$$p_1 \to (p_2 \to (\cdots \to (p_k \to q))),$$

which is of length $\mathcal{O}(k)$. In contrast, the standard translation produces a highly redundant formula of length $2^{\mathcal{O}(k)}$. Thus, there are formulas on which our translation is exponentially more succinct. In general, however, the new translation does not avoid an exponential blowup in formula size. See Theorem 2 below for example formulas.

Given the exponential blowup induced by both translations, it is a natural question whether the exponential blowup in translating PAL to EL can be avoided at all. We answer this question to the negative: PAL is exponentially more succinct than epistemic logic if there are at least two agents and we assume *unrestricted models*, i.e., models whose relations are not required to be equivalence relations. Our aim is to prove the following theorem.

$$
\begin{array}{rcl}
p^\sigma & := & p \\
(\neg\varphi)^\sigma & := & \neg\varphi^\sigma \\
(\varphi \wedge \psi)^\sigma & := & \varphi^\sigma \wedge \psi^\sigma \\
(K_a\varphi)^\sigma & := & K_a(\bigwedge_{\nu \in \mathsf{pre}(\sigma)} (\nu/\sigma)^\nu \to \varphi^\sigma) \\
([\varphi]\psi)^\sigma & := & \varphi^\sigma \to \psi^{\sigma\cdot\varphi}
\end{array}
$$

Figure 2: The new translation.

Theorem 2. *For $i \geq 0$, define*

- $\varphi_0 := \top$;
- $\varphi_{i+1} := \langle\langle\varphi_i\rangle\widehat{K}_a\top\rangle\widehat{K}_b\top$.

On unrestricted models, every EL formula ψ equivalent to φ_i is of length at least 2^i, for all $i \geq 0$.

Define a sequence of EL formulas $\psi_0, \psi_0', \psi_1, \psi_1', \psi_2, \ldots$ as follows:

- $\psi_0 := \top$;
- $\psi_i' := \psi_i \wedge \widehat{K}_a\psi_i$;
- $\psi_{i+1} := \psi_{i-1}' \wedge \widehat{K}_b\psi_{i-1}'$.

Using the translation from PAL to EL given above, which also applies to the case of unrestricted models, it is straightforward to prove by induction on i that, for $i \geq 0$, φ_i is equivalent to ψ_i. Thus, for proving Theorem 2 it suffices to prove that every EL formula χ that is equivalent to ψ_i on unrestricted models is of length at least 2^i, for all $i \geq 0$.

In what follows, a *path set* is a subset of N^* (where N is the set of agents). For φ an EL formula φ over the set of agents N, we define the *path-set* P_φ of φ by structural induction as follows:

- $P_p := \{\varepsilon\}$;
- $P_{\neg\varphi} := P_\varphi$;
- $P_{\varphi\wedge\psi} := P_\varphi \cup P_\psi$;
- $P_{K_a\varphi} := \{\varepsilon\} \cup \{aw \mid w \in P_\varphi\}$.

Now, let χ be an EL formula that is equivalent to ψ_i, for some $i > 0$. We show the following:

$$P_{\psi_i} \subseteq P_\chi. \qquad (*)$$

To prove $(*)$, assume to the contrary that there is a $\widehat{w} \in P_{\psi_i} \setminus P_\chi$. Define a model $\mathcal{M} = (S, \sim, V)$ as follows:

- $S = P_{\psi_i}$;
- the relation $\sim_\sigma$, $\sigma \in N$, is defined by setting $w \sim_\sigma v$ if $v = w\sigma$, for $w, v \in S$;
- $V(p) = \emptyset$ for all $p \in \mathsf{PL}$.

$$
\begin{array}{rcl}
R(a) & := & \{(\varepsilon, a)\} \\
R(\neg\varphi) & := & R(\varphi) \cup \{(\varepsilon, \neg\varphi)\} \\
R(\varphi \wedge \psi) & := & R(\varphi) \cup R(\psi) \cup \{(\varepsilon, \varphi \wedge \psi)\} \\
R(K_i\varphi) & := & R(\varphi) \cup \{(\varepsilon, K_i\varphi)\} \\
R([\varphi]\psi) & := & R(\varphi) \cup \{(\varphi \cdot \sigma, \vartheta) \mid (\sigma, \vartheta) \in R(\psi)\} \\
 & & \cup \{(\varepsilon, [\varphi]\psi)\}
\end{array}
$$

Figure 3: The relevant pairs.

$$
\begin{array}{rcl}
B^\sigma_q & := & p^\sigma_q \leftrightarrow q \\
B^\sigma_{\neg\varphi} & := & p^\sigma_{\neg\varphi} \leftrightarrow \neg p^\sigma_\varphi \\
B^\sigma_{\varphi\wedge\psi} & := & p^\sigma_{\varphi\wedge\psi} \leftrightarrow (p^\sigma_\varphi \wedge p^\sigma_\psi) \\
B^\sigma_{K_i\varphi} & := & p^\sigma_{K_i\varphi} \leftrightarrow K_i(\bigwedge_{\nu\in \mathsf{pref}(\sigma)} p^\nu_{\nu/\sigma} \rightarrow p^\sigma_\varphi) \\
B^\sigma_{[\varphi]\psi} & := & p^\sigma_{[\varphi]\psi} \leftrightarrow (p^\sigma_\varphi \rightarrow p^{\sigma\cdot\varphi}_\psi)
\end{array}
$$

Figure 4: The biimplications B^σ_φ.

A second model $\mathcal{M}' = (S', \sim', V')$ is defined as the restriction of $\mathcal{M}$ to the set of states

$$S' := S \setminus \{w \in N^* \mid w = \widehat{w}w' \text{ for some } w' \in N^*\}.$$

It is not too hard to show that $\mathcal{M}$ is a model of ψ_i, but $\mathcal{M}'$ is not. Details are left to the reader.

Lemma 3. *$\mathcal{M}, \varepsilon \models \psi_i$ and $\mathcal{M}', \varepsilon \not\models \psi_i$.*

We now show that χ cannot distinguish ε in $\mathcal{M}$ from ε in $\mathcal{M}'$, i.e., $\mathcal{M}, \varepsilon \models \chi$ iff $\mathcal{M}', \varepsilon \models \chi$. This clearly is a contradiction to Lemma 3 and the fact that χ is equivalent to φ_i. Thus, we have established $(*)$. The fact that χ cannot distinguish ε in $\mathcal{M}$ from ε in $\mathcal{M}'$ is an immediate consequence of the following lemma, which is proved in the appendix.

Lemma 4. *For all $s \in S'$ and $\varphi \in \mathsf{sub}(\chi)$ such that*

$$\{sw \mid w \in P_\varphi\} \subseteq P_\chi,$$

we have $\mathcal{M}, s \models \varphi$ iff $\mathcal{M}', s \models \varphi$.

Finally, $|\chi| \geq 2^i$ is a consequence of $(*)$ together with the following lemma and the fact that, as is easily proved by structural induction, we have $|\varphi| \geq |P_\varphi|$ for all formulas φ.

Lemma 5. *For all $i \geq 0$, $|P_{\psi_i}| \geq 2^i$.*

Proof. It is straightforward to prove by induction on i that, for all $i \geq 0$, we have $\{a, b\}^i \subseteq P_{\psi_i}$. □

This finishes the proof of Theorem 2. We believe that PAL is also exponentially more succinct than EL on epistemic structures, but leave the proof as an open problem.

4. UPPER BOUNDS FOR PAL WITHOUT COMMON KNOWLEDGE

Given the succinctness of PAL established in the previous section, the question arises whether a penalty has to be paid for this succinctness in terms of computational complexity: is reasoning in PAL more expensive than reasoning in epistemic logic? Interestingly, this is not the case. We show that satisfiability in PAL is NP-complete in the single-agent case and PSpace-complete in the multi-agent case, just as in epistemic logic [13]. Lower bounds are immediate since PAL contains EL as a fragment. The idea for obtaining the upper bounds is to convert the equivalence-preserving, but exponential translation given in Section 3 into a satisfiability-preserving and polynomial translation. We start with the single-agent case.

4.1 Reducing Single-Agent PAL

We start with introducing some relevant notions. As these notions will also be useful for dealing with multi-agent PAL, we do not restrict ourselves to single-agent formulas here. Let φ be a PAL formula. With $\mathsf{sub}(\varphi)$, we denote the set of all subformulas of φ, including φ. With $\Sigma(\varphi)$, we denote the set of all pairs (σ, ψ), where $\psi \in \mathsf{sub}(\varphi)$ and σ is a finite (and possibly empty) sequence of formulas from $\mathsf{sub}(\varphi)$. The subset $R(\varphi) \subseteq \Sigma(\varphi)$ of *relevant pairs for* φ is defined inductively as in Figure 3.

Intuitively, $R(\varphi)$ gives us a representation of the subordinate translations that occur when inductively translating the PAL formula φ to the EL formula φ^ε. The central observation is that, while there are exponentially many *calls* to subordinate translations ψ^σ while translating φ into φ^ε, there are only polynomially many sub-translations with different arguments ψ, σ. Using structural induction on φ, it is easy to show that the number of relevant pairs is polynomial in $|\varphi|$ and that each pair in $R(\varphi)$ is of size polynomial in $|\varphi|$.

Lemma 6. *For all PAL formulas φ, we have the following:*

1. *$|R(\varphi)| \leq |\varphi|$;*
2. *for all $(\sigma, \psi) \in R(\varphi)$, the length of the sequence σ is bounded by $|\varphi|$.*

We now convert the equivalence-preserving and exponential translation from PAL to EL into a satisfiability-preserving polynomial one. Intuitively, we introduce a propositional letter p^ψ_σ for each subordinate translation ψ^σ and enforce that p^ψ_σ is true precisely where ψ^σ is true. This can be done in an incremental fashion without actually producing the (exponentially long) formulas of ψ^σ.

Let φ_0 be a single-agent PAL formula whose satisfiability is to be decided. We introduce a set of propositional letters $L_{\varphi_0} := \{p^\sigma_\varphi \mid (\sigma, \varphi) \in R(\varphi_0)\}$. W.l.o.g., assume that no letter from L_{φ_0} occurs in φ_0. For every $(\sigma, \varphi) \in R(\varphi_0)$, we define a biimplication B^σ_φ as in Figure 4. Note that the right-hand side of the biimplications is derived in a straightforward way from the equivalence-preserving translation given in Figure 2. Now define

$$\varphi_0^* := p^\varepsilon_{\varphi_0} \wedge \bigwedge_{(\sigma,\varphi)\in R(\varphi_0)} KB^\sigma_\varphi.$$

Observe that $|\varphi_0^*|$ is polynomial in φ_0: by Point 2 of Lemma 6, $|B^\sigma_\varphi|$ is linear in $|\varphi_0|$ for each $(\sigma, \varphi) \in R(\varphi_0)$. By Point 1 of Lemma 6, $|\varphi_0^*|$ is thus quadratic in $|\varphi|$. Clearly, φ_0^* is an EL formula. As the next lemma shows, we have obtained a satisfiability-preserving reduction as desired. A proof can be found in the appendix.

Lemma 7. *φ_0 is satisfiable iff φ_0^* is satisfiable.*

Taking together Lemma 7, the fact that $|\varphi_0^*|$ is quadratic in $|\varphi_0|$, and the known NP upper bound of single-agent EL (i.e., modal S5 [9]), we obtain an NP upper bound for single-agent PAL.

Theorem 8. *Satisfiability in single-agent PAL is* NP*-complete.*

4.2 Reducing Multi-Agent PAL

The general idea for obtaining a PSPACE-upper bound for multi-agent PAL is to proceed analogously to the single-agent case. However, there is a complication: in the second conjunct of the formula φ_0^* of Lemma 7, we use K as a *master modality* that allows us to access all states that are (directly or indirectly) reachable from some given state. Alas, a master modality is not available in multi-agent PAL. For this reason, we reduce multi-agent PAL to the extension of EL with the *everybody knows operator*. The addition of this operator provides us with a restricted version of the master modality that is sufficient for our purposes.[2] Since adding the everybody knows operator to EL does not increase the computational complexity, we obtain the desired PSPACE upper bound.

Epistemic logic is extended to *epistemic logic with everybody knows (ELE)* by adding the *everybody knows operator* $E_A\varphi$, where A is a finite set of agents. The semantics of the new operator is as follows:

$$\mathcal{M}, s \models E_A\varphi \text{ iff } s \sim_a t \text{ implies } \mathcal{M}, t \models \varphi, \text{ for all } t \in S \text{ and all } a \in A.$$

PSPACE-completeness of satisfiability in multi-agent ELE is folklore. For the sake of completeness, we prove it in the extended version of this paper [15].

We now reduce satisfiability in multi-agent PAL to satisfiability in ELE. Let φ_0 be the PAL formula whose satisfiability is to be decided, and let A be the set of agents used in φ_0. As in the single agent case, we introduce a set of propositional letters $L_{\varphi_0} := \{p_\varphi^\sigma \mid (\sigma, \varphi) \in R(\varphi_0)\}$ that are assumed to be disjoint from the propositional letters used in φ_0. The *modal depth* $\mathsf{md}(\varphi)$ of a PAL formula φ is defined inductively in the usual way:

$$\begin{array}{rcl} \mathsf{md}(p) & := & 0 \\ \mathsf{md}(\neg\varphi) & := & \mathsf{md}(\varphi) \\ \mathsf{md}(\varphi \wedge \psi) & := & \mathsf{md}([\varphi]\psi) := \max(\mathsf{md}(\varphi), \mathsf{md}(\psi)) \\ \mathsf{md}(K_a\varphi) & := & \mathsf{md}(\varphi) + 1. \end{array}$$

Define an ELE formula

$$\varphi_0^* := p_{\varphi_0} \wedge \bigwedge_{j \le \mathsf{md}(\varphi_0)} \bigwedge_{(\sigma,\varphi) \in R(\varphi_0)} E_A^j B_\varphi^\sigma,$$

where B_φ^σ is the biimplication as defined in Figure 4, $E_A^j\varphi$ is an abbreviation for the j-fold nesting $E_A \cdots E_A\varphi$ if $j > 0$, and $E_A^0\varphi$ is simply φ. The proof of the following lemma is analogous to the proof of Lemma 7. The only difference concerns the "if" direction, where we cannot assume anymore that the accessibility relations are universal relations. To compensate for this, it is not hard to argue that the second conjunct of φ_0^* ensures that B_φ^σ is satisfied at all relevant states in models of φ_0^*, for all $(\sigma, \varphi) \in R(\varphi_0)$. Details are left to the reader.

Lemma 9. *φ_0 is satisfiable iff φ_0^* is satisfiable.*

By Lemma 6 and the fact that $\mathsf{md}(\varphi_0)$ is bounded by $|\varphi_0|$, $|\varphi_0^*|$ is polynomial in $|\varphi_0|$. From Lemma 9 and PSPACE-completeness of satisfiability in ELE, we thus obtain the following result.

Theorem 10. *Satisfiability in multi-agent PAL is* PSPACE*-complete.*

5. UPPER BOUNDS FOR PAL WITH COMMON KNOWLEDGE

It is known that (multi-agent) PALC is more expressive than the standard epistemic language extended with a common knowledge operator (ELC). For example, it is shown in [2] that there is no formula of ELC that is equivalent to the PALC formula $[p]\neg C^*\neg q$. Thus, to obtain complexity results for PALC we cannot proceed analogous to the PAL case, i.e., first exhibit an equivalence-preserving translation to the logic obtained by dropping public announcements, and then use this translation to devise a decision procedure. The solution is to establish an equivalence-preserving translation from PALC to a more expressive language than ELC: propositional dynamic logic (PDL). Since satisfiability in both ELC (which is a fragment of PALC) and PDL is EXPTIME-complete, we can then continue as in the previous section.

Recall that PDL formulas and programs are built according to the following syntax rules:[3]

$$\begin{array}{rcl} \varphi & ::= & p \mid \varphi \vee \varphi \mid \neg\varphi \mid [[\alpha]]\varphi \\ \alpha & ::= & a \mid \alpha \cup \alpha \mid \alpha;\alpha \mid \alpha^* \mid \varphi? \end{array}$$

where p ranges over propositional letters and a over agents (usually called *atomic programs* in PDL). We define the semantics of PDL based on epistemic models by simultaneously definining the consequence relation together with accessibility relations $\sim_\alpha$ for complex programs α. Let $\mathcal{M} = (S, \sim, V)$ be an epistemic model. Then, for all $s, t \in S$, we have:

$$\begin{array}{lll} s \sim_{\alpha\cup\beta} t & \text{iff} & s \sim_\alpha t \text{ or } s \sim_\beta t \\ s \sim_{\alpha;\beta} t & \text{iff} & s \sim_\alpha u \text{ and } u \sim_\beta t \text{ for some } u \in S \\ s \sim_{\alpha^*} t & \text{iff} & \exists u_0, \ldots, u_n \in S,\ n \ge 0, \text{ such that} \\ & & s = u_0,\ t = u_n, \text{ and } u_i \sim_\alpha u_{i+1} \text{ for } i < n \\ s \sim_{\varphi?} t & \text{iff} & s = t \text{ and } \mathcal{M}, s \models \varphi \\ \mathcal{M}, s \models [[\alpha]]\varphi & \text{iff} & s \sim_\alpha t \text{ implies } \mathcal{M}, t \models \varphi, \text{ for all } t \in S \end{array}$$

where the clauses for the Booleans are as in Section 2. To distinguish PDL on unrestricted models from PDL on epistemic models, we will from now on call the latter ePDL.

Figure 5 defines, for each PALC formula φ and finite sequence of PALC formulas σ, an ePDL formula φ^σ. Observe that the only difference to Figure 2 is the additional line dealing with the common knowledge operator. As in the case of PAL, we use φ^ε as the ePDL-translation of the PALC formula φ. The following lemma shows that φ^ε is

[2] Adding *everybody knows* does not actually increase the expressive power. However, it allows us to formulate the restricted master modality exponentially more succinct than in standard EL, which is crucial for obtaining a polynomial translation.

[3] We use the notation $[[\alpha]]\varphi$ instead of the more familiar $[\alpha]\varphi$ to distinguish this operator from the public announcement operator.

$$
\begin{array}{rcl}
p^\sigma & := & p \\
(\neg\varphi)^\sigma & := & \neg\varphi^\sigma \\
(\varphi \wedge \psi)^\sigma & := & \varphi^\sigma \wedge \psi^\sigma \\
(K_i\varphi)^\sigma & := & K_i(\bigwedge_{\nu \in \mathsf{pre}(\sigma)} (\nu/\sigma)^\nu \to \varphi^\sigma) \\
(C_A\varphi)^\sigma & := & [[(\bigcup_{a\in A} a; (\bigwedge_{\nu \in \mathsf{pre}(\sigma)} (\nu/\sigma)^\nu)?)^*]]\varphi^\sigma \\
([\varphi]\psi)^\sigma & := & \varphi^\sigma \to \psi^{\sigma\cdot\varphi}
\end{array}
$$

Figure 5: The formulas φ^σ for PALC.

indeed equivalent to φ. The proof is a straightforward extension of the proof of Lemma 1. Details are given in the appendix.

Lemma 11. *For all models $\mathcal{M} = (S, \sim, V)$, PALC formulas φ, finite sequences of PALC formulas σ, and states $s \in S(\mathcal{M}|\sigma)$, we have $\mathcal{M}, s \models \varphi^\sigma$ iff $\mathcal{M}|\sigma, s \models \varphi$.*

Our aim is to show ExpTime-completeness of PALC. As already the fragment ELC of PALC is ExpTime-hard [10], it remains to establish an upper bound. In the following, we prove this bound by a satisfiability-preserving and polynomial reduction to ePDL. Let us first fix the complexity of this logic.

Lemma 12. *Satisfiability in ePDL is* ExpTime*-complete.*

Proof. The lower bound stems from ELC [10]. The upper bound is easily obtained by a reduction to converse-PDL, which is ExpTime-complete [10, 17, 12]: to decide whether a PDL formula φ is satisfiable in an epistemic model, simply replace all atomic programs a in φ with $(a \cup a^-)^*$, and check whether the resulting IPDL formula is satisfiable in an unrestricted model. □

We may now reduce satisfiability in PALC to satisfiability in ePDL using the same approach as in the previous section. Let φ_0 be the PALC formula whose satisfiability is to be decided. The definition of the set of relevant pairs $R(\varphi_0)$ can be extended to PALC formulas by adding the clause

$$R(C_A\varphi) \quad := \quad R(\varphi) \cup \{(\varepsilon, C_A\varphi)\}.$$

As usual, we then introduce a set of propositional letters $L_{\varphi_0} := \{p^\sigma_\varphi \mid (\sigma, \varphi) \in R(\varphi_0)\}$ that are disjoint from the propositional letters used in φ_0. Let $a_1, \ldots, a_k$ be the agents referred to in φ_0. We define an ePDL-formula

$$\varphi_0^* := p_{\varphi_0} \wedge \bigwedge_{(\sigma,\varphi)\in R(\varphi_0)} [[(a_1 \cup \cdots \cup a_k)^*]]B^\sigma_\varphi,$$

where the biimplications B^σ_φ are defined as in Figure 4, with the following additional clause for common knowledge:

$$B^\sigma_{C_A\varphi} \quad := \quad p^\sigma_{C_A\varphi} \leftrightarrow [[(\bigcup_{a\in A} a; (\bigwedge_{\nu \in \mathsf{pre}(\sigma)} p^\nu_{\nu/\sigma})?)^*]]p^\sigma_\psi.$$

The proof of the following lemma is analogous to the proof of Lemmas 7 and 9.

Lemma 13. *φ_0 is satisfiable iff φ_0^* is satisfiable.*

Since Lemma 6 can easily be extended to the PALC case, $|\varphi_0^*|$ is polynomial in $|\varphi_0|$. From Lemmas 12 and 13, we thus obtain the following result.

Theorem 14. *Satisfiability in PALC is* ExpTime*-complete.*

In [4], van Benthem et al. introduce a generalization of the common knowledge operator $C_A(\varphi, \psi)$ that is called the *relativized common knowledge operator* and has the following semantics:

$$\mathcal{M}, w \models C_A(\varphi, \psi) \quad \text{iff} \quad \mathcal{M}, w \models \varphi \to [[(\bigcup_{a\in B} a; \varphi?)^*]]\psi,$$

where the formula on the right-hand side is to be read as a PDL formula. Clearly, $C_A\varphi$ is equivalent to $C_A(\top, \varphi)$. Intuitively, relativized common knowledge resembles the until operator from temporal logic.

Let PAL-RC denote the variant of PALC in which common knowledge is replaced with relativized common knowledge, and let EL-RC be the extension of EL with the relativized common knowledge operator. The introduction of the new operator is a reaction to the fact that there exists no equivalence-preserving translation from PALC to ELC, i.e, to the logic obtained by dropping the public announcement operator. By moving from common knowledge to the stronger relativized common knowledge, such a translation is recovered: as shown in [4], there exists an equivalence-preserving translation from PAL-RC to EL-RC.

We can easily modify the translation given in Figure 5 such that it maps formulas of PAL-RC to formulas of ePDL:

$$C_A(\varphi, \psi)^\sigma := \varphi^\sigma \to [[(\bigcup_{a\in A} a; (\bigwedge_{\nu \in \mathsf{pre}(\sigma)} (\nu/\sigma)^\nu)?; \varphi^\sigma?)^*]]\psi^\sigma$$

Then, we can continue along the lines of the reduction from PALC to ePDL to obtain a reduction from PAL-RC to ePDL. As in the PALC case, we obtain the following theorem.

Theorem 15. *Satisfiability in PAL-RC is* ExpTime*-complete.*

6. CONCLUSION

We have analyzed the succinctness and computational complexity of (several variations of) epistemic logic extended with a public announcement operator. The main results are that, first, there are certain properties that can be expressed exponentially more succinct in PAL than in EL, and, second, despite this succinctness the computational complexity of PAL and EL coincides. As future work, it would be nice to prove the exponential succinctness of PAL as compared with EL also on epistemic structures. It is not clear whether a relatively simple argument such as the one given in Section 3 can be used in this case. Moreover, it would be interesting to analyze the computational complexity of other announcement operators, in particular of private announcements as considered, e.g., in [2, 11].

Acknowledgements

I'd like to thank Johan van Benthem, whose talk in Dresden started this work. I am also grateful to the following people for explanations and clarifying discussions: Johan van Benthem, Jelle Gerbrandy, Mitch Harris, Wiebe van der Hoek, and Mehrnoosh Sadrzadeh.

7. REFERENCES

[1] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change. *Journal of Symbolic Logic*, 50:510–530, 1985.

[2] A. Baltag, L. S. Moss, and S. Solecki. The logic of public announcements and common knowledge for distributed applications (extended abstract). In I. Gilboa, editor, *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the Seventh Conference (TARK 1998)*, pages 43–56. Morgan Kaufmann, 1998.

[3] J. van Benthem. Logics for information update. In J. van Benthem, editor, *Proceedings of TARK-VIII*, pages 51–88. Morgan Kaufmann, 2001.

[4] J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. Unpublished Manuscript.

[5] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[6] H. van Ditmarsch, W. van der Hoek, and B. Kooi. Dynamic epistemic logic with assignment. In F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. Singh, and M. Wooldridge, editors, *Proceedings of AAMAS 2005 (Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems)*, pages 141–148. ACM Inc., 2005.

[7] H. van Ditmarsch, W. van der Hoek, and B. Kooi. Public announcements and belief expansion. In R. Schmidt, I. Pratt-Hartmann, M. Reynolds, and H. Wansing, editors, *Advances in Modal Logic, Volume 5*, pages 335–346. King's College Publications, 2005.

[8] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Kluwer academic publishers. To appear.

[9] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[10] M. J. Fischer and R. E. Ladner. Propositional modal logic of programs. In *Conference record of the ninth annual ACM Symposium on Theory of Computing*, pages 286–294. ACM Press, 1977.

[11] J. Gerbrandy. *Bisimulations on Planet Kripke*. ILLC Dissertation Series, Amsterdam, 1999.

[12] D. Giacomo and Massacci. Combining deduction and model checking into tableaux and algorithms for converse-PDL. *Information and Computation*, 162, 2000.

[13] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–380, 1992.

[14] J. Hintikka. *Knowledge and Belief*. Cornell University Press, Ithaca, New York, 1962.

[15] C. Lutz. Complexity and Succinctness of Public Announcement Logic. LTCS Report 05-09, avialable at http://lat.inf.tu-dresden.de/research/reports.html

[16] J. A. Plaza. Logics of public communications. In M. L. Emrich, M. Pfeifer, M. Hadzikadic, and Z. M. Ras, editors, *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems*, pages 201–216, 1989.

[17] V. R. Pratt. A near optimal method for reasoning about action. *Journal of Computer and System Sciences*, 20:231–254, 1980.

APPENDIX

Lemma 1. For all models $\mathcal{M} = (S, \sim, V)$, PAL formulas φ, finite sequences of PAL formulas σ, and states $s \in S(\mathcal{M}|\sigma)$, we have $\mathcal{M}, s \models \varphi^\sigma$ iff $\mathcal{M}|\sigma, s \models \varphi$.

Proof. The proof is by induction on $|\varphi| + |\sigma|$. The base case is $|\varphi| + |\sigma| = 1$. Then $\varphi = p \in \mathsf{PL}$ and $\sigma = \varepsilon$. We have $\mathcal{M}|\sigma = \mathcal{M}$ and $\varphi^\sigma = \varphi$ and are done. For the induction step, let $\mathcal{M}|\sigma = (T, \approx, U)$ and make a case distinction on the form of φ:

- $\varphi = p$. Trivial by definition of p^σ.
- $\varphi = \neg\psi$ or $\varphi = (\psi \wedge \vartheta)$. Easy using the definition of $(\neg\psi)^\sigma$, the semantics, and the induction hypothesis.
- $\varphi = K_a\psi$. Let $\mathcal{M}, s \models (K_a\psi)^\sigma$. We have to show that $\mathcal{M}|\sigma, s \models K_a\psi$, i.e. that $\mathcal{M}|\sigma, t \models \psi$ for all $t \in T$ with $s \approx_a t$. Hence, let $t \in T$ with $s \approx_a t$. Then we have $\mathcal{M}, t \models (\nu/\sigma)^\nu$ for all $\nu \in \mathsf{pre}(\sigma)$: assume to the contrary that $\mathcal{M}, t \not\models (\nu/\sigma)^\nu$ for some $\nu \in \mathsf{pre}(\sigma)$. By induction hypothesis, we get $\mathcal{M}|\nu, t \not\models (\nu/\sigma)$. Thus the state t is not present in $\mathcal{M}|\nu \cdot \nu/\sigma$ and consequently $t \notin T$, which is a contradiction. Thus $\mathcal{M}, t \models (\nu/\sigma)^\nu$ for all $\nu \in \mathsf{pre}(\sigma)$, implying $\mathcal{M}, t \models \bigwedge_{\nu \in \mathsf{pre}(\sigma)} (\nu/\sigma)^\nu$. Since $s \approx_a t$, we additionally have $s \sim_a t$. Together with $\mathcal{M}, s \models (K_a\psi)^\sigma$, we get $\mathcal{M}, t \models \psi^\sigma$. By induction hypothesis, we obtain $\mathcal{M}|\sigma, t \models \psi$ as required.
  Now let $\mathcal{M}|\sigma, s \models K_a\psi$. We have to show that $\mathcal{M}, s \models (K_a\psi)^\sigma$, i.e. that $\mathcal{M}, t \models \psi^\sigma$ for all $t \in S$ with (i) $s \sim_a t$ and (ii) $\mathcal{M}, t \models \bigwedge_{\nu \in \mathsf{pre}(\sigma)} (\nu/\sigma)^\nu$. Hence, let $t \in S$ such that (i) and (ii) are satisfied. We show by induction on the length of ν that t is a state in $\mathcal{M}|\nu$ for all $\nu \in \mathsf{pre}(\sigma) \cup \{\sigma\}$.
  - $\nu = \varepsilon$. Trivial since $\mathcal{M}|\varepsilon = \mathcal{M}$.
  - $\nu = \nu' \cdot \vartheta$. By induction hypothesis, t is a state in $\mathcal{M}|\nu'$. By (ii), we have $\mathcal{M}, t \models (\nu'/\sigma)^{\nu'}$. By (outer) induction hypothesis, this yields $\mathcal{M}|\nu', t \models (\nu'/\sigma)$. Thus, t is a state in $(\mathcal{M}|\nu')|(\nu'/\sigma) = \mathcal{M}|\nu$.

  Thus, $s, t \in T$. Hence, (i) yields $s \approx_a t$. Together with $\mathcal{M}|\sigma, s \models K_a\psi$, we get $\mathcal{M}|\sigma, t \models \psi$. By induction hypothesis, we get $\mathcal{M}, t \models \psi^\sigma$ as required.
- $\varphi = [\psi]\vartheta$. Then $\mathcal{M}, s \models ([\psi]\vartheta)^\sigma$ iff $\mathcal{M}, s \not\models \psi^\sigma$ or $\mathcal{M}, s \models \vartheta^{\sigma \cdot \psi}$ iff $\mathcal{M}|\sigma, s \not\models \psi$ or $\mathcal{M}|\sigma \cdot \psi, s \models \vartheta$ iff $\mathcal{M}|\sigma, s \not\models \psi$ or $(\mathcal{M}|\sigma)|\psi, s \models \vartheta$ iff $\mathcal{M}|\sigma, s \models [\psi]\vartheta$.
  The first "iff" holds by definition of $([\psi]\vartheta)^\sigma$, the second by induction hypothesis, the third since $\mathcal{M}|\sigma \cdot \psi = (\mathcal{M}|\sigma)|\psi$, and the fourth by the semantics.

□

Lemma 3. For all $s \in S'$ and $\varphi \in \mathsf{sub}(\chi)$ such that

$$\{sw \mid w \in P_\varphi\} \subseteq P_\chi,$$

we have $\mathcal{M}, s \models \varphi$ iff $\mathcal{M}', s \models \varphi$.

Proof. The proof is by induction on the structure of φ. As the induction start and the Boolean cases are trivial, we only treat the case $\varphi = K_\sigma\psi$.

"$\Leftarrow$". Let $\mathcal{M}', s \models \varphi$. We have to show that, for all $t \in S$, $s \sim_\sigma t$ implies $\mathcal{M}, t \models \varphi$. Hence, let $t \in S$ with $s \sim_\sigma t$. Then $t = s\sigma$. We first show that t does no have prefix $\widehat{w}$. Assume that the contrary holds. Since $s \in S'$ and thus s does not have prefix $\widehat{w}$, we have $t = \widehat{w}$. Since $\sigma \in P_\varphi$

and $\{sw \mid w \in P_\varphi\} \subseteq P_\chi$, the latter yields $\widehat{w} \in P_\chi$, in contradiction to $\widehat{w} \notin P_\chi$. Thus, t does not have prefix $\widehat{w}$, i.e., $t \in S'$. We have $P_\psi = \{w \mid \sigma w \in P_\varphi\}$. Thus, $t = s\sigma$ and $\{sw \mid w \in P_\varphi\} \subseteq P_\chi$ yield $\{tw \mid w \in P_\psi\} \subseteq P_\chi$. Finally, IH, $\mathcal{M}', s \models \varphi$, and $s \sim'_\sigma t$ yield $\mathcal{M}, t \models \psi$ as required.

"$\Rightarrow$". Let $\mathcal{M}, s \models \varphi$. We have to show that, for all $t \in S'$, $s \sim'_\sigma t$ implies $\mathcal{M}', t \models \varphi$. Hence, let $t \in S'$ with $s \sim'_\sigma t$. Then $t = s\sigma$. We have $P_\psi = \{w \mid \sigma w \in P_\varphi\}$. Thus, $t = s\sigma$ and $\{sw \mid w \in P_\varphi\} \subseteq P_\chi$ yield $\{tw \mid w \in P_\psi\} \subseteq P_\chi$. Finally, IH, $\mathcal{M}, s \models \varphi$, and $s \sim_\sigma t$ yield $\mathcal{M}', t \models \psi$ as required. □

Lemma 7. The single-agent PAL formula φ_0 is satisfiable iff the single-agent EL-formula φ_0^* is satisfiable.

Proof. "if". Let $\mathcal{M} = (S, \sim, V)$ be a model of φ_0^*, and let $s_0 \in S$ with $\mathcal{M}, s_0 \models \varphi_0^*$. By standard results on the unimodal logic S5, we may w.l.o.g. assume that $\sim = S \times S$ [5]. Thus, the second conjunct of φ_0^* implies that $K, s \models B_\varphi^\sigma$ for all $s \in S$ and all $(\varphi, \sigma) \in R(\varphi_0)$.

We show that, for all $s \in S$ and all $(\sigma, \varphi) \in R(\varphi_0)$, we have

$$\mathcal{M}, s \models \varphi^\sigma \text{ iff } \mathcal{M}, s \models p_\varphi^\sigma.$$

By Lemma 1, from this we get $\mathcal{M}|\sigma, s \models \varphi$ iff $\mathcal{M}, s \models p_\varphi^\sigma$. Since $\mathcal{M}, s_0 \models p_{\varphi_0}^\varepsilon$, this yields $\mathcal{M}, s_0 \models \varphi_0$ as required. The proof is by induction on $|\varphi| + |\sigma|$. For the induction start, we have $\varphi = q$ and $\sigma = \varepsilon$. Then $\varphi^\sigma = q$. Since $\mathcal{M}, s \models B_\varphi^\sigma = p_q^\sigma \leftrightarrow q$, we are done. For the induction step, we make a case distinction according to the structure of φ:

- $\varphi = q$. Identical to the induction start.
- $\varphi = \neg\psi$. Then $\mathcal{M}, s \models (\neg\psi)^\sigma$ iff $\mathcal{M}, s \models \neg\psi^\sigma$ iff $\mathcal{M}, s \not\models \psi^\sigma$ iff $\mathcal{M}, s \not\models p_\psi^\sigma$ iff $\mathcal{M}, s \models \neg p_\psi^\sigma$ iff $\mathcal{M}, s \models p_{\neg\psi}^\sigma$.
  The first "iff" holds by definition of $(\neg\psi)^\sigma$, the second by the semantics, the third by induction hypothesis, the fourth by the semantics, and the last since $\mathcal{M}, s \models B_{\neg\psi}^\sigma = p_{\neg\psi}^\sigma \leftrightarrow \neg p_\psi^\sigma$.
- $\varphi = \psi \wedge \vartheta$. Similar to the previous case.
- $\varphi = K\psi$. We have $\mathcal{M}, s \models (K\psi)^\sigma$ iff
  $$\mathcal{M}, s \models K(\bigwedge_{\nu \in \mathsf{pre}(\sigma)} (\nu/\sigma)^\nu \to \varphi^\sigma),$$
  which is the case iff, for all $t \in S$,
  $$s \sim t \text{ implies } \mathcal{M}, t \models (\bigwedge_{\nu \in \mathsf{pre}(\sigma)} (\nu/\sigma)^\nu \to \varphi^\sigma). \qquad (\dagger)$$
  By induction hypothesis, we get that, for all $t \in S$, (i) $\mathcal{M}, t \models \nu/\sigma^\nu$ iff $\mathcal{M}, t \models p_{\nu/\sigma}^\nu$ for all $\nu \in \mathsf{pre}(\sigma)$ and (ii) $\mathcal{M}, t \models \varphi^\sigma$ iff $\mathcal{M}, t \models p_\varphi^\sigma$. Thus, (†) holds iff, for all $t \in S$,
  $$s \sim t \text{ implies } \mathcal{M}, t \models (\bigwedge_{\nu \in \mathsf{pre}(\sigma)} p_{\nu/\sigma}^\nu \to p_\varphi^\sigma). \qquad (\ddagger)$$
  Since $\mathcal{M}, s \models p_{K\varphi}^\sigma \leftrightarrow K(\bigwedge_{\nu \in \mathsf{pref}(\sigma)} p_{\nu/\sigma}^\nu \to p_\varphi^\sigma)$, (‡) holds iff $\mathcal{M}, s \models p_{K\varphi}^\sigma$ and we are done.
- $\varphi = [\psi]\vartheta$. We have $\mathcal{M}, s \models ([\varphi]\psi)^\sigma$ iff $\mathcal{M}, s \models \varphi^\sigma \to \psi^{\sigma\cdot\varphi}$ iff $\mathcal{M}, s \models p_\varphi^\sigma \to p_\psi^{\sigma\cdot\varphi}$ iff $\mathcal{M}, s \models p_{[\varphi]\psi}^\sigma$.
  The first "iff" holds by definition of $([\varphi]\psi)^\sigma$, the second by induction hypothesis, and the third since $\mathcal{M}, s \models p_{[\varphi]\psi}^\sigma \leftrightarrow (p_\varphi^\sigma \to p_\psi^{\sigma\cdot\varphi})$.

"only if". Let $\mathcal{M} = (S, \sim, V)$ be a model of φ_0, and let $s_0 \in S$ with $\mathcal{M}, s_0 \models \varphi_0$. Define a model $\mathcal{M}'$ as $\mathcal{M}$ by additionally setting, for $(\sigma, \varphi) \in R(\varphi_0)$,

$$V(p_\varphi^\sigma) := \{s \in S \mid \mathcal{M}, s \models \varphi^\sigma\}.$$

Using the definition of the translation φ^σ and the implications B_φ^σ, it is straightforward to show by induction on $|\varphi| + |\sigma|$ that

$$\mathcal{M}', s \models B_\varphi^\sigma \text{ for all } (\sigma, \varphi) \in R(\varphi_0) \text{ and } s \in S.$$

Since $\mathcal{M}, s_0 \models \varphi_0$, we have $\mathcal{M}', s_0 \models p_{\varphi_0}^\varepsilon$. Thus, $\mathcal{M}', s_0 \models \varphi_0^*$. □

Lemma 11. For all models $\mathcal{M} = (S, \sim, V)$, PALC formulas φ, finite sequences of PALC formulas σ, and states $s \in S(\mathcal{M}|\sigma)$, we have $\mathcal{M}, s \models \varphi^\sigma$ iff $\mathcal{M}|\sigma, s \models \varphi$.

Proof. The proof is analogous to that of Lemma 1. We only treat the additional case for common knowledge:

- $\varphi = C_A\psi$. Let $\mathcal{M}, s \models (C_A\psi)^\sigma$. We have to show that $\mathcal{M}|\sigma, s \models C_A\psi$, i.e. that $\mathcal{M}|\sigma, t \models \psi$ for all $t \in T$ with $s\,(\bigcup_{a\in A} \approx_a)^*\, t$. Hence, let $t \in T$ with $s\,(\bigcup_{a\in A} \approx_a)^*\, t$. Then there are $s_1, \ldots, s_k \in T$ with $s_1 = s$, $s_k = t$, and $(s_i, s_{i+1}) \in \bigcup_{a\in A} \approx_a$ for $1 \le i < k$. We have $\mathcal{M}, s_i \models (\nu/\sigma)^\nu$ for all $\nu \in \mathsf{pre}(\sigma)$ and $1 < i \le k$: assume to the contrary of what is to be shown that $\mathcal{M}, s_i \not\models (\nu/\sigma)^\nu$ for some $\nu \in \mathsf{pre}(\sigma)$ and some i with $1 < i \le k$. By induction hypothesis, we get $\mathcal{M}|\nu, s_i \not\models (\nu/\sigma)$. Thus the state s_i is not present in $\mathcal{M}|\nu\cdot\nu/\sigma$ and consequently $s_i \notin T$, which is a contradiction. Thus $\mathcal{M}, s_i \models (\nu/\sigma)^\nu$ for all $\nu \in \mathsf{pre}(\sigma)$ and $1 < i \le k$. Moreover, we clearly have $(s_i, s_{i+1}) \in \bigcup_{a\in A} \sim_a$ for $1 \le i < k$. It follows that $(s, t) \in \sim_{\alpha^*}$, where
  $$\alpha := \bigcup_{a\in A} a; (\bigwedge_{\nu\in\mathsf{pre}(\sigma)} (\nu/\sigma)^\nu)?.$$
  Together with $\mathcal{M}, s \models (C_A\psi)^\sigma$, we get $\mathcal{M}, t \models \psi^\sigma$. By induction hypothesis, we obtain $\mathcal{M}|\sigma, t \models \psi$ as required.
  Now let $\mathcal{M}|\sigma, s \models C_A\psi$. We have to show that $\mathcal{M}, s \models (C_A\psi)^\sigma$, i.e. that $\mathcal{M}, t \models \psi^\sigma$ for all $t \in S$ with $(s, t) \in \sim_{\alpha^*}$. Hence, let $t \in S$ with $(s, t) \in \sim_{\alpha^*}$. Then there are $s_1, \ldots, s_k \in T$ with $s_1 = s$, $s_k = t$, and $(s_i, s_{i+1}) \in \sim_\alpha$ for $1 \le i < k$. We show by induction on the length of ν that s_i is a state in $\mathcal{M}|\nu$ for all $\nu \in \mathsf{pre}(\sigma) \cup \{\sigma\}$ and all i with $1 < i \le k$.
  - $\nu = \varepsilon$. Trivial since $\mathcal{M}|\varepsilon = \mathcal{M}$.
  - $\nu = \nu' \cdot \vartheta$. By induction hypothesis, s_i is a state in $\mathcal{M}|\nu'$. Since $(s_{i-1}, s_i) \in \sim_\alpha$, we have $\mathcal{M}, s_i \models (\nu'/\sigma)^{\nu'}$. By (outer) induction hypothesis, this yields $\mathcal{M}|\nu', s_i \models (\nu'/\sigma)$. Thus, s_i is a state in $(\mathcal{M}|\nu')|(\nu'/\sigma) = \mathcal{M}|\nu$.

  Thus, we have $s_1, \ldots, s_k \in T$. As $(s_i, s_{i+1}) \in \sim_\alpha$ implies $(s_0, s_{i+1}) \in \bigcup_{a\in A} \sim_a$, this yields
  $$(s, t) \in (\bigcup_{a\in A} \approx_a)^*.$$
  Together with $\mathcal{M}|\sigma, s \models C_A\psi$, we get $\mathcal{M}|\sigma, t \models \psi$. By induction hypothesis, we get $\mathcal{M}, t \models \psi^\sigma$ as required.

□

A complete and decidable security-specialised logic and its application to the TESLA protocol

Alessio Lomuscio and Bożena Woźna*
Department of Computer Science, UCL
Gower Street, London WC1E 6BT,UK
email: {B.Wozna,A.Lomuscio}@cs.ucl.ac.uk

ABSTRACT

We examine a logic to reason about security protocols by means of temporal and epistemic concepts. We report results on completeness and decidability of the formalism as well as its expressiveness. As a case study we apply the formalism in the analysis of TESLA, a secure stream multi-cast protocol.

Categories and Subject Descriptors

F.3.1 [**Specifying and Verifying and Reasoning about Programs**]: Specification techniques; D.2.4 [**Software/Program Verification**]: Model checking; I.2.4 [**Knowledge Representation Formalisms and Methods**]: Modal logic; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Verification, Theory, Security

Keywords

Epistemic logic, security protocols, TESLA

1. INTRODUCTION

There is a long tradition in the use of formal tools (process algebras, theorem proving, model checking, formal logics, etc) to analyse cryptographic protocols. In particular, since the late 80s there has been an attention to modelling security protocols in terms of the evolution of the states of knowledge of the principals. The work on BAN logic [6] was the seminal contribution in this line. BAN logic provides primitives to represent and reason about principals' beliefs, reception of messages, an array of cryptographic-specific concepts such as freshness, keys, etc, as well as axioms and rules tailored to the particular protocol under analysis governing the evolution of these.

It is nowadays customary ([18, 4], etc) to point out that BAN logic lacks several essential technical features (no representation of temporal flows, no well-defined semantics, no models for the intruder, etc) thereby making it impossible to use it consistently to prove that a particular protocol is correct or otherwise. Advances and improvements over the original formulation [1, 12, 15] have not solved this crucial concern.

Given that the emphasis in BAN logic was analysing whether particular protocols established *authentication* of the principals participating in it, one crucial assumption was the benevolence of principals. On the contrary, more recently attention has shifted to analysing whether or not a particular protocol can be exploited by an *adversary*, or *intruder*, achieving a property originally not intended by the protocol designer. In this line perfect encryption is normally assumed, and the intruder follows the Dolev-Yao model [9], i.e. it can read and compose messages but cannot break encrypted messages. This analysis is often carried out in combination with verification techniques, notably model checking, and epistemic notions are not analysed.

Recently, attention has been given again to modelling epistemic states of the principals in a protocol. In [14] a notion of algorithmic knowledge has been proposed. The semantics of the logic assumes that adversaries use algorithms to compute their knowledge. A related paper [20] explores an epistemic logic where the notion of explicit knowledge is linked to deductions in a logical framework. Similarly, in [2] a logic for awareness (originally introduced in [10]) is augmented with security-specialised predicates.

The referenced papers above do not treat temporal concepts explicitly. In [8] a linear time temporal epistemic logic is proposed, where properties of a protocol may be shown via resolution, although an explicit model of the adversary is not provided. In [21] it is shown that the verification problem for a linear temporal epistemic logic augmented with the past operators and security-specific predicates, and *tagged protocols* is decidable.

Many of the papers discussed above [1, 6, 12, 15], do not provide completeness results for the language and/or do not provide a grounded semantics [22] for the modalities. The main aim of the present paper is to propose a decidable security-specialised logic to reason about security concepts, that:

*The authors acknowledge support from the EPSRC (grant GR/S49353), the Nuffield Foundation (grant NAL/690/G), and The Royal Society (grant 2004/R3 EU).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

- has an intuitive, computationally-grounded semantics,
- offers a model of the adversary,
- enables the representation of epistemic and temporal concepts,
- enjoys soundness and completeness.

The logic presented in this paper is a combination of CTL [7], and a standard epistemic logic [11] augmented to include an additional epistemic operator representing explicit knowledge. It is further shown that the expressivity of the resulting system enables us to represent a non-standard notion of deductive knowledge. In addition we show how to apply the logic to model properties of the TESLA protocol [19], a protocol so far analysed only via process algebras [5] and Lynch-Vaandrager automata [3].

The rest of the paper is organised as follows. In Section 2 we present our security-specialised logic. Section 3 describes the standard Dolev-Yao adversary model in the logic of Section 2. In Section 4 we describe a scheme of the TESLA protocol, and analyse it formally in Section 5. We conclude in Section 6 with some final remarks.

2. A TDL LOGIC

In this section we present the syntax and semantics of a multi-modal temporal epistemic logic with security-specialised primitives, and we show that this logic is finitely axiomatisable. We call the logic *a temporal deductive logic* (TDL).

2.1 Syntax

We assume familiarity with the basic concepts of security protocols and we take $\mathbb{K} = \{k_1, k_2, \ldots\}$ to be a set of symmetric and asymmetric keys, $\mathbb{N}$ to be a set of nonces, $\mathbb{T} = \{t_1, t_2, \ldots\}$ to be a set of plain-texts, and $\mathbb{F} = \{f(k) \mid k \in \mathbb{K}$ and $f : \mathbb{K} \to \{0, 1, \ldots\}$ is a pseudo-random function$\}$ to be a set of commitments to keys. The commitment to a key k is an integer value that is computed by applying a pseudo-random function f to key k. It is assumed that f is effectively impossible to invert, so key k cannot be computed from the commitment to k. We assume that keys, nonces, texts, and commitments can be distinguished, that is, $\mathbb{K}$, $\mathbb{N}$, $\mathbb{T}$ and $\mathbb{F}$ are disjoint sets. A set of *messages* $\mathbb{M}$ is defined inductively by the following grammar:

$$m := t \mid k \mid n \mid f(k) \mid m \cdot m \mid \{m\}_k \mid \mathtt{MAC}(k, m)$$

where $t \in \mathbb{T}$, $k \in \mathbb{K}$, $n \in \mathbb{N}$, $f(k) \in \mathbb{F}$, m is a generic message, and $\mathtt{MAC} : \mathbb{K} \times \mathbb{M} \to \{0, 1, \ldots\}$ is a pseudo-random function, called *message authentication code*. It is assumed that MAC is effectively impossible to invert, so key k cannot be computed from the MAC value.

We write $m \cdot m'$ for the concatenation of m and m', $\{m\}_k$ for encryption of m with the key k, and $\mathtt{MAC}(k, m)$ for the message authentication code of m and k. We assume that the set $\mathbb{K}$ is closed under inverses, i.e. for a given key $k \in \mathbb{K}$ there is an inverse key $k^{-1} \in \mathbb{K}$ such that $\{\{m\}_k\}_{k^{-1}} = m$. If the cryptosystem uses symmetric keys, then $k = k^{-1}$; for the public cryptosystem k and k^{-1} are different. We also define a *submessage* binary relation $\sqsubseteq$ on $\mathbb{M}$ as the smallest reflexive and transitive relation satisfying the following conditions: (1) $m \sqsubseteq m \cdot m'$, (2) $m \sqsubseteq m' \cdot m$, (3) $m \sqsubseteq \{m\}_k$.

Let $\mathcal{PV}$ be a set of propositional variables. We assume that $\mathcal{AG} = \{1, \ldots, N\}$ is a finite set of agents. The set $\mathcal{WF}(TDL)$ of well-formed TDL formulas is defined by the following grammar:

$$\varphi := p \mid has_i(m) \mid sent_i(m) \mid received_i(m) \mid faked_i(m) \mid dropped_i(m) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathrm{E}\bigcirc\varphi \mid \mathrm{E}(\varphi \mathcal{U} \varphi) \mid \mathrm{A}(\varphi \mathcal{U} \varphi) \mid \mathcal{K}_i\varphi \mid \mathcal{X}_i\varphi \mid \mathcal{A}_i\varphi$$

where $p \in \mathcal{PV}$, $i \in \mathcal{AG}$, $m \in \mathbb{M}$. The terms $has_i(m)$, $sent_i(m)$, $received_i(m)$, $dropped_i(m)$, and $faked_i(m)$ are security-specialised propositional variables, which, as one would expect, are read as "*agent i has message m*", "*agent i sent message m*", "*agent i received message m*", "*agent i dropped message m*, and "*agent i faked message m*", respectively. $has_i(m)$ means that agent i has in his explicit possession the message m. What "explicit possession" means depends on the application, the capabilities of the principals, and the protocol they are running. In Section 3 we provide a concrete implementation of the $has_i(m)$ that captures Dolev-Yao capabilities.

We would like to emphasise that the above security-specialised propositional variables are *propositions*, not predicates. This is because we assume that the message m, which is carried by $has_i(m)$, $sent_i(m)$, $received_i(m)$, $dropped_i(m)$, and $faked_i(m)$, is not a value, but rather an intrinsic part of the propositions.

The above syntax extends CTL [7] with a standard epistemic modality $\mathcal{K}_i$ as well as operators for explicit knowledge ($\mathcal{X}_i$) and awareness ($\mathcal{A}_i$) as in [11]. The formula $\mathcal{X}_i\alpha$ is read as "*agent i knows explicitly that α*", $\mathcal{A}_i\alpha$ is read as "*agent i is aware of α*", and $\mathcal{K}_i\alpha$ is read as "*agent i knows (implicitly) that α*". The formula $\mathrm{E}\bigcirc\alpha$ is read as "*there exists a computation path such that at the next step of the path α holds*", $\mathrm{E}(\alpha\mathcal{U}\beta)$ is read as "*there exists a computation path such that β eventually occurs and α continuously holds until then*", and $\mathrm{A}(\alpha\mathcal{U}\beta)$ is read as "*for all computation paths β eventually occurs and α continuously holds until then*". We shall further use the shortcut $\mathcal{D}_i\alpha$ to represent $\mathrm{E}(\mathcal{K}_i\alpha\mathcal{U}\mathcal{X}_i\alpha)$. The formula $\mathcal{D}_i\varphi$ is read as "*agent i may deduce α (by some computational process)*". Note that TDL differs from [16] only in a security-specialised subset of the set $\mathcal{PV}$.

2.2 Interpreted Systems

In this section we will briefly review the multi-agent framework from [11]; especially we will focus on *interpreted systems*, over which a semantics for TDL will be given.

A multi-agent system (MAS) consists of n agents and an environment, each of which is in some particular local state at a given point in time. We assume that an agent's local state encapsulates all the information the agent has access to. In the security settings, the local state of an agent might include some initial information about keys, the messages that the agent has sent and received so far, and so on. The local states of the environment describe information that is relevant to the system but that is not included in any local agent's state. Note also that the environment can be viewed as just another agent, as we will do here.

A multi-agent system is not a static entity. Its computations are usually defined by means of runs (see [11]), where a *run* is a function from the natural numbers to the set of all the possible global states. Thus, in these settings, an *interpreted system* for a multi-agent system is defined as a set of runs together with a valuation function for the propositional variables of the language under consideration. More formally, let each agent $i \in \mathcal{AG}$ be associated with a set of local states L_i, and the environment be associated with a set of local states L_e. Then, an *interpreted system* is a tuple $IS = (S, T, \sim_1, \ldots, \sim_n, \mathcal{V})$, where $S \subseteq \prod_{i=1}^{n} L_i \times L_e$ is a set

of global states; $T \subseteq S \times S$ is a serial (temporal) relation on S; for each agent $i \in \mathcal{AG}$, $\sim_i \subseteq S \times S$ is an equivalence (epistemic) relation defined by: $s \sim_i s'$ iff $l_i(s') = l_i(s)$, where $l_i : S \to L_i$ is a function that returns the local state of agent i from a global state; $\mathcal{V} : S \longrightarrow 2^{\mathcal{PV}}$ is a valuation function. $\mathcal{V}$ assigns to each state a set of proposition variables that are assumed to be true at that state. For more details we refer to [11].

In order to give a semantics to TDL we extend the above definition by means of local awareness functions, used to indicate the facts that agents are aware of. As in [11], we are not very specific with the notion of awareness, but by being aware we intutively mean "*to be able to figure out the truth*", or "*to be able to compute the truth within time T*", etc.

DEFINITION 1 (MODEL). *Given $\mathcal{AG} = \{1, \ldots, n\}$, a* model *is a tuple $M = (S, T, \sim_1, \ldots, \sim_n, \mathcal{V}, \mathbb{A}_1, \ldots, \mathbb{A}_n)$, where S, T, $\sim_i$, and $\mathcal{V}$ are defined as above, and $\mathbb{A}_i : L_i \longrightarrow 2^{\mathcal{WF}(TDL)}$ is an awareness function assigning a set of formulas to each state, for each $i \in \mathcal{AG}$.*

Note that the set of formulas that the agent is aware of can be arbitrary and may not be closed under sub-formulas. Note also that the definition of the model is an extension of the awareness structure (see [11]) by a temporal relation. Moreover, it restricts the standard awareness function to be defined over local states only.

In the paper we focus on a specific class of multi-agent systems, appropriate to modelling security protocols. These are message-passing systems in which one or more of the agents is an adversary controling the communication channel.

We assume that the local state of an agent is a sequence of events of the form $(e_0, \ldots, e_m)$, where e_0 is the initial event, and for $i \in \{1, \ldots, m\}$, e_i is a term of the form $sent(i, m)$ or $recv(m)$, where m is a message and i is an agent. The term $sent(i, m)$ stands for the agent has sent message m to agent i. Similarly the term $recv(m)$ represents that the agent has received message m. Note that in $recv(m)$ the sender is not specified. This is because the receiver will not in general be able to determine the sender of a message he has received.

2.3 Satisfaction

A *path* in M is an infinite sequence $\pi = (s_0, s_1, \ldots)$ of global states such that $(s_i, s_{i+1}) \in T$ for each $i \in \mathbb{N}$. For a path $\pi = (s_0, s_1, \ldots)$, we take $\pi(k) = s_k$. By $\Pi(s)$ we denote the set of all the paths starting at $s \in S$.

DEFINITION 2 (SATISFACTION). *Let M be a model, s a state, and α, β TDL formulas. The satisfaction relation $\models$, indicating truth of a formula in model M at state s, is defined inductively as follows:*

$(M, s) \models p$ *iff* $p \in \mathcal{V}(s)$,
$(M, s) \models \alpha \vee \beta$ *iff* $(M, s) \models \alpha$ *or* $(M, s) \models \beta$,
$(M, s) \models \neg\alpha$ *iff* $(M, s) \not\models \alpha$,
$(M, s) \models \mathrm{E}\bigcirc\alpha$ *iff* $(\exists \pi \in \Pi(s))(M, \pi(1)) \models \alpha$,
$(M, s) \models \mathrm{E}(\alpha \mathcal{U} \beta)$ *iff* $(\exists \pi \in \Pi(s))(\exists m \geq 0)[(M, \pi(m)) \models \beta$ *and* $(\forall j < m)(M, \pi(j)) \models \alpha]$,
$(M, s) \models \mathrm{A}(\alpha \mathcal{U} \beta)$ *iff* $(\forall \pi \in \Pi(s))(\exists m \geq 0)[(M, \pi(m)) \models \beta$ *and* $(\forall j < m)(M, \pi(j)) \models \alpha]$,
$(M, s) \models \mathcal{A}_i\alpha$ *iff* $\alpha \in \mathbb{A}_i(l_i(s))$,
$(M, s) \models \mathcal{K}_i\alpha$ *iff* $(\forall s' \in S)$ $(s \sim_i s'$ *implies* $(M, s') \models \alpha)$,
$(M, s) \models \mathcal{X}_i\alpha$ *iff* $(M, s) \models \mathcal{K}_i\alpha$ *and* $(M, s) \models \mathcal{A}_i(\alpha)$.

Henceforth, we will only consider models with a fixed interpretation for the security-specialised propositional variables $sent_i(m)$ and $received_i(m)$; in particular, we take $\models$ to be defined for these propositions as follows:

$(M, s) \models sent_i(m)$ iff $(\exists m' \in \mathbb{M})(\exists j \in \mathcal{AG})$ such that $m \sqsubseteq m'$ and $sent(j, m') \in l_i(s)$,
$(M, s) \models received_i(m)$ iff $recv(m) \in l_i(s)$.

We leave definitions of the other security-specialised propositions open. Namely, for distinct protocols these propositions will be defined differently.

Note that since $\mathcal{D}_i\alpha$ is a shortcut for $\mathrm{E}(\mathcal{K}_i\alpha \mathcal{U} \mathcal{X}_i\alpha)$, as defined on page 2, we have that $(M, s) \models \mathcal{D}_i\alpha$ iff $(M, s) \models \mathrm{E}(\mathcal{K}_i\alpha \mathcal{U} \mathcal{X}_i\alpha)$. We would like to emphasise that satisfaction for $\mathcal{X}_i$ can be defined using only the $\sim_i$ relation and the $\mathbb{A}_i$ function, but we will find it convenient in the axiomatisation to have a dedicated operator $\mathcal{A}_i$ for awareness of agent i. This is in line with [11]. Note also that the definition of the satisfaction relation is quite general; in particular, it contains nothing that is specific to security protocols, except the interpretation of the security-specialised propositional variables.

The remaining operators can be introduced as abbreviations in the usual way, i.e., $\alpha \wedge \beta \stackrel{def}{=} \neg(\neg\alpha \vee \neg\beta)$, $\alpha \Rightarrow \beta \stackrel{def}{=} \neg\alpha \vee \beta$, $\alpha \Leftrightarrow \beta \stackrel{def}{=} (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$, $\mathrm{A}\bigcirc\alpha \stackrel{def}{=} \neg\mathrm{E}\bigcirc\neg\alpha$, $\mathrm{E}\Diamond\alpha \stackrel{def}{=} \mathrm{E}(\top \mathcal{U} \alpha)$, $\mathrm{A}\Diamond\alpha \stackrel{def}{=} \mathrm{A}(\top \mathcal{U} \alpha)$, $\mathrm{E}\Box\alpha \stackrel{def}{=} \neg\mathrm{A}\Diamond\neg\alpha$, $\mathrm{A}\Box\alpha \stackrel{def}{=} \neg\mathrm{E}\Diamond\neg\alpha$, $\mathrm{A}(\alpha \mathcal{W} \beta) \stackrel{def}{=} \neg\mathrm{E}(\neg\alpha \mathcal{U} \neg\beta)$, $\mathrm{E}(\alpha \mathcal{W} \beta) \stackrel{def}{=} \neg\mathrm{A}(\neg\alpha \mathcal{U} \neg\beta)$.

Let M be a model. We say that a TDL formula φ is *valid in M* (written $M \models \varphi$), if $M, s \models \varphi$ for all states $s \in S$, and that a TDL formula φ is *valid* (written $\models \varphi$), if φ is valid in all the models M .

2.4 A Complete Axiomatic System for TDL

An *axiomatic system* consists of a collection of *axioms* and *inference rules*. An axiom is a formula, and an inference rule has the form "from previously derived formulas $\varphi_1, \ldots, \varphi_m$ infer formula φ". We say that φ is *provable* (written $\vdash \varphi$) if there is a sequence of formulas ending with φ, such that each formula is either an instance of an axiom, or follows from other provable formulas by applying an inference rule. An axiom system is said to be *sound* with respect to the models defined as in Definition 1, if $\vdash \varphi$ implies $\models \varphi$. An axiom system is said to be *complete* with respect to the models defined as in Definition 1, if $\models \varphi$ implies $\vdash \varphi$.

Let $i \in \{1, \ldots, n\}$. Consider an axiom system for TDL as defined below:

PC. All substitution instances of classical tautologies.
T1. $\mathrm{E}\bigcirc\top$ T2. $\mathrm{E}\bigcirc(\alpha \vee \beta) \Leftrightarrow \mathrm{E}\bigcirc\alpha \vee \mathrm{E}\bigcirc\beta$
T3. $\mathrm{E}(\alpha \mathcal{U} \beta) \Leftrightarrow \beta \vee (\alpha \wedge \mathrm{E}\bigcirc\mathrm{E}(\alpha \mathcal{U} \beta))$
T4. $\mathrm{A}(\alpha \mathcal{U} \beta) \Leftrightarrow \beta \vee (\alpha \wedge \mathrm{A}\bigcirc\mathrm{A}(\alpha \mathcal{U} \beta))$
K1. $(\mathcal{K}_i\alpha \wedge \mathcal{K}_i(\alpha \Rightarrow \beta)) \Rightarrow \mathcal{K}_i\beta$ K2. $\mathcal{K}_i\alpha \Rightarrow \alpha$
K3. $\neg\mathcal{K}_i\alpha \Rightarrow \mathcal{K}_i\neg\mathcal{K}_i\alpha$ X1. $\mathcal{X}_i\alpha \Leftrightarrow \mathcal{K}_i\alpha \wedge \mathcal{A}_i\alpha$
A1. $\mathcal{A}_i\alpha \Rightarrow \mathcal{K}_i\mathcal{A}_i\alpha$ A2. $\neg\mathcal{A}_i\alpha \Rightarrow \mathcal{K}_i\neg\mathcal{A}_i\alpha$
R1. From α and $\alpha \Rightarrow \beta$ infer β
R2. From α infer $\mathcal{K}_i\alpha$, $i = 1, \ldots, n$
R3. From $\alpha \Rightarrow \beta$ infer $\mathrm{E}\bigcirc\alpha \Rightarrow \mathrm{E}\bigcirc\beta$
R4. From $\gamma \Rightarrow (\neg\beta \wedge \mathrm{E}\bigcirc\gamma)$ infer $\gamma \Rightarrow \neg\mathrm{A}(\alpha \mathcal{U} \beta)$
R5. From $\gamma \Rightarrow (\neg\beta \wedge \mathrm{A}\bigcirc(\gamma \vee \neg\mathrm{E}(\alpha \mathcal{U} \beta)))$ infer $\gamma \Rightarrow \neg\mathrm{E}(\alpha \mathcal{U} \beta)$

THEOREM 1 ([16]). *The system TDL is sound and complete, i.e. $\models \varphi$ iff $\vdash \varphi$, for any formula $\varphi \in \mathcal{WF}(TDL)$.*

3. THE DOLEV-YAO ADVERSARY

The Dolev-Yao adversary [9] can compose messages, replay them, or decipher them if he has the right keys. In

the paper we use a formalisation of the Dolev-Yao adversary given in [14]. Namely, we define a derivation relation $H \vdash_{DY} m$ between a set of messages H and a message m. Intuitively, $\vdash_{DY}$ means that an adversary can "extract" message m from a set H of received messages and keys, using the admissible operations.

The derivation relation $\vdash_{DY}$ is defined as follows:

- if $m \in H$, then $H \vdash_{DY} m$
- if $H \vdash_{DY} \{m\}_k$ and $H \vdash_{DY} k^{-1}$, then $H \vdash_{DY} m$
- if $H \vdash_{DY} m \cdot m'$, then $H \vdash_{DY} m$
- if $H \vdash_{DY} m \cdot m'$, then $H \vdash_{DY} m'$

In our framework, to capture the capabilities of the Dolev-Yao adversary, we specify how the adversary can extract a message, by defining an awareness function $\mathbb{A}_i^{DY}$ for the intruder i, and by introducing the following fixed interpretation for the proposition $has_i(m)$:

$M, s \models has_i(m)$ iff there exists m' such that $m \sqsubseteq m'$ and $recv(m') \in l_i(s)$.

The function $\mathbb{A}_i^{DY} : L_i \rightarrow 2^{\mathcal{WF}(TDL)}$ is defined by: $has_i(m) \in \mathbb{A}_i^{DY}(l)$ iff one of the following conditions holds:

- $recv(m) \in l$.
- $m \in e_0(l)$, where the function e_0 returns the set of messages contained in the initial event of the local state l.
- there exists m' such that $has_i(m \cdot m') \in \mathbb{A}_i^{DY}(l)$ or $has_i(m' \cdot m) \in \mathbb{A}_i^{DY}(l)$.
- there exists an asymmetric key k such that $has_i(\{m\}_k) \in \mathbb{A}_i^{DY}(l)$ and $has_i(k^{-1}) \in \mathbb{A}_i^{DY}(l)$.
- there exists a symmetric key k such that $has_i(\{m\}_k) \in \mathbb{A}_i^{DY}(l)$ and $has_i(k) \in \mathbb{A}_i^{DY}(l)$.
- $m = m_1 \cdot m_2$ and $has_i(m_1) \in \mathbb{A}_i^{DY}(l)$ and $has_i(m_2) \in \mathbb{A}_i^{DY}(l)$.
- $m = \{m_1\}_k$ and $has_i(m_1) \in \mathbb{A}_i^{DY}(l)$ and $has_i(k) \in \mathbb{A}_i^{DY}(l)$.

We can now show that using $\mathbb{A}_i^{DY}$ an adversary can check that he explicitly knows $has_i(m)$ in a state s iff m can be derived from the messages received in the states s together with the messages initially known via the relation $\vdash_{DY}$.

LEMMA 1. *Let $M = (S, T, \sim_1, \ldots, \sim_n, \mathcal{V}, \mathbb{A}_1, \ldots, \mathbb{A}_n)$ be a model such that $\mathbb{A}_i = \mathbb{A}_i^{DY}$. Then, $M, s \models \mathcal{X}_i(has_i(m))$ iff $\{m' \mid recv(m') \in l_i(s)\} \cup e_0(l_i(s)) \vdash_{DY} m$.*

Proof. (Left to right). Assume $M, s \models \mathcal{X}_i(has_i(m))$ hold. By the definition of $\models$, we have that $M, s \models \mathcal{K}_i(has_i(m))$ and $has_i(m) \in \mathbb{A}_i(l_i(s))$. Since $has_i(m) \in \mathbb{A}_i(l_i(s))$, by the definition of $\mathbb{A}_i^{DY}$ we have that either $m \in \{m' \mid recv(m') \in l_i(s)\} \cup e_0(l_i(s))$, or there exists m'' such that $m \sqsubseteq m''$ and $m'' \in \{m' \mid recv(m') \in l_i(s)\} \cup e_0(l_i(s))$. This implies that $\{m' \mid recv(m') \in l_i(s)\} \cup e_0(l_i(s)) \vdash_{DY} m$.
(Right to left). Assume $\{m' \mid recv(m') \in l_i(s)\} \cup e_0(l_i(s)) \vdash_{DY} m$ hold. Then, we have that either $recv(m) \in l_i(s)$ or there exists m' such that $m \sqsubseteq m'$ and $recv(m') \in l_i(s)$. So, by the definition of $\mathbb{A}_i^{DY}$ we have that $has_i(m) \in \mathbb{A}_i^{DY}(l_i(s))$. Moreover, by the definition of $\models$ we have that $M, s \models has_i(m)$. Since $\mathbb{A}_i^{DY}$ is defined over the local states of agent i, we have that for all $t \in S$ such that $s \sim_i t$, $M, t \models has_i(m)$ holds. Hence, by the definition of $\models$ we have that $M, s \models \mathcal{K}_i(has_i(m))$. Hence, we conclude that $M, s \models \mathcal{X}_i(has_i(m))$. □

A model of the Dolev-Yao adversary was already considered in the interpreted system framework in [14, 20]. However, in those papers, the notion of *algorithmic knowledge* was used to capture the capabilities of the Dolev-Yao adversary instead.

4. TESLA

The *timed efficient stream loss-tolerant authentication* (TESLA) protocol was developed by A. Perrig et al. [19]. It was designed to enable authentication of a continuous stream of packets over an unreliable channel. The developers of TESLA proposed five variants of the protocol; only the first of these is discussed here.

Every variant of the TESLA protocol assumes a single sender broadcasting a continuous stream of packets. Since receivers act independently of one another, only one receiver will be considered below. TESLA uses cryptographic primitives like *commitments* to keys and `MAC` values. Recall that the commitment to a key k is computed by applying a pseudo-random function f to k, and the `MAC` value is computed by applying a *message authentication code* function `MAC` to a part of the packet content and a key. It is assumed that both the sender and the receiver know the pseudo-random function f, and the message authentication code function `MAC`. With the exception of the two initial packets, each TESLA packet (in the variant we analyse here) contains: (1) the message to be delivered; (2) a *commitment* to the key to be used to encode the `MAC` of the next packets; (3) the key that was used to encode the `MAC` of the previous sent packet; (4) the `MAC` of the current packet.

We now introduce a TESLA variant assuming that the protocol uses one pseudo-random function only. We have introduced such a simplification, because in our opinion the extra pseudo-random function does not provide any additional security; the same assumption is made in [3, 5].

Let $\mathbf{S}$ stand for the sender, $\mathbf{R}$ stand for the receiver, and $[x, y]$ denote the concatenation of x and y. Moreover, assume that $\mathbf{S}$ has a digital signature key pair, with private key $k_{\mathbf{S}}^{-1}$ and public key $k_{\mathbf{S}}$ known to $\mathbf{R}$. Further, assume that $\mathbf{R}$ chooses a random and unpredictable nonce. Then, the initial n steps, for $n > 1$, of the protocol for one sender and one receiver are the following:

(-1) $\mathbf{R} \rightarrow \mathbf{S} : n_{\mathbf{R}}$
(0) $\mathbf{S} \rightarrow \mathbf{R} : \{f(k_1), n_{\mathbf{R}}\}_{k_{\mathbf{S}}^{-1}}$
(1) $\mathbf{S} \rightarrow \mathbf{R} : [P_1, \mathtt{MAC}(k_1, P_1)]$, for $P_1 = [t_1, f(k_2)]$
(2) $\mathbf{S} \rightarrow \mathbf{R} : [P_2, \mathtt{MAC}(k_2, P_2)]$, for $P_2 = [t_2, f(k_3), k_1]$
...
(n) $\mathbf{S} \rightarrow \mathbf{R} : [P_n, \mathtt{MAC}(k_n, P_n)]$, for $P_n = [t_n, f(k_{n+1}), k_{n-1}]$

In the first step (i.e. step (-1.)) $\mathbf{R}$ sends a nonce $n_{\mathbf{R}}$ to $\mathbf{S}$ in order to ensure freshness of the session. In step (0), $\mathbf{S}$ encodes with $k_{\mathbf{S}}^{-1}$ the pair composed by the nonce just received and the commitment to key k_1 (i.e. $f(k_1)$), produced by the agreed pseudo-random function f. This is sent to $\mathbf{R}$. $\mathbf{R}$ knows $k_{\mathbf{S}}$, so he can decode the message $\{f(k_1), n_{\mathbf{R}}\}_{k_{\mathbf{S}}^{-1}}$. Given this, $\mathbf{R}$ knows that commitment $f(k_1)$ is authentic; however, because he does not know the reverse of f, he cannot compute the value of the key k_1. In step (1), $\mathbf{S}$ transmits in clear the first packet P_1 together with its `MAC`, i.e. the value $\mathtt{MAC}(k_1, P_1)$. P_1 contains message t_1 and commitment to key k_2 only. k_2 is used to compute the `MAC` of the next packet. After this step $\mathbf{R}$ knows the content of t_1, but he cannot be sure of its authenticity. This is because he does not know the value of the key k_1, and so he cannot check the `MAC` value. The authentication of packet P_1 is done by the sender's disclosing of k_1 in step (2). In steps (2),...,(n) $\mathbf{S}$ sends packet P_i, for $i \in \{1, \ldots, n\}$, authenticated by the `MAC` value computed from key k_i and P_i (i.e. the value $\mathtt{MAC}(k_i, P_i)$). P_i contains message t_i, commitment

to key k_{i+1} that is computed by applying pseudo-random function f to k_{i+1} , and key k_{i-1} thereby enabling $\mathbf{R}$ to verify the commitment $f(k_{i-1})$ and the MAC of packet P_i.

The TESLA scheme is susceptible to packet loss. In particular, once a packet is dropped no further packets can be authenticated; in the next section we will formally show that TESLA satisfies this property. Moreover, the scheme can be subverted if an attacker gets packet P_{i+1} before the receiver gets P_i. This is because the attacker would then know the secret key k_i that is used to compute the MAC value of P_i, and use this to forge all subsequent traffic by impersonating $\mathbf{S}$. To prevent this attack, the TESLA participants are initially synchronised, they know the precise schedule of packets, and $\mathbf{S}$ has to send packets at regular intervals that were agreed with $\mathbf{R}$ during the synchronisation process. More details are in [19]. This is not essential for our analysis.

TESLA guarantees the following security property: "*the receiver does not accept as authentic any message unless it was actually sent by the sender*". A streaming scheme that provides this guarantee is usually called a *secure stream authentication scheme* [19]. The main aim of the rest of the paper is to analyse this statement formally.

5. MODELLING OF TESLA IN TDL

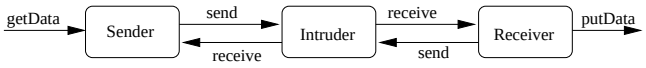


Figure 1: The Dolev-Yao assumption

So far, the TESLA protocol has been modelled and checked via process algebras [5] and Lynch-Vaandrager automata [3] only. Here, we are interested in applying the interpreted systems framework and the TDL language, which has a computationally-grounded semantics, to perform a *semantic* analysis of this protocol. So, let us begin by building a model $M = (S, T, \sim_{\mathbf{S}}, \sim_{\mathbf{R}}, \sim_{\mathbf{I}}, \mathcal{V}, \mathbb{A}_{\mathbf{S}}, \mathbb{A}_{\mathbf{R}}, \mathbb{A}_{\mathbf{I}})$ for TESLA. In order to perform this task, we have to provide definitions of the set of reachable global states S, the transition relation T, the valuation function $\mathcal{V}$, and the awareness functions $\mathbb{A}_{\mathbf{S}}$, $\mathbb{A}_{\mathbf{R}}$ and $\mathbb{A}_{\mathbf{I}}$.

There are three active components in the TESLA protocol: a sender, a receiver, and an intruder. In the formalism of interpreted systems, it is convenient to see the sender and the receiver as agents, and the intruder as the environment. We assume that the communication channel is under complete control of the intruder, hence all messages from the sender to the receiver and vice versa go through the intruder (see Figure 1). The intruder can overhear, capture, drop, resend, delay and fake messages. We do not allow the intruder to encrypt and decrypt messages unless he has the appropriate key, and we force the intruder to send well-formed packets only. Namely, a packet sent (re-sent, faked) by the intruder has to contain a message body, a key commitment, a key, and a MAC value computed from the packet and some key. Moreover, we assume that the sender, the receiver, and the intruder use a shared pseudo-random function f and a shared message authentication code function MAC. Namely, they can all compute $f(m)$ and $\mathtt{MAC}(m, n)$, if they have m and n. Further, we assume that the receiver and the intruder know the public key of the sender, and that the sender and the intruder start off with disjoint sets of keys.

Each of the principals as well as the intruder can be modelled by considering their local states. Before we define them, we first introduce the following notion. For $t_i \in \mathbb{M}$, $f(k_{i+1}) \in \mathbb{F}$, $k_i \in \mathbb{K}$, $n_{\mathbf{R}} \in \mathbb{N}$ we define a message P_i, called a *packet*, as follows:

$$P_i = \begin{cases} (t_i \cdot f(k_{i+1}) \cdot k_{i-1}) \cdot \mathtt{MAC}(k_i, (t_i \cdot f(k_{i+1}) \cdot k_{i-1})), \\ \text{if } i > 1; \\ (t_i \cdot f(k_{i+1})) \cdot \mathtt{MAC}(k_i, (t_i \cdot f(k_{i+1}))), \text{ if } i = 1; \\ \{f(k_{i+1}) \cdot n_{\mathbf{R}}\}_{k_{\mathbf{S}}^{-1}}, \quad \text{if } i = 0. \end{cases}$$

We assume that the receiver knows the precise sending schedule of packets, and that this information is incorporated into packet P_0. Notice, that from this assumption it follows that P_0 always has to arrive to the receiver safely for the protocol to function. Further, we assume the sender and the receiver to keep the three most recent packets, and the intruder to keep five.

For the sender $\mathbf{S}$ we take the following set of local states:

$$L_{\mathbf{S}} = \{[\cdot], [recv(n_{\mathbf{R}})], [sent(\mathbf{R}, P_0)]\} \cup \\ \{[sent(\mathbf{R}, P_{i-1}), sent(\mathbf{R}, P_i)] \mid i > 0\} \cup \\ \{[sent(\mathbf{R}, P_{i-1}), sent(\mathbf{R}, P_i), sent(\mathbf{R}, P_{i+1})] \mid i > 0\}$$

$[\cdot]$ represents $\mathbf{S}$'s initial state in the protocol. In any state, we assume that $\mathbf{S}$ has all the information he needs to prepare a packet, i.e. he has a complete set of messages $M_{\mathbf{S}} \subseteq \mathbb{M}$. Further, we assume that $M_{\mathbf{S}}$ constitutes $\mathbf{S}$'s initial database that remains accessible to him throughout the run. For convenience we do not add this explicitly to the local states. The local state $[recv(n_{\mathbf{R}})]$ represents the message from $\mathbf{R}$ to establish communication, whereas $[sent(\mathbf{R}, P_0)]$ represents the state in which $\mathbf{S}$ has just sent P_0 to $\mathbf{R}$. $[sent(\mathbf{R}, P_{i-1})$, $sent(\mathbf{R}, P_i)]$ and $[sent(\mathbf{R}, P_{i-1})$, $sent(\mathbf{R}, P_i)$, $sent(\mathbf{R}, P_{i+1})]$ represent states, in which $\mathbf{S}$ has sent packets P_j, where $j \leq i + 1$ and $i > 0$.

We take the following set of local states for the receiver $\mathbf{R}$:

$$L_{\mathbf{R}} = \{[\cdot], [sent(\mathbf{S}, n_{\mathbf{R}})], [stop], [recv(P_0)]\} \cup \\ \{[recv(P_0), recv(P_2)]\} \cup \{[recv(P_i), recv(P_{i+1})] \mid i \geq 0\} \cup \\ \{[recv(P_{i-1}), recv(P_i), recv(P_{i+1})] \mid i > 0\} \cup \\ \{[recv(P_{i-1}), recv(P_i), recv(P_{i+2})] \mid i > 0\} \cup \\ \{[recv(P_i), recv(P_{i+1}), recv(P'_{i+2})] \mid i \geq 0\} \cup \\ \{[recv(P_0), recv(P'_1)]\} \cup \{[recv(P_0), recv(P'_1), recv(P'_2)]\} \\ \cup \{[recv(P_0), recv(P'_1), recv(P_2)]\}$$

$[\cdot]$ represents $\mathbf{R}$'s initial state in the protocol, in which $\mathbf{R}$ has his set of nonces only. $[sent(\mathbf{S}, n_{\mathbf{R}})]$ represents the local state in which $\mathbf{R}$ has just sent the nonce $n_{\mathbf{R}}$ to $\mathbf{S}$, and he is waiting for packets. $[stop]$ represents the local state, in which $\mathbf{R}$ has just stopped collecting packets. The local states $[recv(P_0)]$, $[recv(P_0), recv(P_2)]$, $[recv(P_i)$, $recv(P_{i+1})]$ for $i \geq 0$, $[recv(P_{i-1}), recv(P_i)$, $recv(P_{i+2})]$ and $[recv(P_{i-1})$, $recv(P_i), recv(P_{i+1})]$ for $i > 0$ represent the packets $\mathbf{R}$ has received from $\mathbf{S}$, whereas $[recv(P_0)$, $recv(P'_1)]$, $[recv(P_0)$, $recv(P'_1), recv(P'_2)]$, $[recv(P_0), recv(P'_1)$, $recv(P_2)]$, and $[recv(P_i), recv(P_{i+1}), recv(P'_{i+2})]$ for $i \geq 0$ represent the faked packets $\mathbf{R}$ has received. Specifically, these states represent situations in which $\mathbf{I}$ has produced a new packet, say P'_i, and has sent it to $\mathbf{R}$ instead of forwarding the intercepted packet P_i.

It remains to model the intruder's set of local states. For $\mathbf{I}$ we take the following set:

$$L_{\mathbf{I}} = \{[\cdot], [recv(n_{\mathbf{R}})], [recv(P_0)]\} \cup \\ \{[recv(P_i), recv(P_{i+1})] \mid i \geq 0\} \cup \\ \{[recv(P_{i-1}), recv(P_i), recv(P_{i+1})] \mid i > 0\} \cup$$

$$\{[recv(P_0), recv(P_1), send(\mathbf{R}, P_1')]\} \cup$$
$$\{[recv(P_0), recv(P_1), send(\mathbf{R}, P_1'), recv(P_2)]\} \cup$$
$$\{[recv(P_0), recv(P_1), send(\mathbf{R}, P_1'), recv(P_2), send(\mathbf{R}, P_2')]\} \cup$$
$$\{[recv(P_{i-1}), recv(P_i), recv(P_{i+1}), send(\mathbf{R}, P_{i+1}')] \mid i > 0\}$$

$[\cdot]$ represents $\mathbf{I}$'s initial state in the protocol. In any state, we assume that $\mathbf{I}$ has all the information needed to prepare well-formed packets, with $M_{\mathbf{I}} \subseteq \mathbb{M}$ such that $M_{\mathbf{I}} \cap M_{\mathbf{S}} = \emptyset$. Further, we assume that $M_{\mathbf{I}}$ can grow during the run. $[recv(n_{\mathbf{R}})]$ represents $\mathbf{I}$'s state following the interception of $\mathbf{R}$'s initial message to $\mathbf{S}$. $[recv(P_0)]$, $[recv(P_i), recv(P_{i+1})]$ and $[recv(P_{i-1}), recv(P_i),\ recv(P_{i+1})]$ represent the packets intercepted by $\mathbf{I}$ sent by $\mathbf{S}$ to $\mathbf{R}$. $[recv(P_0), recv(P_1), send(\mathbf{R}, P_1')]$, $[recv(P_0), recv(P_1), send(\mathbf{R}, P_1'), recv(P_2)]$, $[recv(P_0), recv(P_1), send(\mathbf{R}, P_1'), recv(P_2), send(\mathbf{R}, P_2')]$, and $[recv(P_{i-1}), recv(P_i), recv(P_{i+1}), send(\mathbf{R}, P_{i+1}')]$ for $i > 0$, represent the packets intercepted by $\mathbf{I}$ sent by $\mathbf{S}$ to $\mathbf{R}$ and their faked versions.

Given the sets of local states for the TESLA agents, i.e. the sets $L_{\mathbf{S}}$, $L_{\mathbf{R}}$ and $L_{\mathbf{I}}$, the following sets of actions are available to the agents:

- $Act_{\mathbf{S}} = \{sendP_i \mid i \geq 0\} \cup \{acceptP_i \mid i \geq 0\} \cup \{\lambda\}$,
- $Act_{\mathbf{R}} = \{\lambda, nonce, stop\} \cup \{acceptP_i \mid i > 0\}$,
- $Act_{\mathbf{I}} = \{\lambda\} \cup \{dropP_i \mid i > 0\} \cup \{fakeP_i \mid i > 0\} \cup \{acceptP_i \mid i > 0\}$,

where λ stands for no action.

The protocols executed by agents are defined as follows:

Sender:

- $P_{\mathbf{S}}([\cdot]) = \{\lambda\}$
- $P_{\mathbf{S}}([recv(n_{\mathbf{R}})]) = \{sendP_0\}$
- $P_{\mathbf{S}}([sent(\mathbf{R}, P_0)]) = \{sendP_1\}$
- $P_{\mathbf{S}}([sent(\mathbf{R}, P_{i-1}), sent(\mathbf{R}, P_i)]) = \{sendP_{i+1}\}$,
- $P_{\mathbf{S}}([sent(\mathbf{R}, P_{i-1}), sent(\mathbf{R}, P_i), sent(\mathbf{R}, P_{i+1})]) = \{acceptP_{i-1}, \lambda\}$, for $i \geq 0$.

Intruder:

- $P_{\mathbf{I}}([\cdot]) = P_{\mathbf{I}}([recv(n_{\mathbf{R}})]) = \{\lambda\}$
- $P_{\mathbf{I}}([recv(P_0)]) = \{\lambda, dropP_1, fakeP_1\}$
- $P_{\mathbf{I}}([recv(P_i), recv(P_{i+1})]) = \{\lambda, dropP_{i+2}, fakeP_{i+2}\}$, for $i \geq 0$
- $P_{\mathbf{I}}([recv(P_{i-1}), recv(P_i), recv(P_{i+1})]) = \{acceptP_{i-1}, \lambda\}$, for $i > 0$
- $P_{\mathbf{I}}([recv(P_0), recv(P_1), send(\mathbf{R}, P_1')]) = \{fakeP_2\}$,
- $P_{\mathbf{I}}([recv(P_0), recv(P_1), send(\mathbf{R}, P_1'), recv(P_2)]) = \{\lambda\}$,
- $P_{\mathbf{I}}([recv(P_0), recv(P_1), send(\mathbf{R}, P_1'), recv(P_2), send(\mathbf{R}, P_2')]) = \{\lambda\}$,
- $P_{\mathbf{I}}([recv(P_{i-1}), recv(P_i), recv(P_{i+1}), send(\mathbf{R}, P_{i+1}')]) = \{\lambda\}$, for $i > 0$

Receiver:

- $P_{\mathbf{R}}([\cdot]) = \{nonce\}$
- $P_{\mathbf{R}}([sent(\mathbf{S}, n_{\mathbf{R}})]) = P_{\mathbf{R}}([recv(P_0)]) = \{\lambda\}$
- $P_{\mathbf{R}}([recv(P_i), recv(P_{i+1})]) = \{\lambda\}$, for $i \geq 0$
- $P_{\mathbf{R}}([recv(P_{i-1}), recv(P_i), recv(P_{i+1})]) = \{acceptP_i\}$, for $i > 0$
- $P_{\mathbf{R}}([stop]) = P_{\mathbf{R}}([recv(P_0), recv(P_2)]) = \{stop\}$,
- $P_{\mathbf{R}}([recv(P_{i-1}), recv(P_i), recv(P_{i+2})]) = \{stop\}$, for $i > 0$
- $P_{\mathbf{R}}([recv(P_0), recv(P_1')]) = \{\lambda\}$,
- $P_{\mathbf{R}}([recv(P_0), recv(P_1'), recv(P_2')]) = \{stop\}$,
- $P_{\mathbf{R}}([recv(P_0), recv(P_1'), recv(P_2)]) = \{stop\}$,
- $P_{\mathbf{R}}([recv(P_{i-1}), recv(P_i), recv(P_{i+1}')]) = \{stop\}$, for $i > 0$

The evolution of the system during the execution of TESLA is defined by means of an evolution function $t : (L_{\mathbf{S}} \times L_{\mathbf{R}} \times L_{\mathbf{I}}) \times Act \rightarrow 2^{L_{\mathbf{S}} \times L_{\mathbf{R}} \times L_{\mathbf{I}}}$, where Act is a subset of $Act_{\mathbf{S}} \times Act_{\mathbf{R}} \times Act_{\mathbf{I}}$. More precisely, let us assume that the system starts from the initial state $([\cdot], [\cdot], [\cdot])$, and let rv denote the even *recv*, and st the even *sent*. Then, the evolution function for TESLA is defined as follows:

Intruder forwards all packets

1. $t(([\cdot], [\cdot], [\cdot]), (\lambda, nonce, \lambda)) = ([rv(n_{\mathbf{R}})], [st(\mathbf{S}, n_{\mathbf{R}})], [rv(n_{\mathbf{R}})])$;
2. $t([rv(n_{\mathbf{R}})], [st(\mathbf{S}, n_{\mathbf{R}})], [rv(n_{\mathbf{R}})]), (sendP_0, \lambda, \lambda))$ $= ([st(\mathbf{R}, P_0)], [rv(P_0)], [rv(P_0)])$;
3. $t([st(\mathbf{R}, P_0)], [rv(P_0)], [rv(P_0)]), (sendP_1, \lambda, \lambda)) =$ $([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1)], [rv(P_0), rv(P_1)], [rv(P_0), rv(P_1)])$;
4. $t([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1)], [rv(P_0), rv(P_1)], [rv(P_0), rv(P_1)])$, $(sendP_2, \lambda, \lambda)) = ([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)]$, $[rv(P_0), rv(P_1), rv(P_2)], [rv(P_0), rv(P_1), rv(P_2)])$;
5. $t([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)],\ [rv(P_0), rv(P_1), rv(P_2)]$, $[rv(P_0), rv(P_1), rv(P_2)]), (acceptP_0, acceptP_1, acceptP_0)) =$ $([st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [rv(P_1), rv(P_2)], [rv(P_1), rv(P_2)])$;
6. $t([st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [rv(P_1), rv(P_2)], [rv(P_1), rv(P_2)])$, $(sendP_3, \lambda, \lambda)) = ([st(\mathbf{R}, P_1), st(\mathbf{R}, P_2), st(\mathbf{R}, P_3)]$, $[rv(P_1), rv(P_2), rv(P_3)], [rv(P_1), rv(P_2), rv(P_3)])$;
7. $t([st(\mathbf{R}, P_1), st(\mathbf{R}, P_2), st(\mathbf{R}, P_3)],\ [rv(P_1), rv(P_2), rv(P_3)]$, $[rv(P_1), rv(P_2), rv(P_3)]), (acceptP_1, acceptP_2, acceptP_1)) =$ $([st(\mathbf{R}, P_2), st(\mathbf{R}, P_3)], [rv(P_2), rv(P_3)], [rv(P_2), rv(P_3)]])$;

...

- $t([st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1})], [rv(P_n), rv(P_{n+1})]$, $[rv(P_n), rv(P_{n+1})]), (sendP_{n+2}, \lambda, \lambda)) =$ $([st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1}), st(\mathbf{R}, P_{n+2})],\ [rv(P_n), rv(P_{n+1})$, $rv(P_{n+2})], [rv(P_n), rv(P_{n+1}), rv(P_{n+2})])$, for $n > 0$;
- $t([st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1}), st(\mathbf{R}, P_{n+2})]$, $[rv(P_n), rv(P_{n+1}), rv(P_{n+2})], [rv(P_n), rv(P_{n+1}), rv(P_{n+2})])$, $(acceptP_n, acceptP_{n+1}, dropP_n)) = ([st(\mathbf{R}, P_{n+1}), st(\mathbf{R}, P_{n+2})]$, $[rv(P_{n+1}), rv(P_{n+2})], [rv(P_{n+1}), rv(P_{n+2})])$, for $n > 0$;

Intruder blocks packet P_1

1. $t(([st(\mathbf{R}, P_0)], [rv(P_0)], [rv(P_0)]),\ (sendP_1, \lambda, dropP_1)) =$ $([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1)], [rv(P_0)], [rv(P_0), rv(P_1)])$;
2. $t([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1)], [rv(P_0)], [rv(P_0), rv(P_1)])$, $(sendP_2, \lambda, \lambda)) = ([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)]$, $[rv(P_0), rv(P_2)], [rv(P_0), rv(P_1), rv(P_2)])$;
3. $t([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [rv(P_0), rv(P_2)]$, $[rv(P_0), rv(P_1), rv(P_2)]), (\lambda, stop, \lambda)) = ([st(\mathbf{R}, P_0)$, $st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [stop], [rv(P_0), rv(P_1), rv(P_2)])$;
4. $t([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [stop], [rv(P_0), rv(P_1)$, $rv(P_2)]), (\lambda, stop, \lambda)) = ([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)]$, $[stop], [rv(P_0), rv(P_1), rv(P_2)])$;

Intruder blocks packet P_n, $n > 1$

1. $t(([st(\mathbf{R}, P_{n-2}), st(\mathbf{R}, P_{n-1})], [rv(P_{n-2}), rv(P_{n-1})]$, $[rv(P_{n-2}), rv(P_{n-1})]), (sendP_n, \lambda, dropP_n)) =$ $([st(\mathbf{R}, P_{n-2}), st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n)]$, $[rv(P_{n-2}), rv(P_{n-1})], [rv(P_{n-2}), rv(P_{n-1}), rv(P_n)])$;
2. $t(([st(\mathbf{R}, P_{n-2}), st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n)]$, $[rv(P_{n-2}), rv(P_{n-1})], [rv(P_{n-2}), rv(P_{n-1}), rv(P_n)])$, $(accept_{n-2}, \lambda, accept_{n-2})) = ([st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n)]$, $[rv(P_{n-2}), rv(P_{n-1})], [rv(P_{n-1}), rv(P_n)])$;
3. $t(([st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n)], [rv(P_{n-2}), rv(P_{n-1})]$, $[rv(P_{n-1}), rv(P_n)]), (sendP_{n+1}, \lambda, \lambda)) = ([st(\mathbf{R}, P_{n-1})$, $st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1})], [rv(P_{n-2}), rv(P_{n-1})$, $rv(P_{n+1})], [rv(P_{n-1}), rv(P_n), rv(P_{n+1})])$;
4. $t(([st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1})], [rv(P_{n-2})$, $rv(P_{n-1}), rv(P_{n+1})], [rv(P_{n-1}), rv(P_n), rv(P_{n+1})])$, $(\lambda, stop, \lambda)) = ([st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1})]$, $[stop], [rv(P_{n-1}), rv(P_n), rv(P_{n+1})])$;
5. $t([st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1})], [stop]$, $[rv(P_{n-1}), rv(P_n), rv(P_{n+1})]), (\lambda, stop, \lambda)) =$ $([st(\mathbf{R}, P_{n-1}), st(\mathbf{R}, P_n), st(\mathbf{R}, P_{n+1})], [stop]$, $[rv(P_{n-1}), rv(P_n), rv(P_{n+1})])$;

Intruder fakes packet P_1

1. $t(([st(\mathbf{R}, P_0)], [rv(P_0)], [rv(P_0)]), (sendP_1, \lambda, fakeP_1)) =$ $([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1)], [rv(P_0), rv(P_1')]$, $[rv(P_0), rv(P_1), st(\mathbf{R}, P_1')])$;

2.a $t([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1)], [rv(P_0), rv(P_1')], [rv(P_0), rv(P_1)$, $st(\mathbf{R}, P_1')]), (sendP_2, \lambda, fakeP_2)) = ([st(\mathbf{R}, P_0)$, $st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [rv(P_0), rv(P_1'), rv(P_2')]$, $[rv(P_0), rv(P_1), st(\mathbf{R}, P_1'), rv(P_2)], st(\mathbf{R}, P_2'))$;

3.a $t([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [rv(P_0), rv(P_1'), rv(P_2')]$, $[rv(P_0), rv(P_1), st(\mathbf{R}, P_1'), rv(P_2), st(\mathbf{R}, P_2')]), (\lambda, stop, \lambda)) =$ $([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [stop]$, $[rv(P_0), rv(P_1), st(\mathbf{R}, P_1'), rv(P_2), st(\mathbf{R}, P_2')])$;

4.a $t([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)], [stop]$, $[rv(P_0), rv(P_1), st(\mathbf{R}, P_1'), rv(P_2), st(\mathbf{R}, P_2')])$, $(\lambda, stop, \lambda)) = ([st(\mathbf{R}, P_0), st(\mathbf{R}, P_1), st(\mathbf{R}, P_2)]$, $[stop], [rv(P_0), rv(P_1), st(\mathbf{R}, P_1'), rv(P_2), st(\mathbf{R}, P_1')])$;

2.b $t([st(\mathbf{R},P_0),st(\mathbf{R},P_1)],[rv(P_0),rv(P_1')],$
$[rv(P_0),rv(P_1),st(\mathbf{R},P_1')]),(sendP_2,\lambda,\lambda)) =$
$([st(\mathbf{R},P_0),st(\mathbf{R},P_1),st(\mathbf{R},P_2)],[rv(P_0),rv(P_1'),$
$rv(P_2)],[rv(P_0),rv(P_1),st(\mathbf{R},P_1'),rv(P_2)]);$

3.b $t([st(\mathbf{R},P_0),st(\mathbf{R},P_1),st(\mathbf{R},P_2)],[rv(P_0),rv(P_1'),$
$st(\mathbf{R},P_1'),rv(P_2)],[rv(P_0),rv(P_1),rv(P_2)]),(\lambda,stop,\lambda))$
$=([st(\mathbf{R},P_0),st(\mathbf{R},P_1),st(\mathbf{R},P_2)],[stop],$
$[rv(P_0),rv(P_1),st(\mathbf{R},P_1'),rv(P_2)]);$

4.b $t([st(\mathbf{R},P_0),st(\mathbf{R},P_1),st(\mathbf{R},P_2)],[stop],[rv(P_0),rv(P_1),$
$st(\mathbf{R},P_1'),rv(P_2)]),(\lambda,stop,\lambda)) = ([st(\mathbf{R},P_0),st(\mathbf{R},P_1),$
$st(\mathbf{R},P_2)],[stop],[rv(P_0),rv(P_1),st(\mathbf{R},P_1'),rv(P_2)]);$

Intruder fakes packet P_n, $n>1$

1. $t(([st(\mathbf{R},P_{n-2}),st(\mathbf{R},P_{n-1})],[rv(P_{n-2}),rv(P_{n-1})],$
$[rv(P_{n-2}),rv(P_{n-1})]),(sendP_n,\lambda,fakeP_n)) =$
$([st(\mathbf{R},P_{n-2}),st(\mathbf{R},P_{n-1}),st(\mathbf{R},P_n)],[rv(P_{n-2}),$
$rv(P_{n-1}),rv(P_n')],[rv(P_{n-2}),rv(P_{n-1}),rv(P_n)]);$
2. $t(([st(\mathbf{R},P_{n-2}),st(\mathbf{R},P_{n-1}),st(\mathbf{R},P_n)],[rv(P_{n-2}),$
$rv(P_{n-1}),rv(P_n')],[rv(P_{n-2}),rv(P_{n-1}),rv(P_n)]),$
$(\lambda,stop,\lambda)) = ([st(\mathbf{R},P_{n-2}),st(\mathbf{R},P_{n-1}),st(\mathbf{R},P_n)],$
$[stop],[rv(P_{n-2}),rv(P_{n-1}),rv(P_n)]);$
3. $t([st(\mathbf{R},P_{n-2}),st(\mathbf{R},P_{n-1}),st(\mathbf{R},P_n)],[stop],$
$[rv(P_{n-2}),rv(P_{n-1}),rv(P_n)]),(\lambda,stop,\lambda)) =$
$([st(\mathbf{R},P_{n-2}),st(\mathbf{R},P_{n-1}),st(\mathbf{R},P_n)],[stop],$
$[rv(P_{n-2}),rv(P_{n-1}),rv(P_n)]);$

The above evolution function determines not only the set of reachable global states $S \subseteq L_{\mathbf{S}} \times L_{\mathbf{R}} \times L_{\mathbf{I}}$, but also gives the transition relation T; namely, for all the s, $s' \in S$, $(s,s') \in T$ iff there exists $act \in Act$ such that $t(s,act) = s'$.

We have now defined states, actions, protocols, and transitions for the model M for TESLA. To conclude, we define a valuation function $\mathcal{V} : S \to 2^{\mathcal{PV}}$ and the awareness functions $\mathbb{A}_X : L_X \to 2^{\mathcal{WF}(TDL)}$, for $X \in \{\mathbf{S},\mathbf{R},\mathbf{I}\}$. We first introduce the following set $\mathcal{PV}$ of propositional variables, which we find useful in analysis of the TESLA scenario:

$$\mathcal{PV} = \{has_{\mathbf{R}}(m), sent_{\mathbf{S}}(m), received_{\mathbf{R}}(m),\\ dropped_{\mathbf{I}}(m), faked_{\mathbf{I}}(m) \mid m \in M_{\mathbf{S}} \cup M_{\mathbf{I}}\}$$

We define $\mathcal{V} : S \to 2^{\mathcal{PV}}$ by assuming that:

- for $i>0$, $has_{\mathbf{R}}(t_i) \in \mathcal{V}(s)$ if there exist packets P_{i-1}, P_i and P_{i+1} such that $f(k_i) \sqsubseteq P_{i-1}$, $t_i \sqsubseteq P_i$, $k_i \sqsubseteq P_{i+1}$, $recv(P_{i-1}) \in l_{\mathbf{R}}(s)$, $recv(P_i) \in l_{\mathbf{R}}(s)$ and $recv(P_{i+1}) \in l_{\mathbf{R}}(s)$,
- $sent_{\mathbf{S}}(m) \in \mathcal{V}(s)$ if there exists packet P_i such that $m \sqsubseteq P_i$ and $sent(\mathbf{R},P_i) \in l_{\mathbf{S}}(s)$, for any $m \in M_{\mathbf{S}}$,
- $received_{\mathbf{R}}(m) \in \mathcal{V}(s)$ if $recv(m) \in l_{\mathbf{R}}(s)$, for any $m \in M_{\mathbf{S}} \cup M_{\mathbf{I}}$,
- $dropped_{\mathbf{I}}(m) \in \mathcal{V}(s)$ if $recv(m) \notin l_{\mathbf{R}}(s)$ and $recv(m) \in l_{\mathbf{I}}(s)$, for any $m \in M_{\mathbf{S}}$,
- $faked_{\mathbf{I}}(m) \in \mathcal{V}(s)$ if there exist packets P_j such that $m \sqsubseteq P_j$ and $send(\mathbf{R},P_j) \in l_{\mathbf{I}}(s)$, for any $m \in M_{\mathbf{S}} \cup M_{\mathbf{I}}$.

For $X \in \{\mathbf{S},\mathbf{I}\}$, we take the following awareness function $\mathbb{A}_X : L_X \to 2^{\mathcal{WF}(TDL)}$: $\mathbb{A}_X(l) = \emptyset$, for any $l \in L_X$. In turn, $\mathbf{R}$'s awareness function $\mathbb{A}_{\mathbf{R}} : L_{\mathbf{R}} \to 2^{\mathcal{WF}(TDL)}$ is defined as follows. For any $l \in L_{\mathbf{R}}$, $\alpha \in \mathbb{A}_{\mathbf{R}}(l)$ if one of the following holds:

- $\alpha = received_{\mathbf{R}}(m)$ and $recv(m) \in l$ and $m \in M_{\mathbf{S}} \cup M_{\mathbf{I}}$,
- $\alpha = faked_{\mathbf{I}}(m)$ and $l = [stop]$ and $m \in M_{\mathbf{S}} \cup M_{\mathbf{I}}$,
- $\alpha = dropped_{\mathbf{I}}(m)$ and $l = [stop]$ and $m \in M_{\mathbf{S}}$,
- $\alpha = has_{\mathbf{R}}(m)$ and ($recv(m) \in l$ or $\exists m'$ such that $m \sqsubseteq m'$ and $recv(m') \in l$) and $m \in M_{\mathbf{S}} \cup M_{\mathbf{I}}$.

We have now completed defining the model M for TESLA.

Note that when we put a bound on the number of packets to be sent, M becomes finite.

Given the model M as defined above, we proceed to the semantical analysis of the TESLA protocol. First, we would like to establish whether or not TESLA satisfies the desired security property: *"the receiver does not accept as authentic any message unless it was actually sent by the sender"*. It can be expressed by the following TDL formula: for any $i>0$,

$$has_{\mathbf{R}}(t_i) \Rightarrow (sent_{\mathbf{S}}(P_{i-1}) \wedge sent_{\mathbf{S}}(P_i) \wedge sent_{\mathbf{S}}(P_{i+1})) \quad (1)$$

It is easy (albeit tedious) to check manually that formula 1 is valid in M.

As one can notice, formula 1 is a pure propositional formula, so in order to verify the TESLA security property there is no need to use strong formalisms like epistemic or temporal logics. Indeed, this property can be checked via process algebras [5] or via Lynch-Vaandrager automata [3]. However, unlike the process algebras and Lynch-Vaandrager automata, the TDL language together with its semantic model is a powerful formalism that allows us to model and verify other properties of TESLA. In particular, using the model M and TDL we can show that TESLA satisfies a much stronger property: *"the receiver does not accept as authentic any message unless he knows that it was actually sent by the sender"*. Indeed it can be checked that, for any $i>0$,

$$M \models \quad has_{\mathbf{R}}(t_i) \Rightarrow \mathcal{K}_{\mathbf{R}}(sent_{\mathbf{S}}(P_{i-1}) \wedge \\ sent_{\mathbf{S}}(P_i) \wedge sent_{\mathbf{S}}(P_{i+1})) \quad (2)$$

Note that our model M of TESLA does not fulfil the contrappositive of the above, i.e. the following holds:

$$M \not\models \quad \mathcal{K}_{\mathbf{R}}(sent_{\mathbf{S}}(P_{i-1}) \wedge sent_{\mathbf{S}}(P_i) \wedge \\ sent_{\mathbf{S}}(P_{i+1})) \Rightarrow has_{\mathbf{R}}(t_i) \quad (3)$$

The above seems reasonable, and it does not violate the security property given the presence of the intruder.

Further, we can check that TESLA actually meets the following property: *"it is always the case that the receiver does not accept as authentic any message unless he knows that it was actually sent by the sender"*.

$$M \models \quad \mathrm{A}\Box(has_{\mathbf{R}}(t_i) \Rightarrow \mathcal{K}_{\mathbf{R}}(sent_{\mathbf{S}}(P_{i-1}) \wedge \\ sent_{\mathbf{S}}(P_i) \wedge sent_{\mathbf{S}}(P_{i+1}))) \quad (4)$$

The principals know about the presence of the intruder, so, the following should holds: for some $i>0$:

$$M \models K_{\mathbf{S}}\mathrm{E}\Diamond(sent_{\mathbf{S}}(P_i) \wedge \neg received_{\mathbf{R}}(P_i)) \quad (5)$$

Indeed, a manual checking of M that we have done confirmed the above observation.

Also, by analysing TESLA one can expect that the protocol should fulfil the following property: *if a packet is faked, then the receiver should discover this.* It expresses the fact that the receiver is able to check the source of messages. This can be formalised as follows:

$$M \models faked_{\mathbf{I}}(P_i) \Rightarrow \mathcal{D}_{\mathbf{R}}(faked_{\mathbf{I}}(P_i)) \quad (6)$$

Again, a manual checking of M confirmed the above.

Investigating the TESLA model M, we could observe that if the receiver obtains some packets P_{i-1}, P_i, and P_{i+1} with a message $t_i \sqsubseteq P_i$, and he does not accept t_i as authentic, then he knows that at least one of the packets was not sent by the intended sender. Moreover, if a packet was indeed faked, the receiver is able to deduce this fact. So, property 7 should hold, and a manual check of M that we have done

confirmed this. For any $i > 0$,

$$M \models (received_{\mathbf{R}}(P_{i-1}) \wedge received_{\mathbf{R}}(P_i) \wedge received_{\mathbf{R}}(P_{i+1}) \wedge \neg has_{\mathbf{R}}(t_i)) \Rightarrow \big(\mathcal{K}_{\mathbf{R}}(\neg sent_{\mathbf{S}}(P_{i-1}) \vee \neg sent_{\mathbf{S}}(P_i) \vee \neg sent_{\mathbf{S}}(P_{i+1})) \wedge (\mathcal{D}_{\mathbf{R}}(fake_{\mathbf{I}}(P_{i-1})) \vee \mathcal{D}_{\mathbf{R}}(fake_{\mathbf{I}}(P_i)) \vee \mathcal{D}_{\mathbf{R}}(fake_{\mathbf{I}}(P_{i+1})))\big) \quad (7)$$

Messages are sent to the receiver according to the schedule. This schedule is known to the intruder as well. So, when the intruder tries to fake the current transmission, he has to follow the sending schedule, otherwise, the receiver will stop the reception immediately. This means that the intruder has to send a packet at each interval, which was agreed by the sender and the receiver at the beginning of the transmission under consideration. This observation can be formalised as follows. For any $i > 0$,

$$M \models \mathrm{A}\Box(dropped_{\mathbf{I}}(P_i) \Rightarrow \mathcal{D}_{\mathbf{R}}(dropped_{\mathbf{I}}(P_i)) \quad (8)$$

It can be checked that property 8 also holds.

6. CONCLUSION

In the paper we have proposed a new security-specialised logic TDL defined on a computationally-grounded semantics, and with a sound and complete axiomatisation. TDL employs different concepts of knowledge (based on the notion of awareness), as well as time in order to represent the knowledge of principals that operate in an environment that is under control of an intruder. Further, we have applied the logic to reason about TESLA, which we have formalised in the interpreted system framework. A temporal epistemic analysis has allowed us not only to reconfirm the TESLA authentication property but also to check more sophisticated specifications.

Obviously we do not suggest one should check security protocols manually as done here. Work is in progress to extend a BDD-based model checker [17] to encode the epistemic notions described here. Since the semantics of TDL is defined on interpreted systems we are hopeful that the exercise conduced here manually can be automated. Since TDL has a computationally-grounded semantics, a model checking technique for it can be defined, which is the scope of a further investigation.

7. REFERENCES

[1] M. Abadi and M. R. Tuttle. A semantics for a logic of authentication. In *Proc. of PODC '91*, pp. 201–216. ACM Press, 1991.

[2] R. Accorsi, D. Basin, and L. Viganò. Towards an awareness-based semantics for security protocol analysis. In *Post Proc. of LACPV'01*, volume 55(1) of ENTCS, pp 5–24. Elsevier, 2003.

[3] M. Archer. Proving correctness of the basic TESLA multicast stream authentication protocol with TAME. In *Proc. of WITS'02*, 2002.

[4] C. Boyd and W. Mao. On a limitation of BAN logic. In *Proc. of EUROCRYPT '93*, pp. 240–247. Springer-Verlag, 1994.

[5] P. J. Broadfoot and G. Lowe. Analysing a stream authentication protocol using model checking. In *Proc. of ESORICS'02*, pp. 146–161. Springer-Verlag, 2002.

[6] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *Proc. of the Royal Society of London A*, volume 426, pp. 233–271 , 1989.

[7] E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons for branching-time temporal logic. In *Proc. of WLP*, volume 131 of *LNCS*, pp. 52–71. Springer-Verlag, 1981.

[8] C. Dixon, M. C. Fernandez Gago, M. Fisher, and W. van der Hoek. Using temporal logics of knowledge in the formal verification of security protocols. In *Proc. of TIME'04*, pp. 148–151. IEEE, 2004.

[9] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Trans. Inf. Theory*, 29(2):198–208, 1983.

[10] R. Fagin and J. Y. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34(1):39–76, 1988.

[11] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[12] Li Gong, R. Needham, and R. Yahalom. Reasoning About Belief in Cryptographic Protocols. In *Proceedings of IEEE Symposium on Research in Security and Privacy*, pp. 234–248. IEEE Press, 1990.

[13] J. Y. Halpern, Y. Moses, and M. Y. Vardi. Algorithmic knowledge. In *Proc. of TARK 1994*, pages 255–66. Morgan Kaufmann Publishers, 1994.

[14] J.Y. Halpern and R. Pucella. Modeling Adversaries in a Logic for Security Protocol Analysis. In *Proc. of FASec'02*, volume 2629 of *LNCS*, pp. 115–132. Springer-Verlag, 2002.

[15] R. Kailar and V. D. Gligor. On belief evolution in authentication protocols. In *Proc. of Computer Security Foundations Workshop IV*, pp. 103–116. IEEE Press, 1991.

[16] A. Lomuscio and B. Woźna. A combination of explicit and deductive knowledge with branching time: completeness and decidability results. *In Post-proc. of DALT'05*, volume 3904 of LNAI. 2006. To appear.

[17] A. Lomuscio and F. Raimondi. *MCMAS: A model checker for multi-agent systems.* In *Proc. of TACAS'06*. Springer Verlag, March, 2006. To appear.

[18] D. M. Nessett. A critique of the Burrows Abadi Needham logic. *Operating Systems Review*, 24(3):35–38, 1990.

[19] A. Perrig, R. Canetti, J.D. Tygar, and Dawn X. Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pp. 56–73, 2000.

[20] R. Pucella. Deductive Algorithmic Knowledge. In *Proc. of SAIM'04*, AI&M 22-2004, 2004.

[21] R. Ramanujam and S. P. Suresh. Deciding knowledge properties of security protocols. In *Proc. of TARK'05*, pp. 218–235, 2005.

[22] M. Wooldridge. *Reasoning about Rational Agents*. MIT Press, 2000.

On the Logic of Coalitional Games

Thomas Ågotnes
Department of Informatics, University of Bergen
PB. 7800, N-5020 Bergen, Norway
agotnes@ii.uib.no

Wiebe van der Hoek
Department of Computer Science, University of Liverpool
Liverpool L69 7ZF, UK
wiebe@csc.liv.ac.uk

Michael Wooldridge
Department of Computer Science, University of Liverpool
Liverpool L69 7ZF, UK
mjw@csc.liv.ac.uk

ABSTRACT

We develop a logic for representing and reasoning about coalitional games without transferable payoffs. Although a number of logics of cooperation have been proposed over the past decade (notably Coalition Logic [14] and Alternating-time Temporal Logic [1]), these logics focused primarily on the issue of strategic cooperative ability – what states a coalition can effectively enforce – and have tended to ignore the essential issue of the preferences that agents have over such states; in addition, the connection between such logics and coalitional games, in the sense of cooperative game theory, is left implicit. The Coalitional Game Logic (CGL) that we develop in this paper differs from such previous logics in two important respects. First, CGL includes operators that make it directly possible to represent an agent's preferences over outcomes. Second, we interpret formulae of CGL directly with respect to coalitional games without transferable payoff, thereby establishing an explicit link between formulae of the logic and properties of coalitional games. We show that these coalitional games cannot be seen directly as models for Coalition Logic. We give a complete axiomatization of CGL, prove that it is expressively complete with respect to coalitional games without transferable payoff, show that the satisfiability problem for the logic is NP-complete, and to illustrate its use, we show how the logic can be used to characterise axiomatically a number of well-known solution concepts for coalitional games, including for example non-emptiness of the core.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence-Multiagent Systems; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Modal logic*

General Terms

Theory

Keywords

Game theory, coalitional games, logic

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

If one aims to build software agents that must inhabit and act within some particular environment, then it is very natural to equip such agents with formal *representations* of the environment, in order to permit them to effectively select and execute actions within it. Previous work on knowledge representation in multi-agent systems may be broadly divided into two distinct categories. In the 1990s, considerable emphasis was placed on representations of the *cognitive structure* of agents – their beliefs, desires, intentions, and so forth. Key issues in such representations include for example the way in which communicative actions (speech acts) affect the mental states of conversation participants, and how we can characterise the mental states of agents engaged in teamwork [5]. More recently, an increasing body of work has focused instead on logics that make it possible to represent the strategic structure of multi-agent environments, and in particular, the powers that agents or groups of agents have in such environments [1, 13]. Such logics have proved to have many important applications, for example in the specification and verification of social choice mechanisms [13].

One significant feature of these cooperation logics is that they have a close link with formal games: the semantic models underpinning Coalition Logic (which corresponds to the next-time fragment of ATL [8]) can be understood as extensive games of almost perfect information [13, p.34]. Further links between logics of strategic ability and formal games have been established elsewhere – for example, van der Hoek and colleagues showed how solution concepts such as Nash equilibrium can be axiomatized in a variant of ATL [16]. However, as a general formalism for reasoning about cooperative games, ATL-like cooperation logics have limitations: their modal structures are only intended to support reasoning about cooperative ability, and they provide no direct mechanism for capturing preferences, which are of course a critical component of all but the very simplest multi-agent scenarios.

Our aim in this paper is to develop a logic that is intended to directly and transparently support reasoning about coalitional games – and more precisely, coalitional games without transferable payoff [11, p.268]. We develop a *Coalitional Game Logic* (CGL). Syntactically, CGL appears quite similar to ATL and Coalition Logic, in that it contains modal cooperation expressions. However, it differs from these previous logics in several important respects. First, CGL includes operators that make it *directly* possible to represent an agent's preferences over outcomes. Second, we interpret formulae of CGL directly with respect to coalitional games without transferable payoff, thereby establishing an explicit link between formulae of the logic and properties of coalitional games. Third, we show that a coalitional game indeed cannot be seen as a model for coalition logic with outcomes of the game as states.

Following the presentation of the syntax and semantics of the

logic in the next section, we give a number of technical results relating to it, as follows. First, we prove that the logic is expressively complete with respect to coalitional games without transferable payoff, in the sense that for any two different coalitional games, there exists a formula of CGL that will be true in one game and false in the other. We then give an axiomatization of CGL, and show that it is complete. With respect to model checking and satisfiability, we show that while model checking for the logic is tractable, the satisfiability problem for CGL is NP-complete. We then compare the semantics of CGL and coalition logic. Finally, to illustrate the utility of the logic, we show how to axiomatically characterise a number of well-known solution concepts for coalitional games, including for example non-emptiness of the core.

2. A LOGIC FOR COALITIONAL GAMES

A *coalitional game* (without transferable payoff) is an $(m+3)$-tuple [11, p.268]:

$$\Gamma = \langle N, \Omega, V, \sqsupseteq_1, \ldots, \sqsupseteq_m \rangle \text{ where:}$$

- $N = \{1, \ldots, m\}$ is a non-empty set of *agents*;
- $\Omega = \{\omega_1, \ldots, \omega_n\}$ is a non-empty set of *outcomes*;
- $V : (2^N \setminus \emptyset) \rightarrow 2^\Omega$ is the *characteristic function* of Γ, which for every non-empty coalition C defines the choices $V(C)$, such that if $\omega \in V(C)$ then C can choose outcome ω; and
- $\sqsupseteq_i \subseteq \Omega \times \Omega$ is a complete, reflexive, and transitive *preference relation*, for each agent $i \in N$.

The above definition also includes the additional assumption, used in this paper, that the set of outcomes is finite.

Readers familiar with ATL [1] or Coalition Logic [14] may be tempted to interpret V as an effectivity function and the outcomes Ω as states of the world. This is *not* the intended interpretation, and indeed, we show in Section 6 that there is no direct mapping between the coalition models of [14] and coalitional games by interpreting outcomes Ω as states and V as an effectivity function. For example, it is perfectly consistent in a coalitional game that, at the same time, disjoint coalitions have the abilities to choose *different* outcomes.

We now define a logic for representing the properties of such games. The language is in two parts. First, given a set of outcome symbols Σ_Ω, we have an *outcome language* $\mathcal{L}_o$, defined by the grammar φ_o, below, which expresses the properties of outcomes. The outcome symbols themselves are the main constructs of this language; a formula such as $\omega_1 \vee \omega_2$ (where $\omega_1, \omega_2 \in \Sigma_\Omega$) means that the outcome corresponds to either ω_1 or ω_2. Next, given a set of agent symbols Σ_N and a set of coalition symbols Σ_C, we have a *cooperation language* $\mathcal{L}_c$, for expressing the properties of coalitional cooperation, and the preferences that agents have over possible outcomes. This language is generated by the grammar φ_c below. $\mathcal{L}_c$ has two main constructs. First, $\omega_1 \succeq_i \omega_2$ (where $\omega_1, \omega_2 \in \Sigma_\Omega$, $i \in \Sigma_N$) expresses the fact that agent i either prefers outcome ω_1 over outcome ω_2, or is indifferent between the two. Second, $\langle C \rangle \varphi$ (where $C \in \Sigma_C$) says that C can choose an outcome in which the formula φ will be true. This construct may seem syntactically similar to counterparts in ATL and Coalition Logic, but it stands here for a fundamentally different concept due to the semantic differences mentioned above. Recall that we construct formulae with respect to a set $\Sigma = \Sigma_N \cup \Sigma_C \cup \Sigma_\Omega$ of symbols, where Σ_N is a set of symbols for agents, Σ_C is a set of symbols for coalitions and Σ_Ω is a set of symbols for outcomes. Formally, we have the following:

$$\begin{array}{lcl} \varphi_o & ::= & \sigma_\omega \mid \neg\varphi_o \mid \varphi_o \vee \varphi_o \\ \varphi_c & ::= & (\sigma_\omega \succeq_{\sigma_i} \sigma_{\omega'}) \mid \langle \sigma_C \rangle \varphi_o \mid \neg\varphi_c \mid \varphi_c \vee \varphi_c \end{array}$$

where $\sigma_i \in \Sigma_N$ is an agent symbol, $\sigma_C \in \Sigma_C$ is a coalition symbol, and $\sigma_\omega, \sigma_{\omega'} \in \Sigma_\Omega$ are outcome symbols.

We will usually exploit the direct correspondence between symbols for outcomes/agents and the outcomes/agents that appear in games. In this paper, we will henceforth assume a one-to-one correspondence between Σ_Ω and Ω, between Σ_N and N and between Σ_C and 2^N. So we assume that $\Sigma_\Omega = \{\sigma_\omega : \omega \in \Omega\}$, $\Sigma_N = \{\sigma_i : i \in N\}$ and $\Sigma_C = \{\sigma_C : C \subseteq N\}$. The languages are parameterised by the sets $\Sigma_\Omega, \Sigma_N, \Sigma_C$, i.e., as a consequence of the assumption, by some set N of agents and set Ω of outcomes. In the following, we assume that these two parameters — and thus the languages — are fixed.

An $\mathcal{L}_c$ formula γ is interpreted in a coalitional game Γ as follows, where $\Gamma \models \gamma$ means that γ is true in Γ. When we write $\Gamma \models \gamma$, it is implicitly assumed that the Γ corresponds to the parameters of the language in the way mentioned above, i.e., that the agents and outcomes in the model are exactly the sets used to parameterise our language. First, we define the satisfaction of a $\mathcal{L}_o$ formula α in an outcome ω of a game Γ, written $\Gamma, \omega \models \alpha$:

$$\Gamma, \omega \models \sigma_{\omega'} \text{ iff } \omega = \omega'$$

$$\Gamma, \omega \models \neg\varphi \text{ iff not } \Gamma, \omega \models \varphi$$

$$\Gamma, \omega \models \varphi \vee \psi \text{ iff } \Gamma, \omega \models \varphi \text{ or } \Gamma, \omega \models \psi$$

Satisfaction of γ in Γ is then defined as follows:

$$\Gamma \models (\sigma_{\omega_1} \succeq_{\sigma_i} \sigma_{\omega_2}) \text{ iff } (\omega_1 \sqsupseteq_i \omega_2)$$

$$\Gamma \models \langle \sigma_C \rangle \varphi \text{ iff } \exists \omega \in V(C) \text{ such that } \Gamma, \omega \models \varphi$$

$$\Gamma \models \neg\varphi \text{ iff not } \Gamma \models \varphi$$

$$\Gamma \models \varphi \vee \psi \text{ iff } \Gamma \models \varphi \text{ or } \Gamma \models \psi$$

To simplify the text that follows, we abuse notation somewhat, and write ω for both an outcome (a semantic construct) and the corresponding symbol σ_ω in the language (a syntactic construct). Similarly, we will write i instead of σ_i for agents in the language, and C instead of σ_C for coalitions. So, we will just write $\langle C \rangle \omega$ for $\langle \sigma_C \rangle \sigma_\omega$, although the reader should be aware of the distinction between our object language $\mathcal{L}_c$ and the objects that live in the semantics: outcomes, agents and their preferences, and coalitions.

We will use the usual derived propositional connectives; $\varphi \wedge \psi$ for $\neg(\neg\varphi \vee \neg\psi)$, $\varphi \rightarrow \psi$ for $\neg\varphi \vee \psi$ and $\varphi \leftrightarrow \psi$ for $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$, as well as $\varphi \nabla \psi$ for $(\varphi \vee \psi) \wedge \neg(\varphi \wedge \psi)$ (exclusive or) and $[C]\varphi$ for $\neg\langle C \rangle \neg\varphi$. We also write $(\omega_1 \succ_i \omega_2)$ to abbreviate $((\omega_1 \succeq_i \omega_2) \wedge \neg(\omega_2 \succeq_i \omega_1)$ and $(\omega_1 =_i \omega_2))$ to abbreviate $((\omega_1 \succeq_i \omega_2) \wedge (\omega_2 \succeq_i \omega_1))$.

Note that $\langle C \rangle \top$ iff C can at least bring about something: $V(C) \neq \emptyset$. $[C]\varphi$ means $\neg\langle C \rangle \neg\varphi$, i.e., every choice of C must involve φ. As an example, suppose $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4\}$ and $V(C) = \{\omega_1, \omega_2\}$. Then:

$$\langle C \rangle \omega_1 \wedge \langle C \rangle (\omega_1 \vee \omega_3) \wedge \neg\langle C \rangle \omega_3 \wedge [C](\omega_1 \vee \omega_2) \wedge \neg[C](\omega_1 \vee \omega_3)$$

Note that if $\omega_1 \neq \omega_2$, then we can have $\langle C \rangle \omega_1 \wedge \langle C \rangle \omega_2$, but the formula $\langle C \rangle (\omega_1 \wedge \omega_2)$ can never be true.

A conjunction $\bigwedge_{\delta \in \emptyset} \varphi$ is, by convention, equal to $\top$, like $\bigvee_{\delta \in \emptyset} \varphi$ equals $\bot$, so that, indeed, we get $\Gamma \models [C]\bot$ iff $\Gamma \models \neg\langle C \rangle \top$ iff

$V(C) = \emptyset$. Let us, for any coalition C and set of outcome symbols Δ, suggestively write $\langle [C] \rangle \Delta$ for $\bigwedge_{\delta \in \Delta} \langle C \rangle \delta \wedge [C] \bigvee_{\delta \in \Delta} \delta$.[1] A formula of this form is said to *fully describe C's choices*. It is easy to see that we have the following. Let $\Delta \subseteq \Omega$.

$$\Gamma \models \langle [C] \rangle \Delta \quad \text{iff} \quad V(C) = \Delta$$

Exclusive disjunctions $\varphi \nabla \psi$ play an important role in our proofs. Note that the negation $\neg(\varphi \nabla \psi)$ is the same as $(\neg\varphi \wedge \neg\psi) \vee (\varphi \wedge \psi)$. Moreover, if Φ is a set of formulas, then we define $\nabla_{\varphi \in \Phi} \varphi$ to be true iff exactly one of the φ's is true. Formally: for any set $\Phi = \{\varphi_1, \ldots \varphi_k\}$,[2]

$$\nabla_{\varphi \in \Phi} \varphi \equiv (\bigvee_{i \leq k} \varphi_i \wedge \bigwedge_{j \neq i, j \leq k} \neg \varphi_j)$$

Note that $\Gamma \models [C](\omega_i \vee \omega_j) \leftrightarrow [C](\omega_i \nabla \omega_j)$ when $i \neq j$: using the definition of $[C]$ and contraposition this is the same as $\Gamma \models \langle C \rangle \neg(\omega_i \nabla \omega_j) \leftrightarrow \langle C \rangle \neg(\omega_i \vee \omega_j)$. Now, syntactically, $\langle C \rangle \neg(\omega_i \nabla \omega_j)$ is equivalent to $\langle C \rangle((\omega_i \wedge \omega_j) \vee \neg(\omega_i \vee \omega_j))$. But, inspecting the truth-definition of $\langle C \rangle$, this is again equivalent to $\langle C \rangle \neg(\omega_i \vee \omega_j)$ since the $\mathcal{L}_o$ formula $\omega_i \wedge \omega_j$ is never true.

So, which properties of a coalitional game can be expressed with our cooperation language? The answer, given by the following theorem, is "all".

THEOREM 1. *The logic* CGL *is expressively complete with respect to coalitional games. That is, for any two coalitional games Γ_1, Γ_2 such that $\Gamma_1 \neq \Gamma_2$, there exists a* CGL *formula ζ such that $\Gamma_1 \models \zeta$ and $\Gamma_2 \not\models \zeta$.*

PROOF. *Our proof is constructive. Given a game Γ, we define a formula ζ_Γ that* completely characterises Γ. *ζ_Γ is constructed from two conjuncts, Π_Γ, which characterises the preference relations of Γ, and Ξ_Γ, which characterises the cooperative properties of Γ. Let $\mathcal{C} = 2^N \setminus \emptyset$ collect all non-empty coalitions from N.*

$$\begin{aligned}
\zeta_\Gamma &\equiv \Pi_\Gamma \wedge \Xi_\Gamma \\
\Pi_\Gamma &\equiv \bigwedge_{i \in N} (\bigwedge_{\substack{\omega, \omega' \in \Omega \\ \omega \sqsupseteq_i \omega'}} (\omega \succeq_i \omega') \wedge \bigwedge_{\substack{\omega, \omega' \in \Omega \\ \omega \not\sqsupseteq_i \omega'}} \neg(\omega \succeq_i \omega')) \\
\Xi_\Gamma &\equiv \Big(\bigwedge_{C \in \mathcal{C}} (\bigwedge_{\omega \in V(C)} \langle C \rangle \omega) \wedge [C](\bigvee_{\omega \in V(C)} \omega)\Big)
\end{aligned}$$

By construction, for any Γ_1, we have $\Gamma_1 \models \zeta_{\Gamma_1}$. Moreover, for any coalitional game $\Gamma_2 \neq \Gamma_1$, we have that $\Gamma_2 \not\models \zeta_{\Gamma_1}$. □

From now on, $\mathcal{C}$ is the set of non-empty coalitions. Given a formula φ, then: let $coal(\varphi)$ denote the set of coalitions named in φ; let $ag(\varphi)$ denote the set of agents named in cooperation expressions in φ; and let $out(\varphi)$ be the set of outcomes named in φ.

3. AXIOMS AND COMPLETENESS

We now present an axiomatic system for the language $\mathcal{L}_c$, and prove its soundness and completeness with respect to the class of all coalitional games without transferable payoff.

Table 1 summarizes the axioms and rules of our logic CGL. Formally, CGL is the set of all $\mathcal{L}_c$-formulas derivable under $\vdash$. In the axioms, $\vdash_c$ denotes derivability of classical logic, and $\varphi_o \in \mathcal{L}_o, \varphi, \psi \in \mathcal{L}_c$. The axiom *Taut* and rule *MP* guarantee that we extend classical logic. On top of that, the axiom K and rule *Nec* determine $[C]$ to be a normal necessity operator. Then, *Lin*, *Ref*, and *Trans* determine the preference of each i to be linear, reflexive and transitive, respectively. The only specific cooperation axiom, *Func*, says that whatever a coalition in the end will chose, it must be a unique alternative from Ω.

Taut	$\vdash \varphi$ φ an instance of a propositional tautology
Lin	$\vdash (\omega_1 \succeq_i \omega_2) \vee (\omega_2 \succeq_i \omega_1)$
Ref	$\vdash (\omega \succeq_i \omega)$
Trans	$\vdash (\omega_1 \succeq_i \omega_2) \wedge (\omega_2 \succeq_i \omega_3) \rightarrow (\omega_1 \succeq_i \omega_3)$
K	$\vdash [C](\varphi \rightarrow \psi) \rightarrow (([C]\varphi) \rightarrow ([C]\psi))$
Func	$\vdash [C](\nabla_{\omega \in \Omega} \omega)$
Nec	$\vdash_c \varphi_o \Rightarrow \vdash [C]\varphi_o$
MP	$\vdash \varphi, \vdash \varphi \rightarrow \psi \Rightarrow \vdash \psi$

Table 1: The logic CGL**.**

The following lemma tells us that in the scope of modal operators, disjunctions over *different* outcomes behave the same as exclusive disjunctions over outcomes. Note that this is in general not true for arbitrary disjunctions: $\langle C \rangle(\omega_1 \vee \omega_1)$ is not the same as $\langle C \rangle(\omega_1 \nabla \omega_1)$, the latter is equivalent to $\langle C \rangle \bot$.

LEMMA 1. *Let $\emptyset \neq C \subseteq N$ and $\Delta \subseteq \Omega$.*

1. *The following are equivalent, in* CGL*: $(i) \langle C \rangle \top$, $(ii) \langle C \rangle \bigvee_{\omega \in \Omega} \omega$, *and* $(iii) \langle C \rangle \nabla_{\omega \in \Omega} \omega$*
2. *In the scope of $\langle C \rangle$ and $[C]$ when exchanging arbitrary occurrences of $\nabla_{\omega \in \Delta} \omega$ with that of $\bigvee_{\omega \in \Delta} \omega$ in a formula φ, the result is equivalent to φ.*
3. $\vdash \bigwedge_{\omega \in \Delta} \neg \langle C \rangle \omega \rightarrow (\langle C \rangle \top \rightarrow \bigvee_{\omega \in \Omega \setminus \Delta} \langle C \rangle \omega)$
4. $\vdash (\bigwedge_{\delta \in \Delta} \neg \langle C \rangle \delta) \leftrightarrow [C] \bigvee_{\delta' \in \Omega \setminus \Delta} \delta'$

PROOF.

1. Since $\nabla_{\omega \in \Omega} \omega \Rightarrow \bigvee_{\omega \in \Omega} \omega \Rightarrow \top$, and the $\langle C \rangle$ is a normal diamond operator, we have $(iii) \Rightarrow (ii) \Rightarrow (i)$. By *Func*, we have $(i) \Rightarrow (iii)$.
2. With induction over φ. The only interesting case $\varphi = \langle C \rangle \psi$ follows from the previous item.
3. From axiom *Func* follows $\vdash \langle C \rangle \top \rightarrow \langle C \rangle \nabla_{\omega \in \Omega} \omega$. By item 1, we have $\vdash \langle C \rangle \top \rightarrow \langle C \rangle \bigvee_{\omega \in \Omega} \omega$. Since $\langle C \rangle$ is a normal diamond, we have $\vdash \langle C \rangle \top \rightarrow \bigvee_{\omega \in \Omega} \langle C \rangle \omega$. Applying *Taut* to this gives the desired property.
4. Follows directly from *Func* and some modal reasoning.

□

LEMMA 2.

1. *Let $C \neq \emptyset$ be a coalition. Then* $\vdash \bigvee_{\Delta \subseteq \Omega} \langle [C] \rangle \Delta$
2. *Let $C \neq \emptyset$ be a coalition. Then* $\vdash \nabla_{\Delta \subseteq \Omega} \langle [C] \rangle \Delta$
3. $\vdash \bigwedge_{C \in \mathcal{C}} \bigvee_{\Delta \subseteq \Omega} \langle [C] \rangle \Delta$
4. $\vdash \bigwedge_{C \in \mathcal{C}} \bigtriangleup_{\Delta \subseteq \Omega} \langle [C] \rangle \Delta$
5. $\vdash \bigvee_{\Delta_C \subseteq \Omega, C \in \mathcal{C}} \bigwedge_{C \in \mathcal{C}} \langle [C] \rangle \Delta_C$

PROOF.

[1] Note that the $\langle [C] \rangle$ modality plays the same role wrt. $[C]$ as Levesque's [9] *only knowing* operator plays wrt. the traditional belief operator.

[2] Note that $\nabla_{p,q,r}$ is not the same as $p \nabla (q \nabla r)$!

1. Note that $\langle C\rangle\omega$ is just an atom in the coalitional language. Let $M = \{1, \ldots, n\}$. Then, even by propositional reasoning, $\vdash \bigvee_{I\cup J=M, I\cap J=\emptyset}(\bigwedge_{i\in I}\langle C\rangle\omega_i \wedge \bigwedge_{j\in J}\neg\langle C\rangle\omega_j)$. Now, take a fixed I and J with $I \cap J = \emptyset$ and $I \cup J = M$. If $I = \emptyset$, then $\bigwedge_{i\in I}\langle C\rangle\omega_i \wedge \bigwedge_{j\in J}\neg\langle C\rangle\omega_j$ equals $\bigwedge_{\omega\in\Omega}\neg\langle C\rangle\omega$, which is equivalent to $\langle[C]\rangle\emptyset$. With Lemma 1 item 4, we have, for fixed I and J, each disjunct in this is equivalent to $\langle[C]\rangle\Delta_I$, with $\Delta_I = \{\omega_i \mid i \in I\}$.
2. Note that, using the notation of the previous item, we even have $\vdash \bigwedge_{I\cup J=M, I\cap J=\emptyset}(\bigwedge_{i\in I}\langle C\rangle\omega_i \wedge \bigwedge_{j\in J}\neg\langle C\rangle\omega_j)$. From this the statement follows directly.
3. This is immediate from item 1: if for an arbitrary C we have $\vdash \langle[C]\rangle\varphi_C$, then also $\vdash \bigwedge_{C\subseteq Z}\langle[C]\rangle\varphi_C$, for any $Z \in \mathcal{C}$.
4. Follows from item 2 in the same way as 3 follows from 1.
5. This follows immediately from item 1 and propositional reasoning: note that for every two coalitions C_1 and C_2 we derive $\vdash \bigvee_{\Delta_1\subseteq\Omega}\langle[C_1]\rangle\Delta_1 \wedge \bigvee_{\Delta_2\subseteq\Omega}\langle[C_2]\rangle\Delta_2$, and use $(p \vee q) \wedge r \equiv (p \wedge r) \vee (q \wedge r)$.

□

DEFINITION 1.

1. *For any agent i, we say that a formula is a preference literal for i if it is either $\omega \succeq_i \omega'$ or $\neg(\omega \succeq_i \omega')$, for some ω and ω'. We say that π_i fully describes i's preferences, if*

$$\pi_i \text{ is of the form } \bigwedge_{\omega,\omega'\in\Omega}(\neg)(\omega \succeq_i \omega')$$

We then say that $\pi \in PossPref(i)$.
2. *Given that we have m agents, a conjunction $\Pi = (\pi_1 \wedge \cdots \wedge \pi_m)$, (where each π_i fully describes i's preferences) is said to fully describe the preferences of all the agents. Similarly for coalitions.*
3. *Recall that $\langle[C]\rangle\Delta$, where C is a coalition and Δ is a set of (atoms for) outcomes $\omega_1, \ldots, \omega_u$, is said to fully describe C's choices. Now let $\mathcal{C} = 2^N \setminus \emptyset$, and let, for each $C \in \mathcal{C}$, Δ_C be a set of outcomes. Then $\bigwedge_{C\in\mathcal{C}}\langle[C]\rangle\Delta_C$ is said to fully describe all of N's choices. (Similarly for subsets of N.) We often will denote such a full description Ξ.*

LEMMA 3.

1. *Let $\Pi_1, \Pi_2, \ldots, \Pi_d$ be all full descriptions of N's preferences. Then: $\vdash \bigvee_{k\leq d}\Pi_k$*
2. *Let $\Pi_1, \Pi_2, \ldots, \Pi_d$ be all full descriptions of N's preferences. Moreover, let $\mathcal{C} = 2^N \setminus \emptyset$ and let*

$$(\bigwedge_{C\in\mathcal{C}}\langle[C]\rangle\Delta_C)_1, \ldots, (\bigwedge_{C\in\mathcal{C}}\langle[C]\rangle\Delta_C)_z$$

enumerate all possible full descriptions of all choices of all coalitions (note that $z = (2^n)^m$). Then:

$$\vdash \bigvee_{k\leq d, t\leq z}(\Pi_k \wedge \bigwedge_{C\in\mathcal{C}}\langle[C]\rangle\Delta_C)_t$$

THEOREM 2. *Let φ be a formula of the cooperation language.*

1. *Let N be the set of agents, Ω the set of outcomes, and let $\mathcal{C} = 2^N \setminus \emptyset$. Then φ is equivalent to a formula of the form*

$$\bigvee_{k\in K, t\in T}(\Pi_k \wedge (\bigwedge_{C\in\mathcal{C}}\langle[C]\rangle\Delta_C)_t)$$

where each $\Pi_k = (\pi_1 \wedge \cdots \wedge \pi_m)$ fully describes N's preferences, i.e., each π_i fully describes i's preferences, and each $(\bigwedge_{C\in\mathcal{C}}\langle[C]\rangle\Delta_C)_t$ describes fully what N can choose.

2. *The same holds if we take $\Omega = out(\varphi), N = ag(\varphi)$ and we let C range over all $coal(\varphi)$.*

Complex as it may appear, our normal form is nothing more than an enumeration of possible full preferences combined with full descriptions of choices. The range of these possibilities is determined by the index sets K and T, which act as a kind of constraints: the smaller those index sets, the smaller the possible models for the formula. As a reading guide, note that

$$\bigvee_{k\in\{1,2\}, t\in\{a,b\}}(A_k \wedge B_t)$$

equals $(A_1 \wedge B_a) \vee (A_1 \wedge B_b) \vee (A_2 \wedge B_a) \vee (A_2 \wedge B_b)$.

PROOF. Note that the theorem is semantically obvious, the point is that we should be able to *syntactically prove* it from the sole axioms. This is done by induction over φ.

If $\varphi = \omega_1 \succeq_k \omega_2$. Let us say that a π_k is k-compatible with $(\omega_1 \succeq_k \omega_2)$ if the latter occurs as a conjunct in π_k. We then write $(\omega_1 \succeq_k \omega_2) \in \pi_k$ Then, φ is (in propositional logic) equivalent to

$$\bigvee_{\pi_i\in PossPref(i)(i\neq k), (\omega_1\succeq_k\omega_2)\in\pi_k}(\pi_1 \wedge \cdots \wedge \pi_k \wedge \cdots \wedge \pi_m)$$

This if of the form $\bigvee_{k\in K}\Pi_k$. Since by Lemma 2, item 5 we have that $\top$ is equivalent to $\bigvee_{\Delta_C\subseteq\Omega, C\in\mathcal{C}}\bigwedge_{C\in\mathcal{C}}\langle[C]\rangle\Delta_C$ we see that φ is equivalent to

$$\bigvee_{k\in K}\Pi_k \wedge \bigvee_{\Delta_C\subseteq\Omega, C\in\mathcal{C}}\bigwedge_{C\in\mathcal{C}}\langle[C]\rangle\Delta_C$$

which, by propositional reasoning, is equivalent to

$$\bigvee_{k\in K, \Delta_C\subseteq\Omega, C\in\mathcal{C}}\Pi_k \wedge \bigwedge_{C\in\mathcal{C}}\langle[C]\rangle\Delta_C$$

Suppose $\varphi = \langle E\rangle\varphi_0$. Formula φ_0 regards outcomes, and is equivalent to a disjunction $\bigvee\alpha$ where each α is of the form

$$\alpha = ((\neg)\omega_1 \wedge (\neg)\omega_2 \wedge \cdots \wedge (\neg)\omega_n)$$

Using rule Nec and axiom K, we derive $\vdash \langle E\rangle\varphi_0 \leftrightarrow \langle E\rangle\bigvee\alpha$. Using that $\langle\cdot\rangle$ is a diamond, we then obtain that $\vdash \langle E\rangle\varphi_0 \leftrightarrow \bigvee\langle E\rangle\alpha$. Now we use axiom $Func$ to get rid of every α that contains more than one positive literal ω_i: let β range over all the those α's with at most one positive literal. Then $\vdash \langle E\rangle\varphi_0 \leftrightarrow \bigvee\langle E\rangle\beta$. Now, again in propositional logic, note that every disjunction $\bigvee_{i\in M}\psi_i$ is equivalent to

$$\bigvee_{I\cup J=M, I\cap J=\emptyset}(\bigwedge_{i\in I}\psi_i \wedge \bigwedge_{j\in J}\neg\psi_j).$$

In our case, letting the β's range over $\beta_1, \ldots, \beta_M$,

$$\vdash \langle E\rangle\varphi_0 \leftrightarrow \bigvee_{I\cup J=M, I\cap J=\emptyset}(\bigwedge_{i\in I}\langle E\rangle\beta_i \wedge \bigwedge_{j\in J}\neg\langle E\rangle\beta_j)$$

But, for every fixed $I, J \subseteq M$, $(\bigwedge_{i\in I}\langle E\rangle\beta_i \wedge \bigwedge_{j\in J}\neg\langle E\rangle\beta_j)$ is equivalent to $\langle[E]\rangle\Delta_I$, with $\Delta_I = \{\omega_i \mid i \in I\}$. Hence, we find that $\vdash \langle E\rangle\varphi_0 \leftrightarrow \bigvee_{I\subseteq M}\langle[E]\rangle\Delta_I$. This only limits the abilities of C, and not those of the others: using Lemma 2 item 5 once again:

$$\bigvee_{I\subseteq M}\langle[E]\rangle\Delta_I \leftrightarrow \bigvee_{I\subseteq M, D\neq E, \Delta_D\subseteq\Omega}\langle[E]\rangle\Delta_I \wedge \bigwedge_{D\in\mathcal{C}}\langle[D]\rangle\Delta_D$$

of which the r.h.s. is of the form $\bigvee_{C\in\mathcal{C}, t\in T}(\langle[C]\rangle\Delta_C)_t$, for some index set T. Since $\top$ is provably equivalent with $\bigvee_{k\in K}\Pi_k$ (where now K gives all possible full preference descriptions), the result follows.

Suppose φ is of the form $\varphi_1 \vee \varphi_2$. We can assume that $\varphi_i \equiv \bigvee_{k_i \in K_i, t_i \in T_i}(\Pi_{k_i} \wedge \Xi_{t_i})$, then

$$\varphi \equiv \bigvee_{k \in K_1 \cup K_2, t \in T_1 \cup T_2} (\Pi_k \wedge \Xi_t)$$

which is of the required form.

Let $\varphi = \neg\varphi_1$. Then $\varphi_1 = \bigvee_{k \in K, t \in T}(\Pi_k \wedge \Xi_t)$, for some index sets K and T. From Lemma 3, item 2, we know that $\vdash \bigvee_{k \leq d, t \leq z}(\Pi_k \wedge \Xi_t)$. In words: we know that a big disjunction is valid, but also that φ excludes some of them. Then, using propositional reasoning again, we obtain:

$$\varphi \equiv \bigvee_{k \leq d, k \notin K, t \leq z, t \notin T} (\Pi_k \wedge \Xi_t)$$

□

THEOREM 3 (COMPLETENESS). *We have, for all $\varphi \in \mathcal{L}_c$ over N and Ω: $\models \varphi \Rightarrow \vdash \varphi$*

PROOF. Suppose $\nvdash \varphi$, i.e., $\neg\varphi$ is consistent. We know that

$$\vdash \neg\varphi \leftrightarrow \bigvee_{k \in K, t \in T} (\Pi_k \wedge (\bigwedge_{C \in \mathcal{C}} \langle [C] \rangle \Delta_C)_t)$$

for some index sets K and T. Call the righthandside of this equivalence φ'. Let X be a maximal consistent set around φ'. By virtue of maximal consistent sets, we know that for some k and t, $\varphi'' = (\Pi_k \wedge (\bigwedge_{C \in \mathcal{C}} \langle [C] \rangle \Delta_C)_t) \in X$. But now we can read off the game $\Gamma = \langle N, \Omega, V, \sqsupseteq_1, \ldots, \sqsupseteq_m \rangle$ from φ'' immediately:

1. N and Ω are already given;
2. Let $V(C) = \Delta$, where $\langle [C] \rangle \Delta$ is part of φ'
3. every $\sqsupseteq_i$ relation is immediately read off from the component π_i for Π_k in φ''.

Now, it is easy to see, for every subformula ψ of φ'':

$$\Gamma \models \psi \Leftrightarrow \psi \in X$$

□

4. MODEL CHECKING & SATISFIABILITY

It is trivial to see that the *model checking problem* for CGL (i.e., the problem of determining, for any given Γ and φ, whether or not $\Gamma \models \varphi$ [4]) may be solved in deterministic polynomial time: an obvious recursive algorithm for this problem can be directly extracted from the semantic rules of the language.

The *satisfiability problem* is the problem of checking whether or not, for any given φ there exists a game Γ such that $\Gamma \models \varphi$. For most modal logics, the corresponding satisfiability problem has a trivial NP-hard lower bound, since such logics subsume propositional logic, for which satisfiability is the defining NP-complete problem [3, p.374]. However, our logic is specialised for reasoning about coalitional games, and it is not so obvious that it subsumes propositional logic, since we do not have primitive propositions. We must therefore prove NP-hardness from first principles.

For the proof, we need a few additional constructions. A *partial coalitional game* is a structure $\langle N, \Omega, V, \sqsupseteq_1, \ldots, \sqsupseteq_m \rangle$ where all the components are as in regular coalitional games, except that V is a *partial* function, i.e., it is not required to be defined for every possible coalition. Given a partial game $\Gamma = \langle N, \Omega, V, \sqsupseteq_1, \ldots, \sqsupseteq_m \rangle$, we can use the semantic rules for CGL to interpret some formulae (although because V is not defined for all coalitions, we cannot necessarily interpret all formulae over N, Ω). Where Γ is a partial game and φ is a formula, let us write $\Gamma \models_p \varphi$ to mean that (i) it is possible to evaluate φ with respect to Γ, and (ii) φ is true under this evaluation. Now, we can prove the following.

LEMMA 4. *A CGL formula φ is satisfiable iff there exists a partial game $\Gamma = \langle N, \Omega, V, \sqsupseteq_1, \ldots, \sqsupseteq_m \rangle$ such that:*

1. *$N = ag(\varphi)$,*
2. *$|\Omega| = |out(\varphi)| + 1$ and $out(\varphi) \subseteq \Omega$;*
3. *dom $V = coal(\varphi)$, and*
4. *$\Gamma \models_p \varphi$.*

PROOF. *The right-to-left direction is obvious, so consider the left-to-right direction, and let $\Gamma = \langle N, \Omega, V, \sqsupseteq_1, \ldots, \sqsupseteq_m \rangle$ be a game such that $\Gamma \models \varphi$. Let $A = \Omega \setminus out(\varphi)$, i.e., A is the set of outcomes in Γ not named in φ. Let ω^* be an outcome such that $\omega^* \notin \Omega$, and define a partial game $\Gamma^* = \langle N^*, \Omega^*, V^*, \sqsupseteq_1^* , \ldots, \sqsupseteq_o^* \rangle$ as follows:*

- *$N^* = ag(\varphi)$;*
- *$\Omega^* = out(\varphi) \cup \{\omega^*\}$;*
- *The relation $\sqsupseteq_i^*$ is obtained by first restricting $\sqsupseteq_i$ to $out(\varphi)$, and then defining $\omega^* \sqsupseteq_i \omega$ for all $\omega \in out(\varphi)$;*
- *V^* is the partial function such that V^* is only defined for coalitions named in φ (i.e., $C \in$ dom V^* iff $C \in coal(\varphi)$);*
- *$V^*(C) = \begin{cases} V(C) & \text{if } V(C) \subseteq out(\varphi) \\ (V(C) \setminus A) \cup \{\omega^*\} & \text{otherwise.} \end{cases}$*

Notice that Γ^* satisfies conditions (1)–(3) of the lemma. We now prove that Γ^* satisfies condition (4). More precisely, we show that for all sub-formulae ψ of φ: $\Gamma \models \psi$ iff $\Gamma^* \models \psi$. The inductive base is where $\varphi = (\omega_1 \succeq_i \omega_2)$, and is obvious, since $i \in ag(\varphi)$ and $\{\omega_1, \omega_2\} \subseteq out(\varphi)$, and hence $\omega_1 \sqsupseteq_i^* \omega_2$ iff $\omega_1 \sqsupseteq_i \omega_2$.

For the inductive assumption, assume the result is proved for all sub-formulae; in the inductive step, the significant case is where $\varphi = \langle C \rangle \psi$. If $\Gamma \models \langle C \rangle \psi$ then $\exists \omega \in V(C)$ such that $\Gamma, \omega \models \psi$. There are two possibilities: either $\omega \in out(\varphi)$ (in which case $V^*(C) = V(C)$, and the result is obvious), or else $\omega \notin out(\varphi)$. In the latter case, $V^*(C) = (V(C) \setminus A) \cup \{\omega^*\}$; we claim that $\Gamma^*, \omega^* \models \psi$. To see this, assume w.l.o.g. that ψ is in Conjunctive Normal Form. Now, since $\omega \notin out(\varphi)$, then no positive literals can be satisfied by Γ, ω: only negative literals. But such literals must also be satisfied by Γ^*, ω^*, and so $\Gamma^*, \omega^* \models \psi$.

The case for $\Gamma \not\models \langle C \rangle \psi$ implies $\Gamma^* \not\models \langle C \rangle \psi$ is similar. □

Given this, we can prove:

THEOREM 4. *The satisfiability problem for CGL formulae is NP-complete, even for CGL formulae φ such that $|ag(\varphi)| = 1$.*

PROOF. For membership of NP, we know that φ is satisfiable iff it has a "certificate" for this in the form of a partial game Γ as in Lemma 4. This partial game is of size linear in the size of the formula φ. Since we can check whether $\Gamma \models_p \varphi$ in polynomial time, we conclude that CGL satisfiability is in NP.

For NP-hardness, we reduce SAT, the problem of determining whether a formula $\varphi(x_1, \ldots, x_k)$ of propositional logic, over Boolean variables $x_1, \ldots, x_k$, has a satisfying assignment [12]. The basic idea is to map variables x_i to outcomes ω_i, to introduce an additional outcome $\omega_\perp$ to correspond to the truth value "false", so that $(\omega_x \succ_1 \omega_\perp)$ will mean "x takes the value 'true' ". Formally, let $\varphi^\#$ denote the CGL formula obtained from the propositional logic formula φ by systematically replacing every Boolean variable p by the corresponding CGL expression $(\omega_p \succ_1 \omega_\perp)$. Now, we claim that $\varphi^\#$ is CGL satisfiable iff the input SAT instance φ is a satisfiable formula of propositional logic.

For the $\Rightarrow$ direction, assume $\varphi^\#$ is CGL satisfiable, and consider the associated preference relation $\sqsupseteq_1$ in any Γ such that $\Gamma \models \varphi$. From this relation, extract a valuation ξ for the variables $x_1, \ldots, x_k$

as follows: each variable x_i is true under ξ if $\omega_{x_i} \sqsupset_1 \omega_\perp$, and false otherwise. The interpretation ξ is consistent, since we cannot have both $\omega_{x_i} \sqsupset_1 \omega_\perp$ and $\omega_\perp \sqsupset_1 \omega_{x_i}$. The interpretation ξ satisfies φ by a trivial induction on the structure of φ.

For $\Leftarrow$, assume φ is a satisfiable formula of propositional logic, and let ξ be a valuation that satisfies φ. Then we can reconstruct a game Γ_ξ such that $\Gamma_\xi \models \varphi^\#$, as follows. Γ_ξ contains a single agent, (agent 1), and an outcome ω_{x_i} for each variable x_i appearing in φ. We also define an additional outcome $\omega_\perp$. The preference relation $\sqsupseteq_i$ is then defined as follows:

- For each Boolean variable p such that p is true under ξ, define $\omega_p \sqsupset_i \omega_\perp$.
- For each pair of Boolean variables p_1, p_2 such that p_1 and p_2 are both true or both false under ξ, define $\omega_{p_1} =_i \omega_{p_2}$.
- For each Boolean variable p such that p is false under ξ, define $\omega_\perp \sqsupseteq_i \omega_p$.
- For each pair of Boolean variables p_1, p_2 such that p_1 is true (respectively, false) and p_2 is false (respectively, true) under ξ, define $\omega_{p_1} \sqsupset_i \omega_{p_2}$ (respectively, $\omega_{p_2} \sqsupset_i \omega_{p_1}$).

An induction on $\varphi^\#$ proves that $\Gamma_\xi \models \varphi^\#$. $\square$

5. CHARACTERISING COALITIONAL GAMES

We characterise three solution concepts from the theory of coalitional games, viz. *the core* [7], *stable sets* [10] and *the bargaining set* [2], in CGL. We use the formulations of these solution concepts in [11]; there the two latter solution concepts are however defined only for games with real numbered payoffs and transferable utility and below we translate the definitions to the more general games with preference relations over general outcomes and non-transferable utility.

Henceforth, a *C-feasible outcome* is an outcome which can be chosen by the coalition C and a *feasible outcome* is a N-feasible outcome where N is the set of players in the game. We start by looking at the core, which is a, possibly empty, set of outcomes.

DEFINITION 2 (CORE). *The core of a coalitional game is the feasible outcomes ω for which there is no coalition C with a C-feasible outcome ω' such that $\omega' \succ_i \omega$ for all $i \in C$.*

We write $CM(\omega)$ to mean that ω is in the core.

$$CM(\omega) \equiv \langle N \rangle \omega \wedge \neg \left[\bigvee_{C \subseteq N} \bigvee_{\omega' \in \Omega} (\langle C \rangle \omega') \wedge \bigwedge_{i \in C} (\omega' \succ_i \omega) \right]$$

CNE will then mean that the core is non-empty:

$$CNE \equiv \bigvee_{\omega \in \Omega} CM(\omega)$$

THEOREM 5. *The core of Γ is non-empty iff $\Gamma \models CNE$.*

A *stable set* is a set of outcomes. A coalitional game may have several stable sets, but must not necessarily have any. We characterize stable sets in terms of *imputations* and *objections*. An imputation is a feasible outcome that for each agent i is as least as good as any outcome the singleton coalition $\{i\}$ can choose on his own. The CGL formula $IMP(\omega)$ is true whenever ω is an imputation:

$$IMP(\omega) \equiv \langle N \rangle \omega \wedge \bigwedge_{\omega' \in \Omega} \bigwedge_{i \in N} (\langle \{i\} \rangle \omega' \rightarrow \omega \succeq_i \omega')$$

An imputation ω is a C-objection to an imputation ω' if every agent in C prefers ω over ω' and the coalition C can choose an outcome which for every agent in C is as least as good as ω. ω is an objection to ω' if ω is a C-objection to ω' for some coalition C. Next, $OBJ(\omega, \omega', C)$ expresses that outcome ω is an C-objection to outcome ω', when both ω and ω' are imputations:

$$OBJ(\omega, \omega', C) \equiv (\bigwedge_{i \in C} \omega \succ_i \omega') \wedge \bigvee_{\omega'' \in \Omega} (\langle C \rangle \omega'' \wedge \bigwedge_{i \in C} \omega'' \succeq_i \omega)$$

DEFINITION 3 (STABLE SET). *A set of imputations Y is a stable set if it satisfies:*

Internal stability *If $\omega \in Y$, there is no objection to ω in Y.*

External stability *If $\omega \notin Y$, there is an objection to ω in Y.*

Given a set of outcomes $Y \subseteq \Omega$, the CGL formula $STABLE(Y)$ expresses the fact that Y is a stable set:

$$\begin{array}{l} STABLE(Y) \equiv \\ \quad \bigwedge_{\omega \in Y} IMP(\omega) \\ \quad \wedge \Big(\bigwedge_{\omega \in Y} \bigwedge_{C \subseteq N} \bigwedge_{\omega' \in Y} \neg OBJ(\omega', \omega, C) \Big) \\ \quad \wedge \Big(\bigwedge_{\omega \in \Omega \setminus Y} IMP(\omega) \rightarrow \Big(\bigvee_{C \subseteq N} \bigvee_{\omega' \in Y} OBJ(\omega', \omega, C) \Big) \Big) \end{array}$$

THEOREM 6. *Y is a stable set of Γ iff $\Gamma \models STABLE(Y)$.*

PROOF. Given a coalitional game, let $\mathcal{I}$ denote the set of all imputations. First, we argue that $IMP(\omega)$ and $OBJ(\omega, \omega', C)$ have the correct meaning. Every $\omega \in Y$ is an imputation iff $\omega \in V(N)$ (feasibility) and $\omega \succeq_i \omega'$ for all i and $\omega' \in V(\{i\})$ which is exactly when $IMP(\omega)$ holds. If $\omega, \omega' \in \mathcal{I}$ and $C \subseteq N$, ω is a C-objection to ω' iff $\omega \succ_i \omega'$ for every $i \in C$ and there is a $\omega'' \in V(C)$ such that $\omega'' \succeq_i \omega$ for every $i \in C$, which is exactly when $OBJ(\omega, \omega', C)$ holds.

For the main proof, let $Y \subseteq \Omega$. If there is an ω in Y which is not an imputation, Y is not a stable set and $IMP(\omega)$ is not true and we are done, so assume that Y is a set of imputations. Let

$$\hat{Y} = \{\omega \in \mathcal{I} : \text{ there is no objection to } \omega \text{ in } Y\}$$

It is easy to see that if Y is a stable set iff $Y = \hat{Y}$. We argue that the second and third main conjuncts of the formula $STABLE(Y)$ is true whenever $Y \subseteq \hat{Y}$ and $\hat{Y} \subseteq Y$ hold, respectively, and the theorem follows (the first conjunct is true under the assumption that Y are imputations). The second conjunct is true exactly when for every member of Y there is no C-objection to ω in Y for any C, which is exactly when $Y \subseteq \hat{Y}$ holds. The third conjunct is true iff every imputation which is not in Y has an objection in Y or, contrapositively, that every imputation which does not have an objection in Y is included in Y which is the same as $\hat{Y} \subseteq Y$. $\square$

Existence of a stable set can then be expressed as:

$$ES \equiv \bigvee_{Y \subseteq \Omega} STABLE(Y)$$

COROLLARY 1. *Γ has a stable set iff $\Gamma \models ES$.*

Finally, we focus on the notion of a bargaining set of a coalitional game which is, like a stable set, a set of imputations, but, unlike a stable set, is unique and always exists. The bargaining set of a game can be defined in terms of *objections* and *counterobjections*, but the former concept is not the same as in the definition of stable sets. Let ω be an imputation:

Objection: A pair consisting of a coalition C and a C-feasible

outcome ω' is an objection of an agent $i \in C$ against an agent $j \notin C$ to ω if every agent in C prefers ω' over ω.

Counterobjection: A pair consisting of a coalition D and a D-feasible outcome v is a counterobjection to an objection (ω', C) of i against j to ω, if D includes j but not i, every agent in $D \setminus C$ thinks v is as least as good as ω and every agent in $D \cap C$ thinks v is as least as good as ω'.

The CGL formula $OBJB(\omega', C, \omega)$ means that (ω', C) is an objection of any $i \in C$ against any $j \notin C$ to ω.

$$OBJB(\omega', C, \omega) \equiv \langle C \rangle \omega' \wedge \bigwedge_{k \in C} \omega' \succ_k \omega$$

$ECO(\omega', C, i, j, \omega)$ means that there exists a counterobjection to the objection (ω', C) of i against j to ω.

$$ECO(\omega', C, i, j, \omega) \equiv \bigvee_{v \in \Omega} \bigvee_{D' \subseteq N \setminus \{i\}} (\langle D' \cup \{j\} \rangle v \wedge (\Big(\bigwedge_{k \in (D' \cup \{j\}) \setminus C} v \succeq_k \omega\Big) \wedge \Big(\bigwedge_{k \in (D' \cup \{j\}) \cap C} v \succeq_k \omega'\Big)))$$

DEFINITION 4 (BARGAINING SET). *The bargaining set of a coalitional game is the set of all imputations ω such that there exists a counterobjection to every objection of any player i against any player j to ω.*

$INBARG(\omega)$ means that outcome $\omega \in \Omega$ is in the bargaining set:

$$INBARG(\omega) \equiv IMP(\omega) \wedge \bigwedge_{C \subseteq N} \bigwedge_{i \in C} \bigwedge_{j \in N \setminus C} \bigwedge_{\omega' \in \Omega} [OBJB(\omega', C, \omega) \rightarrow ECO(\omega', C, i, j, \omega)]$$

THEOREM 7. *ω is a member of the bargaining set of Γ iff $\Gamma \models INBARG(\omega)$.*

PROOF. It is easy to see that when ω is an imputation, C is a coalition, $i \in C, j \notin C$ and ω' an outcome, $\Gamma \models OBJB(\omega', C, \omega)$ iff (ω', C) is an objection of i against j to ω. To see that there exist a counterobjection to the objection (ω', C) of i against j to ω iff $\Gamma \models ECO(\omega', C, i, j, \omega)$, observe that (v, D) is a counterobjection iff the disjunct given by v and $D' = D \setminus \{j\}$ is true. The theorem follows immediately. □

We can now define $BS(Y)$, $Y \subseteq \Omega$, to express the fact that Y is the bargaining set.

$$BS(Y) = \bigwedge_{\omega \in Y} INBARG(\omega) \wedge \bigwedge_{\omega \in \Omega \setminus Y} \neg INBARG(\omega)$$

COROLLARY 2. *Y is the bargaining set of Γ iff $\Gamma \models BS(Y)$*

COROLLARY 3. *$\neg BS(\emptyset)$ is a theorem of* CGL.

PROOF. For any coalitional game without transferable payoff Γ, the bargaining set is non-empty [6, 15]. By Corollary 2, $\Gamma \models \neg BS(\emptyset)$ for any Γ, and by Theorem 3 $\vdash \neg BS(\emptyset)$. □

6. RELATION TO COALITION LOGIC

As we noted in section 2, it is rather tempting to believe that the outcomes of coalitional games can be interpreted as states, and that the characteristic function can be interpreted as an effectivity function, and that as a consequence Coalition Logic (CL) [14] could be interpreted directly in coalitional games. In this section, we compare the semantics of CL and CGL, and show that in fact there is a fundamental difference between the two approaches.

We first give a very brief review of some of the concepts of CL. A *coalition model* for agents N over a set of atomic propositions Φ_0 is a triple $M = (S, E, \pi)$, where S is a nonempty set of states, $\pi : \Phi_0 \rightarrow 2^S$ an assignment and E gives a function of type $2^N \rightarrow 2^{2^S}$ for each state $s \in S$. It is required that each $E(s)$ is a *playable effectivity function*, satisfying the following conditions: i) $\forall_{C \subseteq N} \emptyset \notin E(s)(C)$, ii) $\forall_{C \subseteq N} S \in E(s)(C)$, iii) for any X, if $S \setminus X \notin E(s)(N \setminus C)$ then $X \in E(s)(C)$ (*N-maximality*), iv) for all $X \subseteq X' \subseteq S$ and all C, if $X \in E(s)(C)$ then $X' \in E(s)(C)$ (*outcome monotonicity*) and v) if $C_1 \cap C_2 = \emptyset$, $X_1 \in E(s)(C_1)$ and $X_2 \in E(s)(C_2)$, then $X_1 \cap X_2 \in E(s)(C_1 \cup C_2)$ (*superadditivity*). Formulae of coalition logic, and their satisfaction in states s of coalition models M, are defined as follows[3]:

$$M, s \models \langle C \rangle \psi \quad \text{iff } \psi^M \in E(s)(C)$$

where $\psi^M = \{s \in S : M, s \models \psi\}$. If we take the set of atomic propositions to be $\Phi_0 = \Omega \cup \{\omega \succeq_i \omega' : \omega, \omega' \in \Omega, i \in N\}$, then we can read every formula in $\mathcal{L}_o \cup \mathcal{L}_c$ as a formula of coalition logic. Thus we can interpret $\mathcal{L}_c$ formulae in both a game Γ and in a pointed coalition model (M, s), and a $\mathcal{L}_o$ formula in both a pointed game (Γ, ω) and in a pointed model (M, s). Coalitional games and coalition models have many similarities. The former have "outcomes" while the latter have "states". An interesting question is: given a coalitional game Γ, does there exist an equivalent coalition model M with states corresponding to the outcomes of Γ, maybe in addition to a designated "initial" state t? Equivalence here means that Γ and the pointed coalition model (M, t) agree on $\mathcal{L}_c$ formulae and (Γ, ω) and (M, ω) agree on $\mathcal{L}_o$ formulae for any outcome ω. We can say that Γ and M then are *outcome-equivalent*.

In other words, a coalitional game Γ and a coalition model M, defined over atomic propositions Φ_0 above and having states $\Omega \cup \{t\}$, are outcome-equivalent iff for any $\varphi_1 \in \mathcal{L}_c$, any $\varphi_0 \in \mathcal{L}_o$ and any $\omega \in \Omega$: we have both (a) $\Gamma \models \varphi_1$ iff $M, t \models \varphi_1$, and (b) $\Gamma, \omega \models \varphi_0$ iff $M, \omega \models \varphi_0$.

A natural question then is: given a game, does there exist an outcome-equivalent coalition model?

The answer, given by the following theorem, is "no", except for for certain special cases of games. The latter is the class of games where $V(C) = \{\omega\}$ for all coalitions $C \neq N$, for some fixed outcome $\omega \in \Omega$. To give them a name, we will call such games *limited games*, since, first, most games are not of this kind and, second, they are not very interesting. The only coalition in a limited game which possibly can select an outcome different from the fixed outcome ω is the grand coalition.

THEOREM 8. *No non-limited coalitional game with more than one player has an outcome-equivalent coalition model.*

PROOF. Let $\Gamma = \langle N, \Omega, V, \sqsupseteq_1, \ldots, \sqsupseteq_m \rangle$ be a coalitional game, and assume that $M = (S, E, \pi)$ with $S = \Omega \cup \{t\}$ is a coalition model outcome-equivalent to Γ. We argue, by using the properties i) – v) of a playable effectivity function given above, that Γ must be limited. First, observe that for any $\mathcal{L}_o$ formula φ and coalition $C \neq \emptyset$, $\Gamma \models \langle C \rangle \varphi$ iff, by (a), $M, t \models \langle C \rangle \varphi$, i.e.

$$\varphi^M \in E(t)(C) \Leftrightarrow \exists_{\omega \in V(C)} \Gamma, \omega \models \varphi \tag{1}$$

for any $\varphi \in \mathcal{L}_0$, $C \subseteq N$, $C \neq \emptyset$.

Observe that $V(C) \neq \emptyset$ for any coalition $C \neq \emptyset$. This follows from ii) and (1): $S = \top^M \in E(t)(C)$, where $\top$ is some tautology in the $\mathcal{L}_0$ language (e.g., $\omega' \vee \neg\omega'$), so $\exists_{\omega \in V(C)} \Gamma, \omega \models \top$, which ensures that $V(C)$ is non-empty.

We show that for any coalition $C \neq \emptyset$ and any $\omega \in \Omega$

$$\omega \in V(C) \Leftrightarrow \{\omega\} \in E(t)(C) \tag{2}$$

[3]We only give the coalitional clause. Pauly [14] uses $[C]$ where we use $\langle C \rangle$; here we use the latter notation for easier comparison.

For the direction to the right, assume that $\omega \in V(C)$. $\Gamma, \omega \models \omega$ (note the dual role of ω as both an outcome and a formula), so $\omega^M \in E(t)(C)$ by (1). By (b), $M, \omega \models \omega$, so $\omega \in \omega^M$. If $\omega' \in \omega^M$, then $M, \omega' \models \omega$, $\Gamma, \omega' \models \omega$ by (b) and $\omega' = \omega$. Thus, $\{\omega\} = \omega^M \in E(t)(C)$. For the direction to the left, let $\{\omega\} \in E(t)(C)$. Since, again, $\omega^M = \{\omega\}$, by (1) there is a $\omega' \in V(C)$ such that $\Gamma, \omega' \models \omega$. This is only the case when $\omega = \omega' \in V(C)$.

For any non-empty disjoint coalitions C_1 and C_2:

$$(\omega_1 \in V(C_1) \text{ and } \omega_2 \in V(C_2)) \Rightarrow \omega_1 = \omega_2 \quad (3)$$

$$V(C_1) = V(C_2) \quad (4)$$

$$|V(C_1)| = 1 \quad (5)$$

We prove (3)–(5):

(3) Assume otherwise, that $\omega_1 \in V(C_1)$ and $\omega_2 \in V(C_2)$ and $\omega_1 \neq \omega_2$. By (2), $\{\omega_1\} \in E(t)(C_1)$ and $\{\omega_2\} \in E(t)(C_2)$, and by superadditivity it must be the case that $\emptyset = \{\omega_1\} \cap \{\omega_2\} \in E(t)(C_1 \cup C_2)$, but this contradicts i). Thus, (3) must hold.

(4) Assume that $\omega \in V(C_1)$; we show that $\omega \in V(C_2)$. Since $V(C_2)$ is non-empty, let $\omega' \in V(C_2)$. By (3), $\omega' = \omega$. Thus $\omega \in V(C_2)$. By a symmetric argument, $\omega \in V(C_2)$ implies that $\omega \in V(C_2)$.

(5) Since $V(C_1)$ is non-empty, there is an $\omega_1 \in V(C_1)$. If $\omega_2 \in V(C_1)$, then $\omega_2 \in V(C_2)$ by (4) and $\omega_1 = \omega_2$ by (3). Thus, $V(C_1) = \{\omega_1\}$.

Let $a, b \in N$ such that $a \neq b$ (existence is ensured by the assumption of more than one player). By (4) and (5) there is an ω_1 such that $V(\{a\}) = V(\{b\}) = \{\omega_1\}$. For any $d \in N$ such that $d \neq a$ and $d \neq b$, $\{a\}$ and $\{d\}$ are disjoint and we again get that $V(\{d\}) = V(\{a\}) = \{\omega_1\}$. Thus, $V(\{d\}) = \{\omega_1\}$ for any $d \in N$. Let $C \subset N$ be a coalition different from the grand coalition. There is a $d \in N$ such that C and $\{d\}$ are disjoint, so by (4) $V(C) = V(\{d\}) = \{\omega_1\}$, which shows that Γ is limited. □

Thus, in general, a coalitional game is *not* simply a coalition model with outcomes as states. Even though the language of Coalition Logic is similar to the language of our logic, it follows from Theorem 8 that we cannot use the semantic rules of Coalition Logic "directly" to say whether a formula is true or not in a coalitional game. The main reason is that a difference between outcomes in coalitional games and states in coalition models is that an outcome is *local* to the coalition which chooses it, while states are *global*. As a consequence, while it is perfectly possible in a coalitional game that both a coalition C can choose outcome ω ($\omega \in V(C)$) *and* a coalition C', C' and C disjoint, can choose outcome ω' ($\omega' \in V(C')$) when $\omega' \neq \omega$, it is not possible in a coalition model that both C is effective for $\{\omega\}$ and C' is effective for $\{\omega'\}$. The proof of Theorem 8 shows that in general there is no playable effectivity function corresponding, in the sense of (2), to a characteristic function.

7. CONCLUSIONS AND FUTURE WORK

We introduced a knowledge representation language for coalitional games without transferable payoffs, and a logic specifically intended for reasoning about such games. We presented the logic, gave a complete axiomatization for it, showed it was expressively complete with respect to coalitional games without transferable payoff, showed that the satisfiability problem was NP-complete, showed how the logic could be used to capture a range of solution concepts for coalitional games, and finally, showed formally why the logic was fundamentally different to existing cooperation logics. Other solution concepts for coalitional games [11], for which the definitions in terms of payoffs cannot be trivially extended to the more general case of preferences and non-transferable utility, are the *the kernel*, *the nucleoulus* and *the Shapley value*. Logical characterisation of these concepts, in addition to further comparison with Coalition logic, could be interesting future work.

Acknowledgements

The research reported in this paper was carried out when the first author was visiting the Department of Computer Science, University of Liverpool, supported by grant 166525/V30 from the Norwegian Research Council.

8. REFERENCES

[1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, September 2002.

[2] R. J. Aumann and M. Maschler. The bargaining set for cooperative games. *Advances in Game Theory (Annals of Mathematics Studies 52)*, pages 443–467, 1964.

[3] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press: Cambridge, England, 2001.

[4] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press: Cambridge, MA, 2000.

[5] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[6] M. Davis and M. Maschler. Existence of stable payoff configurations for cooperative games. *Bulletin of the American Math. Soc.*, 69:106–108, 1963.

[7] D. B. Gillies. Solutions to general non-zero-sum games. *Contribution to the Theory of Games (Annals of Mathematics Studies, 40)*, IV:47–85, 1959.

[8] V. Goranko. Coalition games and alternating temporal logics. In J. van Benthem, editor, *Proceeding of the Eighth Conference on Theoretical Aspects of Rationality and Knowledge (TARK VIII)*, pages 259–272, Siena, Italy, 2001.

[9] H. J. Levesque. All I know: a study in autoepistemic logic. *Artificial Intelligence*, 42:263–309, 1990.

[10] J. Von Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour*. Princeton University Press: Princeton, NJ, 1944.

[11] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press: Cambridge, MA, 1994.

[12] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley: Reading, MA, 1994.

[13] M. Pauly. *Logic for Social Software*. PhD thesis, University of Amsterdam, 2001. ILLC Dissertation Series 2001-10.

[14] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002.

[15] B. Peleg. Existence theorem for the bargaining set $m_1^{(i)}$. *Bulletin of the American Math. Soc.*, 69:109–110, 1963.

[16] W. van der Hoek, W. Jamroga, and M. Wooldridge. A logic for strategic reasoning. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005)*, pages 157–153, Utrecht, The Netherlands, 2005.

Model Checking Knowledge, Strategies, and Games in Multi-Agent Systems

Alessio Lomuscio, Franco Raimondi
Department of Computer Science
University College London – London, UK
{a.lomuscio,f.raimondi}@cs.ucl.ac.uk

ABSTRACT

We present an OBDD-based methodology for verifying time, knowledge, and strategies in multi-agent systems specified by the formalism of interpreted systems. To this end, we investigate the interpretation of ATL and epistemic formulae in various classes of interpreted systems, we present model checking algorithms and their implementation, and report experimental results.

General Terms

Theory;Verification,Algorithms

Keywords

Model checking multi-agent systems; ATL

1. INTRODUCTION

Model checking was traditionally put forward to verify specifications given in *temporal* logics [6]. Recently, however, researchers have extended model checking techniques to other modal logics, including some typical multi-agent systems (MAS) logics, thereby making it possible to verify formally a range of multi-agent systems against temporal, epistemic, and other modalities. Examples of efforts on this line include [25, 4, 8, 19, 22]. These works share the model checking approach but differ in the choice of the logic-based specification language, and in the specific model checking technique employed.

In parallel developments, Alur et al. introduced Alternating-time Temporal Logic (ATL), a logic to reason about *strategies* in multi-player games [3]. Model checking approaches for this logic have been developed [2, 16]. More recently, van der Hoek and Wooldridge proposed the logic ATEL [11]. ATEL extends ATL with epistemic operators whose semantics is defined over runs of a multi-agent system. However, it has been argued [12, 15, 13, 14] that the interpretation of ATL operators in ATEL might not correspond entirely to the original spirit of ATL [3]. Following [11], various solutions have been put forward [15, 13, 14, 1, 10] to express ATL operators in a semantics based on MAS. Some of these issues relate back to the seminal work by Moore [18].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Against this background, and in parallel with this discussion, [11, 9, 20] suggested different techniques to reduce the problem of ATEL model checking to standard ATL model checking, with the idea of using MOCHA, the only existing model checker for ATL, for model checking MAS. In addition, the algorithms proposed in [16] are based on a reduction of the problem of model checking ATEL to a boolean satisfiability problem; however, no implementation is available yet.

In this paper we try to make further progress in this line by investigating interpretations of epistemic and ATL operators on interpreted systems, a mainstream semantics for MAS, and by presenting a verification methodology for it. In particular:

- we consider various classes of interpreted systems and we explore in detail the interpretation of ATL operators on these classes;
- we present model checking algorithms for ATL operators on various classes of interpreted systems;
- we present an extension for MCMAS [21] (a tool for the automatic verification of temporal and epistemic operators in interpreted systems) supporting ATL operators.

In contrast to previous approaches, our tool does not involve the translation or the reduction of the problem of model checking to plain ATL, and it permits the automatic verification of ATL operators on different classes of interpreted systems, thereby making it possible to express what agents may bring about, and what they may enforce. To our knowledge, this is not supported by any implementation. The tool is available for download from [21] under the terms of the GPL license.

The rest of the paper is organised as follows. In Section 2 we review theoretical preliminaries, and we compare briefly semantics for ATL and for interpreted systems. In Section 3 we discuss different classes of interpreted systems for the evaluation of ATL operators. In Section 4 we introduce the algorithms for model checking and we present the tool MCMAS. In Section 5 we evaluate the effectiveness of our approach by verifying three examples and we report experimental results. We conclude in Section 6.

2. PRELIMINARIES

In this section we review the main formalisms that we shall use in the remainder of this paper. We first introduce the logic ATL, then we describe the formalism of interpreted systems to model multi-agent systems, and we compare the semantics of interpreted systems with ATL semantics.

2.1 ATL

The syntax of the temporal logic ATL (Alternating-time Temporal Logic) [3] is defined as follows. Let AP be a finite set of atomic propositions, let $\Sigma = \{1, \ldots, n\}$ be a set of players, and let $\Gamma \subseteq \Sigma$ be a subset of the set of players. Let $p \in AP$ be an atomic proposition; well-formed ATL formulae are defined by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle\Gamma\rangle\rangle X\varphi \mid \langle\langle\Gamma\rangle\rangle G\varphi \mid \langle\langle\Gamma\rangle\rangle[\varphi U \psi]$$

In [3], the formula $\langle\langle\Gamma\rangle\rangle \cdot \varphi$ is read as "the set of players Γ can enforce $\cdot\varphi$" (where $\cdot$ denotes a temporal modality). ATL can be seen as an extension of the temporal logic CTL [6] where the path quantifiers E and A are replaced with quantification over sets of computations which players can "enforce". Indeed, quantification over all the possible computations is read as the existential quantifier of CTL, while quantification over the empty set is read as the universal quantifier of CTL.

Traditionally, the semantics of ATL formulae is given in terms of *concurrent game structures* (CGS). A CGS is a tuple

$$< \Sigma, S, AP, h, d, \delta >$$

where Σ is a set of players, S is a finite set of states, AP is a set of atomic propositions, $h : AP \rightarrow 2^S$ is an evaluation function, $d : \Sigma \times S \rightarrow I\!N$ is the number of moves available to a player i in a state (moves are labelled with natural numbers), and δ is an evolution function that determines the evolution of the system. Various definitions of δ are possible, we refer to [3] for more details. A *strategy for player* i is a function f_i that maps sequences of states to a natural number, corresponding to a move available to player i at the end of the sequence, i.e., $f_i : S^+ \rightarrow I\!N$, with $f_i(s) < d(i, s)$ for all states in the sequence. Given a state $s \in S$, a set of players Γ, and a set of strategies $F_\Gamma = \{f_i | i \in \Gamma\}$, the set $out(s, F_\Gamma) \subseteq S^+$ is the set of sequences that the group Γ can *enforce* in s, as defined in [3], p. 685. A sequence of states $s_0, s_1, \ldots$ such that each $s_i, i \geq 0$ is related to s_{i+1} via the evolution function δ is denoted with π, and $\pi(i) = s_i$ denotes the i-th state in the sequence. Sequences of states are usually called *computations*. Satisfaction of an ATL formula in a state $s \in S$ of a given CGS is defined as follows:

$s \models p$	iff	$s \in h(p)$,
$s \models \neg\varphi$	iff	$s \not\models \varphi$,
$s \models \varphi_1 \vee \varphi_2$	iff	$s \models \varphi_1$ or $s \models \varphi_2$,
$s \models \langle\langle\Gamma\rangle\rangle X\varphi$	iff	there exists a set of strategies F_Γ s.t. for all computations $\pi \in out(s, F_\Gamma)$, $\pi(1) \models \varphi$.
$s \models \langle\langle\Gamma\rangle\rangle G\varphi$	iff	there exists a set of strategies F_Γ s.t. for all computations $\pi \in out(s, F_\Gamma)$ and $\forall i \geq 0$, $\pi(i) \models \varphi$.
$s \models \langle\langle\Gamma\rangle\rangle[\varphi U \psi]$	iff	there exists a set of strategies F_Γ s.t. for all computations $\pi \in out(s, F_\Gamma)$, $\exists i \geq 0$ s.t. $\pi(i) \models \psi$ and $\forall 0 \leq j < i, \pi(j) \models \varphi$

It is worth noticing here that the definitions above are given with the intuition that *every player has complete information about the system and perfect recall* (see Section 2.3 for further details).

2.2 Interpreted systems

An interpreted system [7] is a formal description of the computations carried out by a set of agents $\Sigma = \{1, \ldots, n\}$. Each agent $i \in \Sigma$ is characterised by a finite set of *private local states* L_i and by a finite set of actions Act_i that may be performed. Actions are performed in compliance with a protocol $P_i : L_i \rightarrow 2^{Act_i}$. Notice that this definition of protocols allows for *non-determinism* in the system. The environment in which agents "live" may be modelled by means of a special agent E, modelled by a set of local states L_E, a set of actions Act_E, and protocol P_E. A tuple $g = (l_1, \ldots, l_n, l_e) \in L_1 \times \cdots \times L_n \times L_E$ is called a *global state* and gives a description of the system at a particular instant of time. The evolution of the agents' local states is described by a function $t_i : L_i \times L_E \times Act_1 \times \cdots \times Act_n \times Act_E \rightarrow L_i$ which gives the "next" local state as a function of the current local state of the agent, the environment, and all the other agents' actions (the evolution function for the environment is defined as $t_E : L_E \times Act_1 \times \cdots \times Act_n \times Act_E \rightarrow L_E$). It is assumed that, in every state, agents evolve simultaneously (this assumption corresponds to the definition of *Moore synchronous* CGS, in which the evolution function δ prescribes a simultaneous evolution of players). The evolution of the (global states of the) system may be described by a function $t : G \times Act \rightarrow G$, where $G \subseteq (L_1 \times \cdots \times L_n \times L_E)$ denotes the set of *reachable* global states, and $Act = Act_1 \times \cdots \times Act_n \times Act_E$ denotes the set of "joint" actions (similarly, one can reason about the joint actions of a group of agents $\Gamma \subseteq \Sigma$). The function t is the composition of all the functions t_i, and it is defined by $t(g, a) = g'$ iff $\forall i, t_i(l_i(g), l_E(g), a) = l_i(g')$, where $l_i(g)$ denotes the local state of agent i in global state g and $a \in Act$. The set G of reachable global states is obtained by considering all the possible evolutions of the system from a set of *initial* global states, denoted with I. Finally, to complete the description of an interpreted system, a set of atomic propositions AP is introduced, together with a valuation function $h : AP \rightarrow 2^G$.

Formally, an interpreted systems IS is a tuple $IS = \left\langle (L_i, Act_i, P_i, t_i)_{i \in \Sigma \cup \{E\}}, I, h \right\rangle$. Interpreted systems have been proven a suitable formalism for reasoning about temporal and epistemic properties of agents [7, 17]. The standard semantics of interpreted systems is given Section 3.

2.3 On the relationship between interpreted systems and concurrent game structures

Interpreted systems and concurrent game structures are closely related. Consider an interpreted system

$$IS = \left\langle (L_i, Act_i, P_i, t_i)_{i \in \Sigma \cup \{E\}}, I, h \right\rangle$$

and a concurrent game structure

$$CGS =< \Sigma, S, AP, h, d, \delta >:$$

- both structures comprise a set of agents (or players) Σ, and both structures comprise a set of states, called *global* states in IS;
- the function d, which returns the number of moves available to a player in a state of a CGS, intuitively corresponds to the protocols P_i in IS;
- the evolution function δ of CGS is an "accessibility" relation between states, as the evolution function t of IS, defined in terms of the evolution functions t_i;
- the valuation functions h of CGS and IS both label (global) states with propositions.

Differently from CGS, however, global states of interpreted systems have a structure, being tuples of *private local states*. This subtle difference is key in understanding the difference in the semantics for ATL operator in CGS and in interpreted systems presented in the next section. Indeed, CGS assume that every player

has *perfect information*, i.e., every player is fully aware of the state $s \in S$ at every time instant. Conversely, in interpreted systems an agent is aware of its private local state only.

A further difference between interpreted systems and CGS is the definition of strategies. In [3], a strategy is defined as a function from *sequences* of states to an action. In the case of interpreted systems, instead, we define below a strategy for agent i to be a function from a (single) local state to an action of agent i (see the next Section for a discussion about this). In this sense, strategies and protocols in interpreted systems are closely related because they associate actions to states and, in the case of *deterministic* interpreted systems (see Section 3.2), strategies and protocols are the same mathematical object. In the remainder of the paper we will refer to protocols when describing interpreted systems, and to strategies when reasoning about ATL formulae in interpreted systems.

Explanations of other technical similarities and differences between the two semantics is not the aim of the present paper.

3. ATL OPERATORS IN INTERPRETED SYSTEMS

In this section we discuss how interpreted systems can provide a semantics for a language that includes operators to reason about time, knowledge, and strategies. Formally, let AP be a set of atomic propositions and let $p \in AP$ be an atomic proposition; the syntax of the language we consider is defined by the following grammar:

$$\begin{aligned}\varphi \quad ::= \quad & p \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\varphi \mid EG\varphi \mid E[\varphi U \psi] \mid \\ & K_i\varphi \mid E_\Gamma\varphi \mid C_\Gamma\varphi \mid D_\Gamma\varphi \mid \\ & \langle\langle\Gamma\rangle\rangle X(\varphi) \mid \langle\langle\Gamma\rangle\rangle G(\varphi) \mid \langle\langle\Gamma\rangle\rangle[\varphi U \psi]\end{aligned}$$

In the grammar above, $\Gamma \subseteq \Sigma$ denotes a set of agents, $K_i\varphi$ is read as "agent i knows φ", $E_\Gamma\varphi$ is read as "everybody in group Γ knows φ", $C_\Gamma\varphi$ is read as "it is common knowledge in Γ that φ" [7], $D_\Gamma\varphi$ is read as "it is distributed knowledge in Γ that φ, and $\langle\langle\Gamma\rangle\rangle \cdot \varphi$ (where $\cdot$ is one of the temporal operators X, G, or U) is read as "group Γ can enforce $\cdot\varphi$". Other temporal and ATL operators expressing eventuality (F), and the universal quantifier (A) for paths, can be derived in a standard way. Notice that, although the temporal operators can be derived from the ATL operators, we treat them separately to stress the difference between temporal reasoning and strategic reasoning; we refer to [3, 6] for details.

To provide a semantics for well-formed formulae we proceed as follows. Given an interpreted system IS, we associate a Kripke model M_{IS} to IS. The *associated model* $M_{IS} = (G, t, (\sim_i)_{i\in\Sigma}, L)$ for a given interpreted system $IS = \left\langle (L_i, Act_i, P_i, t_i)_{i\in\Sigma\cup\{E\}}, I, h \right\rangle$ is defined as follows:

- The set of possible worlds G is the set of reachable states of IS. The set G can be obtained from the set of initial states I by iterating the evolution function t defined by the protocols and the evolution functions of each agent in Section 2.2.
- The temporal relation $t \subseteq G \times Act \times G$ relating two worlds by means of a joint action is the relation defined in Section 2.2.
- Each epistemic accessibility relations $\sim_i$ is defined by the equality of the local components of the global states: $g \sim_i g'$ iff $l_i(g) = l_i(g')$, i.e. iff the local states of agent i in global states g and g' are the same [7].
- The labelling function $L : AP \to 2^G$ is equivalent to the evaluation function h.

Temporal and epistemic formulae are interpreted on M_{IS} in a standard way; we refer to [7, 6, 3] for details. Similarly to Section 2.1, we define a strategy to be a function $f_i : L_i \to Act_i$ from (local) states to actions for agent i, with the requirement that $f_i(l_i(g)) \in P_i(l_i(g))$, i.e. $f_i(l_i(g))$ is one of the actions enabled by the protocol P_i for agent i when its local state is $l_i(g)$. Notice that this definition corresponds to the memoryless, imperfect information semantics of ATL [23]. Different partial recall semantics could be defined by using vectors of local states. Given a group of agents Γ and a state $g \in G$, let $F_\Gamma = \{f_i | i \in \Gamma\}$ be a set of strategies; as in Section 2.1, we denote with $out(g, F_\Gamma)$ the set of computations that group Γ can enforce at state g (computations are defined as in Section 2.1). For convenience, we report here the semantics for ATL and epistemic formulae.

- $M_{IS}, g \models K_i\varphi$ iff for all $g' \in G$, $g \sim_i g'$ implies $M_{IS}, g' \models \varphi$.
- $M_{IS}, g \models \langle\langle\Gamma\rangle\rangle X\varphi$ iff there exists a set of strategies F_Γ such that, for all computations $\pi \in out(g, F_\Gamma)$, $\pi(1) \models \varphi$.
- $M_{IS}, g \models \langle\langle\Gamma\rangle\rangle G\varphi$ iff there exists a set of strategies F_Γ such that, for all computations $\pi \in out(g, F_\Gamma)$ and for all $i \geq 0$, $\pi(i) \models \varphi$.
- $M_{IS}, g \models \langle\langle\Gamma\rangle\rangle[\varphi U \psi]$ iff there exists a set of strategies F_Γ such that for all computations $\pi \in out(g, F_\Gamma)$, there exists $i \geq 0$ such that $\pi(i) \models \psi$, and for all $0 \geq j < i$ $\pi(j) \models \varphi$.

Following [7] (p.118), we write $IS \models \varphi$ if $M_{IS} \models \varphi$.

It is known that non-intuitive results may arise when evaluating ATL operators in MAS logics [15, 13, 14, 1]. We investigate this issue for the semantics of interpreted systems in the next subsections, by considering various classes of interpreted systems.

3.1 Non-deterministic interpreted systems

The most general class of interpreted systems is the one defined in Section 2.2. Consider the following simple example of an interpreted system IS[1] composed by two agents Ag_1 and E (the environment); the agent Ag_1 is modelled by means of three local states $L_1 = \{l_1^1, l_1^2, l_1^3\}$ and two possible actions $Act_1 = \{a_1^1, a_1^2\}$. The protocol for Ag_1 prescribes that every action is allowed in every state. The second agent E is modelled by means of two local states $L_E = \{l_E^1, l_E^2\}$, and we assume that no action is performed by E (and thus no protocol is required). In the initial state the system can be either in global state (l_1^1, l_E^1) or in global state (l_1^1, l_E^2). The evolution function for Ag_1 is defined below:

Loc. st. Ag_1	Loc. st. E	Act. Ag_1	Next state
l_1^1	l_E^1	a_1^1	l_1^2
l_1^1	l_E^1	a_1^2	l_1^3
l_1^1	l_E^2	a_1^1	l_1^3
l_1^1	l_E^2	a_1^2	l_1^2

We assume that in every other state there is no change in local state for Ag_1, and we assume that the local state of E never changes. We introduce an evaluation function h assigning the proposition **WIN** to the global states (l_1^2, l_E^1) and (l_1^2, l_E^2). The model associated to IS is depicted in Figure 1.

In this simple example the formula $\langle\langle Ag_1\rangle\rangle X(\mathbf{WIN})$ is true in the initial states of IS; however, it is not the case that Ag_1 is capable of *enforcing* a state in which the proposition **WIN** holds. Indeed, Ag_1 cannot distinguish between the global states (l_1^1, l_E^1) and (l_1^1, l_E^2),

[1]This is a reduced version of the card game presented in [15, 13].

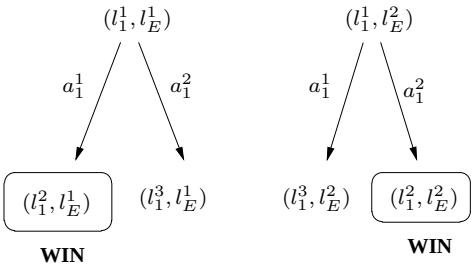


Figure 1: A simple example.

because its local state is the same in the two global states. We conclude that, similarly to what happens in the case of ATEL, ATL operators in non-deterministic interpreted systems do not express what agents may enforce, but what agents may *bring about*. In our example, the formula $\langle\langle Ag_1 \rangle\rangle X(\textbf{WIN})$ has to be read "Ag_1 may bring about a state in which the proposition **WIN** holds, maybe by *guessing* moves".

3.2 Deterministic interpreted systems

The reason the meaning of the ATL operators is subtly different in interpreted systems is that these allow for agents to run non-deterministic protocols, i.e., the same agent may perform different actions in a given local state. To follow the spirit of the original CGS more closely, we can focus on the subclass of interpreted systems, whose protocols are deterministic, i.e., protocols in which only one action is associated to a given local state: $P_i : L_i \rightarrow Act_i$.

We define an interpreted system to be *deterministic* iff the protocol of each agent is *deterministic* (a deterministic protocol associates a unique action to each local state). Notice that, since we assume the transition function to be deterministic, the models associated to deterministic interpreted systems are not branching but linear.

Since in deterministic interpreted systems agents are not allowed to "guess" actions, the evaluation of ATL operators in their associated expresses the original "enforcement" meaning of ATL operators in CGS [3].

3.3 Γ-uniform interpreted systems

Deterministic interpreted systems are close in spirit to the original interpretation of ATL in CGS, but, as exemplified later in Section 5, in many circumstances the class of deterministic interpreted systems is too restrictive to be used in the specification of MAS scenarios. In these circumstances it is useful to reason about non-deterministic interpreted systems that at least are consistent in their selection of actions in a given local state.

By extending the concepts of [15, 14], we define an agent to be *uniform* if throughout a run the agent performs always the same action when in a particular local state. A group of agents $\Gamma \subseteq \Sigma$ is *uniform* if every agent in the group is uniform. We say that an interpreted system is Γ-*uniform* if all agents in Γ are uniform. We define a Γ-uniform interpreted system IS_Γ to be *compatible* with an interpreted system IS if (i) it contains the same agents, with the same set of local states, actions, and evolution function, and (ii) the protocols P_i^Γ for the Γ agents in IS_Γ are a *restriction* of the protocols P_i of IS, in the sense that for all local states $l_i \in L_i$, $P_i^\Gamma(l_i) \in P_i(l_i)$ (i.e. only one action is selected in P_i^Γ, and the action is one of the actions specified in P_i).

Notice that, for a given interpreted system IS and a group of agents Γ, there may be several Γ-uniform interpreted systems compatible with IS (but at most $\prod_{i \in \Gamma} |Act_i|^{|L_i|}$). We denote with $\{IS\}_\Gamma$ the set of Γ-uniform interpreted systems that are compatible with a given interpreted system IS.

Similarly to the general case, we associate a Kripke model to each Γ-uniform interpreted system in $\{IS\}_\Gamma$. We say that a formula φ is true in a class of Γ-uniform interpreted systems, and write $IS \models_\Gamma \varphi$, if φ is true in at least one of the models associated with the Γ-uniform interpreted system in $\{IS\}_\Gamma$.

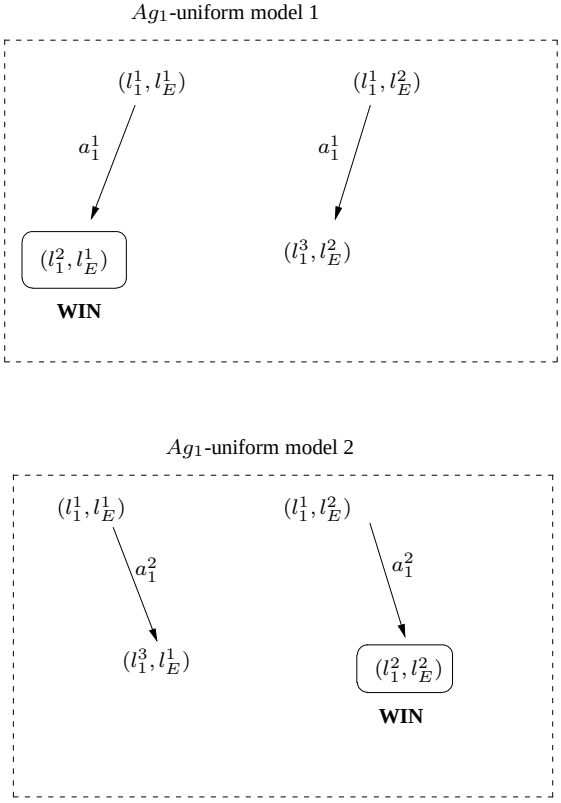


Figure 2: Ag_1-uniform interpreted systems for the interpreted system of Figure 1.

As an example, the uniform interpreted systems for the example of Figure 1 are presented in Figure 2. In this example, in the initial states we have $IS \models \langle\langle Ag_1 \rangle\rangle X(win)$, but $IS \not\models_{Ag_1} \langle\langle Ag_1 \rangle\rangle X(win)$, because there is no Ag_1-uniform interpreted system compatible with the original interpreted system in which the formula is true. Intuitively, a uniform agent chooses one of the actions allowed by the protocols, and it behaves consistently with its choice, i.e., the agent will always choose the same action in epistemically equivalent states.

4. OBDD-BASED MODEL CHECKING FOR INTERPRETED SYSTEMS

In this section we present the algorithms for the verification of temporal, epistemic, and ATL operators in the various classes of interpreted systems presented above. Our approach is similar, in spirit, to the traditional model checking techniques for the temporal logic CTL [6], and to the approach presented in [22].

4.1 Model checking algorithm

To evaluate epistemic, temporal, and ATL operators in models associated to non-deterministic interpreted systems we proceed as follows. Given an interpreted system $IS = \left\langle (L_i, Act_i, P_i, t_i)_{i \in \Sigma \cup \{E\}}, I, h \right\rangle$, note that the number of boolean variables $v_i (i \in I\!N)$ required to encode the local states of an agent is $nv(i) = \lceil log_2 |L_i| \rceil$. Similarly, to encode an agent's action, the number of boolean variables $w_i (i \in I\!N)$ required is $na(i) = \lceil log_2 |Act_i| \rceil$. A global state g can be encoded as a boolean vector $\overline{v} = (v_1, \ldots, v_N)$, where $N = \sum_i nv(i)$. Similarly, a joint action a can be encoded as a boolean vector $\overline{w} = (w_1, \ldots, w_M)$, where $M = \sum_i na(i)$. In turn, a boolean vector can be identified with a boolean formula, represented by a conjunction of literals, i.e. a conjunction of variables or their negation. In this way, a set of global states (and similarly a set of joint actions) can be expressed as the disjunction of the boolean formulae encoding each global state in the set. Given this, protocols can be encoded by implications between boolean formulae representing local states and actions (see [22] for more details). Let $P(\overline{v}, \overline{w})$ be the boolean formula obtained by taking the conjunction of the boolean formulae encoding the protocols for all the agents. The evolution function t_i for a single agent can also be expressed as a boolean formula; we denote with $t(\overline{v}, \overline{w}, \overline{v'})$ the boolean function obtained by taking the conjunction of all the boolean formulae encoding the evolution functions for the agents. Notice that evolution functions require two sets of boolean variables for encoding states; in the above $\overline{v'} = (v'_1, \ldots, v'_N)$ denotes the additional set of boolean variables required. The set of initial states can be encoded by means of a boolean formula $I(\overline{v})$, as well as the evaluation function h.

In addition to the parameters presented above, the algorithm for model checking presented below requires the definition of n boolean functions $R_i^K(\overline{v}, \overline{v'})$ (one for each agent) representing the epistemic accessibility relation, and the definition of a boolean function $R_t(\overline{v}, \overline{v'})$, representing a temporal transition between the global states g and g' encoded by means of the boolean vectors $(v_1, \ldots, v_N)$ and $(v'_1, \ldots, v'_N)$. $R_t(\overline{v}, \overline{v})$ can be obtained from the boolean function t by quantifying over actions. This quantification can be translated into a propositional formula using a disjunction (see [6] for a similar approach to boolean quantification):

$$R_t(\overline{v}, \overline{v'}) = \bigvee_{\overline{w} \in Act} [(t(\overline{v}, \overline{w}, \overline{v'}) \wedge P(\overline{v}, \overline{w})]$$

The formula above provides a boolean relation between global states that can be used in the evaluation of temporal operators. Also, the set of *reachable* states is needed by the algorithm: the set G of reachable global states can be expressed *symbolically* by a boolean formula, and it can be computed as the fix-point of the operator $\tau(Q) = (I(\overline{v}) \vee (\exists(\overline{v'})).(R_t(\overline{v'}, \overline{v}) \wedge Q(\overline{v'}))$. Intuitively, $\tau(Q)$ computes the sets of states that are reachable from Q in a single temporal step. The fix-point of τ can be computed by iterating $\tau(\emptyset)$ as standard (see [6]).

Figure 3 presents the algorithm for model checking, based on the parameters presented above. In the following, Act_Γ denotes a joint action performed by group Γ. Let $a \in Act_\Gamma$ and $b \in Act_{\Sigma \setminus \Gamma}$: we denote with (a, b) the joint action defined by the concatenation of a and b, with the appropriate reordering of elements, if needed. $SAT(\varphi)$ computes the set of global states (expressed as a boolean formula) in which φ holds. The support procedures SAT_K and SAT_X are presented in Figure 4, while SAT_G, SAT_U, SAT_E, SAT_C, and SAT_D are defined using the ideas of [3, 11, 22]. The

$SAT(\varphi)$ {
 φ is an atomic formula: return $h(\varphi)$;
 φ is $\neg\varphi_1$: return $G \setminus SAT(\varphi_1)$;
 φ is $\varphi_1 \wedge \varphi_2$: return $SAT(\varphi_1) \cap SAT(\varphi_2)$;
 φ is $\langle\langle\Gamma\rangle\rangle X\varphi_1$: return $SAT_X(\varphi_1, \Gamma)$;
 φ is $\langle\langle\Gamma\rangle\rangle[\varphi_1 U \varphi_2]$: return $SAT_U(\varphi_1, \varphi_2, \Gamma)$;
 φ is $\langle\langle\Gamma\rangle\rangle G\varphi_1$: return $SAT_G(\varphi_1, \Gamma)$;
 φ is $K_i(\varphi)$: return $SAT_K(\varphi, i)$;
 φ is $E_\Gamma(\varphi)$: return $SAT_E(\varphi, \Gamma)$;
 φ is $C_\Gamma(\varphi)$: return $SAT_C(\varphi, \Gamma)$;
 φ is $D_\Gamma(\varphi)$: return $SAT_D(\varphi, \Gamma)$;
}

Figure 3: Model checking algorithm

$SAT_X(\varphi, \Gamma)$ {
 Y = $\{g | (\exists a \in Act_\Gamma, g' \in G)$ s.t. $(\forall b \in Act_{\Sigma \setminus \Gamma}).[R_t(g, g')$ and $t(g, (a, b), g')$ and $g' \in SAT(\varphi)$ and (a, b) is consistent with the protocols in $g']\}$
 return Y;
}

$SAT_K(\varphi, i)$ {
 X = $SAT(\neg\varphi)$;
 Y = $\{g \in G$ s.t. $R_i^K(g, g')$ and $g' \in X\}$
 return $\neg$Y;
}

Figure 4: Support procedure for SAT_X and SAT_K

key difference in our work is the definition of temporal transitions by means of interpreted systems' parameters (i.e. actions, protocols, evolution functions, etc.) as presented above. The procedure $SAT_X(\varphi, \Gamma)$ uses a double quantification on actions and returns the set of states from which there exists an action for the agents in Γ such that, for all actions of the agents in $\Sigma \setminus \Gamma$, a transition is enabled such that in the next state φ holds. In the support procedures, by slight abuse of notation, we use a compact notation for global states and actions instead of boolean variables and vectors; the intended meaning should be clear from the context.

The algorithm presented above is designed to be compatible with the representation of boolean formulae by means of Ordered Binary Decision Diagrams (OBDDs, [5]): OBDDs are an efficient technique for the representation and manipulation of boolean formulae. Due to space limitation, we refer to [5] for more details about this technique.

The model checking algorithm above terminates by comparing the OBDD representing the set of states in which a formula φ holds with the OBDD for the set of reachable states. Notice that if the two OBDDs are equal, then the Boolean formulae generating them are propositionally equivalent, therefore $IS \models \varphi$.

4.2 Model checking Γ-uniform interpreted systems

The algorithm presented in Figure 3 can also be used for the evaluation of formulae in Γ-uniform interpreted systems. Indeed, the algorithm can be applied to the symbolic encoding of each model associated to each interpreted system in $\{IS\}_\Gamma$.

To evaluate epistemic, temporal, and ATL operators in Γ-uniform interpreted systems, the set $\{IS\}_\Gamma$ is built as follows:

- The set of deterministic joint protocols[2] P_Γ is computed by imposing that all the agents in Γ adhere to a deterministic

[2]A joint protocol extends the definition of protocol to a set of agents. Formally, given a set of agents Γ, a joint protocol is de-

protocol, and by imposing that elements of P_Γ are *restrictions* (in the sense of Section 3.3) of the protocols of IS. An element of P_Γ can be seen as a tuple of deterministic protocols, one for each agent in Γ. Notice that there are at most $\prod_{i\in\Gamma} |A_i|^{|L_i|}$ different elements in P_Γ, where A_i is the maximum number of non-deterministic choices available to an agent.

- For each element $p \in P_\Gamma$, an interpreted system IS_p is defined as:
  - The set of agents, their local states, actions, and evolution functions for IS_p are equal to the corresponding elements of IS.
  - The protocols for the agents in Γ in IS_p are defined by p, while the protocols for agents in $\Sigma\backslash\Gamma$ are equal to the protocols in IS.
- The set of Γ-uniform interpreted systems $\{IS\}_\Gamma$ is defined as $\{IS\}_\Gamma = \{IS_p | p \in P_\Gamma\}$.

To check $IS \models_\Gamma \varphi$, we check whether $M_{IS_p} \models \varphi$ for some $p \in P_\Gamma$.

It is worth noting that the time complexity of verifying $IS \models_\Gamma \varphi$ by means of the algorithm presented above exceeds, in the worst case, the time complexity of $IS \models \varphi$ by a factor of $\prod_{i\in\Gamma} |A_i|^{|L_i|}$. The space complexity of the verification, instead, remains the same because the space employed in the verification of each model M_s associated with Γ-uniform interpreted systems in $\{IS\}_\Gamma$ can be reused.

4.3 MCMAS

In this section we present MCMAS, a tool that implements the algorithms presented in Section 4.2. In MCMAS, interpreted systems are described using the language ISPL (Interpreted Systems Programming Language). Figure 5 gives a short example of this language. We refer to the files available online [21] for the full syntax of ISPL. Formulae to be checked are provided at the end of the specification file.

The tool automatically parses the specification and builds the relevant parameters, stored as OBDDs by using the library provided in [24]. As discussed in Section 3, formulae can be evaluated either in models associated to non-deterministic interpreted systems, or in the class of Γ-*uniform* interpreted systems. MCMAS accepts a command-line parameter to select which class should be considered.

When the class of Γ-*uniform* interpreted systems is chosen, MCMAS determines the set Γ by including in Γ all the agents appearing under the scope of ATL operators in φ (notice that other implementation choices are possible without modifying the structure of MCMAS, for instance providing a separate list of uniform agents, irrespective of φ). To optimise the verification process, the set P_Γ is computed first, and then for each $p \in P_\Gamma$ a model is generated as in Section 4.2 and checked. If the formula is true in the model, the verification process ends with success; otherwise, another element $p \in P_\Gamma$ is chosen and the loop is repeated until all the elements of P_Γ have been tested, and the process terminates with failure.

MCMAS can be run from the command line, and accepts various options to modify verbosity, to inspect OBDDs statistics and memory usage, to enable variable reordering in the OBDDs (see [24]), etc. These options can be used to determine the "critical" points, and to fine tune the performance of the tool.

MCMAS is written in C/C++ and it has been successfully compiled on various platforms, including PowerPC (Mac OS X 10.2 and 10.3), Intel (various Pentium versions using Linux 2.4 and 2.6), and SPARC (SunOS 5.8 and 5.9). The source code has been compiled with gcc/g++ from version 2.95 till version 3.3.

fined as a function from the Cartesian product of the local states of the agents in Γ to the power set of the Cartesian product of the actions of the agents in Γ.

```
Agent SampleAgent
 Lstate = {s0,s1,s2,s3};
 Action = {a1,a2,a3};
 Protocol:
   s0: {a1};
   s1: {a2};
   s2: {a1,a3};
   s3: {a2,a3};
 end Protocol
 Ev:
   s2 if ((AnotherAgent.Action=a7);
   s3 if Lstate=s2;
 end Ev
end Agent
```

Figure 5: ISPL example

5. EXAMPLES AND EXPERIMENTAL RESULTS

5.1 Nim

Nim is a two player game where players in turns remove any number of objects from one of a certain number of heaps. Typically, 3 heaps are presents and the game starts with 3 objects in the first heap, 4 in the second, and 5 in the third. The player who takes the last object wins. In a variation of this game, called *Misère*, the player who takes the last object loses.

This is a game with *perfect* information. It is not difficult to encode the game in the formalism of interpreted systems, by imposing that only one agent makes a move at any given time step. The corresponding ISPL code is available for download from [21].

We considered two examples with 3-4-5 heaps, and with 5-5-5 heaps. We could confirm the known result that the first player can force a win *both* for the Nim and the Misère scenario. To this end, we checked the two formulae:

$$init \rightarrow \langle\langle player1 \rangle\rangle [\neg player2\_removelast \; U \; player1\_removelast]$$

$$init \rightarrow \langle\langle player1 \rangle\rangle [\neg player1\_removelast \; U \; player2\_removelast]$$

(where "init" is a proposition true in the initial state of the system).

We could verify these formulae in 18 seconds (for 3-4-5) and in 4 minutes and 8 seconds (for 5-5-5 heaps) using a 2.8GHz Intel Pentium IV, 1Gb of RAM, running Linux 2.6.8. The model for 3-4-5 heaps requires 51 boolean variables for the symbolic encoding, while 57 boolean variables are needed for the encoding of the 5-5-5 heaps (corresponding approximately to models of size 10^{15} and 10^{17} respectively).

5.2 A simple card game

This example is presented in [13] and in [15] to analyse the effects of incomplete information in MAS: an agent (the player) plays a simple card game against another agent, the environment. There are just three cards in the deck: Ace (A), King (K), and Queen (Q); A wins over K, K wins over Q, and Q wins over A. In the initial state no cards are distributed; in the first step, the environment gives

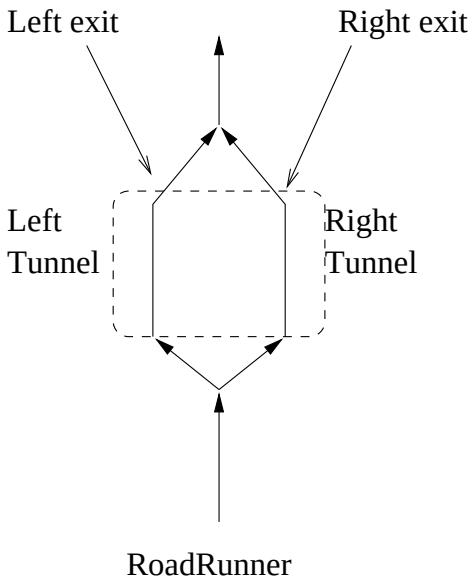


Figure 6: Diagram for RoadRunner and Coyote

a card to the player and takes a card for itself. In the second step, the player can either keep its card, or change it. A description of this scenario in terms of interpreted systems can be easily obtained, and it is available in the downloadable files from [21]. We want to verify whether or not the player has a strategy to win the game in the initial state, i.e. we want to verify the following formula:

$$init \rightarrow \langle\langle player \rangle\rangle F(player\_win)$$

Differently from the original intended meaning of ATL operators, this formula is true when evaluated with MCMAS: indeed, the player can always *guess* a move to win. However, the formula is false if it is evaluated in the class of *uniform* interpreted systems. This result confirms the intuition that the player cannot enforce a winning state.

Verification of this scenario took 0.15 seconds on a 2.8GHz Intel Pentium IV, 1Gb of RAM, running Linux 2.6.8.

5.3 RoadRunner and Coyote

This example illustrates further the different meaning of ATL operators in Γ-uniform interpreted systems and in non-deterministic interpreted systems.

RoadRunner is running in a hilly region of the desert; the main road splits in two small lanes just before the entrance of two tunnels below a mountain. RoadRunner can pick randomly either tunnel; the tunnels are identical and very narrow. Coyote knows RoadRunner has to take one of the two tunnels, and so he has bought a special tunnel-blocking device from ACME Inc. to catch RoadRunner. The device may be placed in front of either exit of the tunnel (see Figure 6).

The example can be modelled as an interpreted system IS by taking two agents, one for Coyote and one for RoadRunner. RoadRunner is described by means of two local states, "left" and "right", and by means of the single action "run"; local states for RoadRunner do not change. Coyote is modelled by means of three local states: "planning", "catch", and "fail". The initial state of Coyote is "planning" and in this local state Coyote is allowed to perform one of the two actions "place Left" or "place Right" denoting where he places the special device. The evolution function for Coyote prescribes that the next local state for Coyote is "catch" if he places the special device in front of the tunnel chosen by RoadRunner.

We introduce the proposition "catch", which is true if Coyote catches RoadRunner. The source code of the example is available from [21]. By using MCMASCoyote discovers that:

$$IS \models \text{init} \rightarrow \langle\langle Coyote \rangle\rangle X(\text{catch})$$

i.e. it is the case that in the initial state Coyote has a strategy to catch RoadRunner.

In fact, any external observer could verify that Coyote knows this very well:

$$IS \models K_{\text{Coyote}}(\text{init} \rightarrow \langle\langle Coyote \rangle\rangle X(\text{catch}))$$

Unfortunately, immediately after placing the ACME device in front of the tunnel, Coyote realises that he was assuming a *lucky* guess on where to place the device. Indeed, under the assumption Coyote is uniform $\Gamma = \{\text{Coyote}\}$, the formula turns to be false

$$IS' \not\models_{\Gamma} \text{init} \rightarrow \langle\langle Coyote \rangle\rangle X(\text{catch})$$

and, as expected, Coyote miserably fails in catching RoadRunner.

From RoadRunner's point of view, however, it is more useful and prudent to reason about what the clumsy Coyote may bring about. Thus, RoadRunner should be more interested in the verification of the non-deterministic interpreted system to discover that it is possible that Coyote catches him. In other words, to analyse the scenario from RoadRunner's point of view, we can check:

$$IS \models K_{\text{RoadRunner}}(\text{init} \rightarrow \langle\langle Coyote \rangle\rangle X(\text{catch}))$$

and also

$$IS \models K_{\text{RoadRunner}}(\text{init} \rightarrow EX(\text{catch})).$$

Given this Coyote should perhaps not be laughing so much.

Verification of all the formulae above took 0.19 seconds on a 2.8GHz Intel Pentium IV, 1Gb of RAM, running Linux 2.6.8.

6. CONCLUSIONS

In this paper we have identified three classes of interpreted systems to evaluate temporal, epistemic, and ATL operators. In the first class agents may guess moves; here, ATL operators express what agents may *bring about*. The second class is constituted by deterministic agents; therefore, ATL operators express what agents may *enforce*. The third class comprises Γ-uniform interpreted systems, non-deterministic systems in which a group of agents Γ chooses consistently to perform certain actions. We introduced model checking algorithms for these classes together with an implementation. Examples have also been presented to motivate the use of the various classes.

Acknowledgements

We gratefully acknowledge Michael Wooldridge and Wojciech Jamroga for their valuable comments on an earlier version of this paper. Any current mistakes rest with the authors.

7. REFERENCES

[1] T. Agotnes. Action and knowledge in Alternating-time Temporal Logic. *Synthese*, 2005. Special issue on Knowledge, Rationality and Action.

[2] R. Alur, T. Henzinger, F. Mang, S. Qadeer, S. Rajamani, and S. Tasiran. MOCHA: Modularity in model checking. In *Proceedings of the 10th International Conference on Computer Aided Verification (CAV'98)*, volume 1427 of *LNCS*, pages 521–525. Springer-Verlag, 1998.

[3] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[4] R. H. Bordini, M. Fisher, C. Pardavila, and M. Wooldridge. Model checking AgentSpeak. In J. S. Rosenschein, T. Sandholm, W. Michael, and M. Yokoo, editors, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-agent systems (AAMAS-03)*, pages 409–416. ACM Press, 2003.

[5] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transaction on Computers*, 35(8):677–691, 1986.

[6] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

[7] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, 1995.

[8] P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In *Proceedings of 16th International Conference on Computer Aided Verification (CAV'04)*, volume 3114 of *LNCS*, pages 479–483. Springer-Verlag, 2004.

[9] V. Goranko and W. Jamroga. Comparing semantics for logics of multi-agent systems. *Synthese*, 139(2):241–280, 2004.

[10] W. Hoek, W. Jamroga, and M. Wooldridge. A logic for strategic reasoning. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 157–164, New York, NY, USA, 2005. ACM Press.

[11] W. Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In M. Gini, T. Ishida, C. Castelfranchi, and W. L. Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 1167–1174. ACM Press, 2002.

[12] W. Jamroga. Some remarks on alternating temporal epistemic logic. In B. Dunin-Kęplicz and R. Verbrugge, editors, *Proceedings of the International Workshop on Formal Approaches to Multi-Agent Systems (FAMAS'03)*, pages 133–140, 2004.

[13] W. Jamroga. *Using Multiple Models of Reality. On Agents who Know how to Play Safer.* PhD thesis, University of Twente, Enschede, The Netherlands, 2004.

[14] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 62:1–35, 2004.

[15] G. Jonker. Feasible strategies in alternating-time temporal epistemic logic. Master's thesis, University of Utrech, The Netherlands, 2003.

[16] M. Kacprzak and W. Penczek. A SAT-based approach to unbounded model checking for alternating-time temporal epistemic logic. *Synthese*, 142:203–227, 2004.

[17] A. Lomuscio. *Knowledge Sharing among Ideal Agents*. PhD thesis, School of Computer Science, University of Birmingham, Birmingham, UK, June 1999.

[18] R. C. Moore. A formal theory of knowledge and action. In J. Allen, J. Hendler, and A. Tate, editors, *Readings in Planning*, pages 480–519. Kaufmann, San Mateo, CA, 1990.

[19] W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, and M. Szreter. VerICS 2004: A model checker for real time and multi-agent systems. In *Proceedings of the International Workshop on Concurrency, Specification and Programming (CS&P'04)*, volume 170 of *Informatik-Berichte*, pages 88–99. Humboldt University, 2004.

[20] S. Otterloo, W. van der Hoek, and M. Wooldridge. Knowledge as strategic ability. *ENCTS*, 85(2):1–23, 2003.

[21] F. Raimondi and A. Lomuscio. MCMAS - A tool for verification of multi-agent systems. http://www.cs.ucl.ac.uk/staff/f.raimondi/MCMAS/.

[22] F. Raimondi and A. Lomuscio. Automatic verification of multi-agent systems by model checking via OBDDs. *Journal of Applied Logic*, 2005. To appear in Special issue on Logic-based agent verification.

[23] P. Y. Schobbens. Alternating-time logic with imperfect recall. In *International workshop on Logic and Communication in Multi-Agent Systems (LCMAS03)*, volume 85(2), pages 1–12, 2004.

[24] F. Somenzi. CUDD: CU decision diagram package - release 2.4.0. http:/ /vlsi.colorado.edu/ ∼fabio/CUDD/cuddIntro.html, 2005.

[25] M. Wooldridge, M. Fisher, M. Huget, and S. Parsons. Model checking multiagent systems with MABLE. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS-02)*, Bologna, Italy, July 2002.

Model Checking for Multivalued Logic of Knowledge and Time

Beata Konikowska and Wojciech Penczek *
Institute of Computer Science, PAS
Ordona 21, 01-237 Warsaw, Poland
email: {beatak,penczek}@ipipan.waw.pl

ABSTRACT

We present a multivalued μK-calculus, an expressive logic to specify knowledge and time in multi-agent systems. We show that the general method of translation [22] from multivalued to two-valued De Morgan algebras can be extended to mv μK-calculus model checking. This way can we reduce the model checking problem for mv μK-calculus to several instances of the model checking problem for two-valued μK-calculus. As a result, properties involving mv μK-calculus or its subsets, like mv CTLK or mv CTL*K, can be verified using any of the available model checking algorithms. Three simple examples are shown to exemplify possible applications of multivalued logics of knowledge and time.

Categories and Subject Descriptors

F.3.1 [**Specifying and Verifying and Reasoning about Programs**]: Specification techniques; D.2.4 [**Software/Program Verification**]: Model checking; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Verification

Keywords

Model checking, interpreted systems, epistemic logic, multivalued modal logic.

1. INTRODUCTION

The field of multi-agent systems (MAS) theories is traditionally concerned with formal representation of mental attitudes of autonomous entities, known as *agents*, in a distributed system. Over the past 20 years, several modal logics, involving operators referring to time, knowledge, beliefs, desires, goals, and intentions, have been developed with this motivation in mind. These logics are perceived as *specifications* of particular classes of MAS systems. Their aim is to offer a description of the *macroscopic* mental properties (such as knowledge, beliefs, etc.) that a MAS should exhibit in a specific class of scenarios. Model checking is a widespread, basic method used in automated verification of concurrent systems, like hardware circuits, communication protocols, and distributed programs [4]. It amounts to verifying that a finite state program P satisfies a property φ (denoted $P \models \varphi$). When P is represented by its model M_P and the property φ is given by a temporal logic formula, one checks whether $M_P \models \varphi$, whereby the process of generating M_P for P and checking that $M_P \models \varphi$ is automated.

Recently, the field of model checking has been extended to cover verification of multi-agent systems [30], which has become a topic of very active research. In particular, recent contributions [12, 13, 24] have focused on extending to such systems the model checking tools and techniques usually employed for verification of reactive systems. Essentially, these study the validity of a formula representing a property of interest in the model representing all the computations of the multiagent system (MAS) under consideration. Iterative approaches to model checking (i.e., those based on explicit enumeration of all the possible states of computation) are known to suffer from the state explosion problem. For this reason, methods based on symbolic representation are currently seen as the most promising ones. In particular, methods based on variants of the Binary Decision Diagrams (BDD's) [26], and satisfiability (SAT) checkers [25, 18] are a continuing focus of research. Verification via the BDD's involves translating the model checking problem into operations on boolean functions represented in a concise and canonical way, whereas SAT-checkers are used for testing satisfiability of the formulas which encode the model checking problems.

In addition to "classic", two-valued model checking within the framework of two-valued modal logics, increasing attention has recently been devoted to multi-valued model checking, based on various variants of multi-valued (mv) modal logics. This paradigm turns out to be very useful in a number of problems for which we need to reason under uncertainty or to employ models containing inconsistency. Uncertainty can occur either when complete information is not known or cannot be obtained , or when this information has been abstracted away [9]. On the other hand, models can be inconsistent as they combine conflicting points of view or

*Also affiliated with Institute of Informatics, Podlasie Academy, Siedlce. The author acknowledges support from Ministry of Science and Education (grant No. 3T11C01128).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

involve components developed by many designers [5].

The aim of this paper is to report on recent progress in applying multi-valued logic to verification of not only temporal, but also epistemic properties of a MAS. Our approach is novel in offering a model checking method for a newly defined, multi-valued μ-calculus combined with knowledge modalities (named mv μK-calculus). We continue here the line of research initiated in our former papers [22, 21], showing that the general method of translating the model checking problem for mv CTL* to the two-valued model checking problem for CTL* considered in those papers can be extended to modal mv μK-calculus.

First, we show that an injection f of a De Morgan algebra $\mathcal{L} = (L, \leq, -\!\!-)$ underlying the mv-logic in question into itself which preserves the bounds in the lattice and complement allows us to translate the model-checking problem for the mv μK-calculus over $\mathcal{L}$ into the model-checking problem for the mv μK-calculus over a simpler algebra $\mathcal{L}' = (L', \leq \cap (L' \times L'), -\!\!- \cap L')$, where $L' = f(L)$. As before, a repeated application of this method reduces an mv model checking problem for the mv μK-calculus to a set of problems we can already solve.

Second, we treat the case when the translation f mentioned above does not preserve complement. Then in order to compute the value of a formula in the target, $\mathcal{L}'$-based model M$'$ for the mv μK-calculus, we need both the values of $f(x)$ and $f(\overline{x})$ for $x \in L$. For this purpose, we use negated propositions and a two-component transition relation, with the second component storing negation of the first.

The rest of the paper is organized as follows. In the next section we discuss the related work. The mv μK-calculus is introduced in Section 3. An example of a coffee machine is presented in Section 4. Then, in Section 5 the general translation method is given. In the last section we give some final remarks.

2. RELATED WORK

Translations from multi-valued temporal logics to the standard model checking problem were defined by several authors. The main idea behind the reduction techniques appears in Fitting's work [7, 8] on the connection between 'multi-expert' and 'multi-valued' semantics for a propositional modal logic. Reduction methods for the special case of **3**-valued logics were discussed in many papers [10, 14, 15]. In our former paper [22], we showed a translation from negation-free mv-CTL* model checking to CTL* model checking for models over finite quasi-boolean lattices. The above translation was generalized to the full multi-valued modal μ-calculus in [11, 2].

On the other hand, there have been recently several new approaches to model checking of temporal logics of knowledge: CTLK [12, 23, 25, 17], CTLKD [31, 28, 27], TCTLKD [32],and an epistemic extension of ATL [19, 20, 16].

3. LOGIC OF KNOWLEDGE AND TIME

In this section we define syntax and semantics of a multi-valued modal logic of knowledge, called mv μK-calculus.

Syntax of mv μK-calculus

Let PV be a set of propositional variables, and Var — a set of fixed point variables.

We consider a positive normal form of mv μK-calculus (called mv μK$^+$), where negation is only applied to propositional variables. It is important that mv μK$^+$ has the same expressive power as mv μK. The language of mv μK$^+$ is defined by:

$$\varphi :: = p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{EX}\varphi \mid \mathrm{AX}\varphi \mid \mathrm{Z} \mid \mu Z.\varphi(Z) \mid \nu Z.\varphi(Z) \mid \mathrm{K}_i\varphi \mid \mathrm{D}_\Gamma\varphi \mid \mathrm{E}_\Gamma\varphi \mid \overline{\mathrm{K}}_i\varphi \mid \overline{\mathrm{D}}_\Gamma\varphi \mid \overline{\mathrm{E}}_\Gamma\varphi,$$

where p ranges over PV, Z — over Var, $i \in A$, and $\Gamma \subseteq A$, with A being a finite set of agents defined below. Here AX and EX represent the familiar temporal modalities "at the next step for all paths" and "at the next step for some path", while ν and μ are the greatest and the least fixed point modalities, respectively. When formulating properties in mv μK$^+$ we also use the following derived operators:

$$\mathrm{AG}\varphi = \nu Z.(\varphi \wedge \mathrm{AX}(Z)), \qquad \mathrm{EF}\varphi = \mu Z.(\varphi \vee \mathrm{EX}(Z)).$$

In turn, $\mathrm{K}_i, \mathrm{E}_\Gamma$ and D_Γ represent the well-known modalities of the logic of knowledge: K_i means "agent i knows", E_Γ — "everybody in group Γ knows" (everyone knows), and D_Γ — "somebody in group Γ knows" (distributed knowledge), with $\overline{\mathrm{K}}_i, \overline{\mathrm{E}}_\Gamma$ and $\overline{\mathrm{D}}_\Gamma$ being their duals. What is more, with help of ν and μ we can also define the more sophisticated group knowledge modality C_Γ (common knowledge), together with its dual $\overline{\mathrm{C}}_\Gamma$, as derived operators[1]

$$\mathrm{C}_\Gamma\varphi = \nu Z.(\mathrm{E}_\Gamma\varphi \wedge \mathrm{E}_\Gamma(Z)), \qquad \overline{\mathrm{C}}_\Gamma\varphi = \mu Z.(\overline{\mathrm{E}}_\Gamma\varphi \vee \overline{\mathrm{E}}_\Gamma(Z)).$$

Semantics for mv μK-calculus

We start with defining quasi-boolean lattices, which are used as logical domains of interpretation for the formulas of our multi-valued logics.

DEFINITION 1. *A* lattice *is a partial order $\mathcal{L} = (L, \leq)$, where, for any two elements $x, y \in L$, there exist their greatest lower bound (called* meet *and denoted by $x \cap y$) and their least upper bound (called* join *and denoted by $x \cup y$) with respect to the lattice order $\leq$.*[2]

In what follows, we consider **finite** lattices only, for which **F** (**T**) denotes the least (greatest, resp.) element.

DEFINITION 2. *A tuple $\mathcal{L} = (L, \leq, -\!\!-)$ is a* quasi-boolean algebra *if $(L, \leq)$ is a lattice and $-\!\!-$ is a unary operator (complement) having the following properties for any $x, y \in L$:*

- $\overline{x \cap y} = \overline{x} \cup \overline{y}$,
- $\overline{x \cup y} = \overline{x} \cap \overline{y}$,
- $x \leq y$ *iff* $\overline{y} \leq \overline{x}$,
- $\overline{\overline{x}} = x$.

$\mathcal{L}$ is distributive if the following two conditions hold for any $x, y, z \in L$:

- $z \cup (x \cap y) = (z \cup x) \cap (z \cup y)$,
- $z \cap (x \cup y) = (z \cap x) \cup (z \cap y)$.

[1] In the logic of knowledge, the usual practice is to introduce C_Γ as a primary operator, but the presence of the fixpoint operators in our logic allows us to derive it from the simpler knowledge modalities.

[2] Clearly, from antisymmetry of a partial order it follows that both the greatest lower bound and the least upper bound of x, y are uniquely determined elements of L.

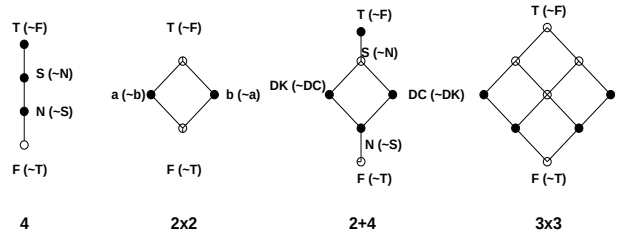


Figure 1: Examples of De Morgan algebras

A distributive, quasi-boolean algebra is called a De Morgan *algebra.*

EXAMPLE 1. *In Figure 3 we show a finite total order* **4** *representing uncertainty, and three well-known finite lattices, the use of which is motivated by clear practical intuitions: the lattices* $\mathbf{2} \times \mathbf{2}$ *and* $\mathbf{3} \times \mathbf{3}$ *representing disagreement, and the lattice* $\mathbf{2} + \mathbf{4}$ *representing both uncertainty and disagreement.*

The semantics of mv $\mu\mathrm{K}^+$ is based on the well-known notion of interpreted systems semantics [6], which can be shortly defined as follows. Assume a set $A = \{1, \ldots, n\}$ of agents, together with sets Loc_i and Act_i of local states and possible actions, respectively, for each agent $i \in A$, and sets Loc_e and Act_e of local states and actions for the environment. The set of global system states is defined as $S \subseteq Loc_1 \times \cdots \times Loc_n \times Loc_e$, where each $(l_1, \ldots, l_n, l_e) \in S$ represents a computational state of the whole system. Further assume a set of protocols $P_i : Loc_i \rightarrow 2^{Act_i}$, for $i = 1, \ldots, n$, representing the functional behaviour of every agent, and a function $P_e : Loc_e \rightarrow 2^{Act_e}$ for the environment. We can model the computation taking place in the system by means of a transition function $t : S \times Act \times S \longrightarrow L$, where $Act \subseteq Act_1 \times \cdots \times Act_n \times Act_E$ is the set of joint actions. We assume[3] $t(s, a, s') = t(s, a', s')$ for each $s, s' \in S$ and $a, a' \in Act$. Based on the above framework, we define the notion of a model underlying the semantics of mv $\mu\mathrm{K}^+$:

DEFINITION 3. *Given a set of agents* $A = \{1, \ldots, n\}$*, an mv modal epistemic* model *(or simply an mv model) is a tuple* $\mathrm{M} = (S, \mathcal{R}, \sim_1, \ldots, \sim_n, V, \mathcal{L})$*, where*

- S *is a finite set of global states of the system,*
- $\mathcal{R} : S \times S \rightarrow L \times L$ *is an mv transition relation on* S*,*
- $\sim_i \subseteq S \times S$ *(*$i \in A$*) is an* epistemic accessibility relation *for each agent* $i \in A$ *defined by* $s \sim_i s'$ *iff* $l_i(s') = l_i(s)$*, where* $l_i : S \longrightarrow Loc_i$ *extracts the local state of agent* i *from a global state* s*. Obviously,* $\sim_i$ *is an equivalence relation,*
- $V : S \times PV \rightarrow L$ *is a valuation function, for propositional variables* PV*,*
- $\mathcal{L} = (L, \leq, \overline{})$ *is a De Morgan algebra.*

Our definition of $\mathcal{R}$ is a bit more general than that of [11] as we assign two values to each transition. In what follows, by $\mathcal{R}_j(s,t)$, for $j = 1, 2$, we mean the j-th component of $\mathcal{R}(s,t)$. In the model corresponding to an interpreted system (defined above), we take $\mathcal{R}_1(s,s') = t(s,a,s')$ for some $a \in Act$. $\mathcal{R}_2$ [4] is used for storing the complement of $\mathcal{R}_1$. Next, we show a simple example of an mv model, which results from the standard two-valued model by using abstraction that collapses concrete states into a single abstract one. Typically, abstraction is used to overcome the state-explosion problem in classical model-checking

[3]This restriction allows to simplify the definition of $\mathcal{R}$ in a model.

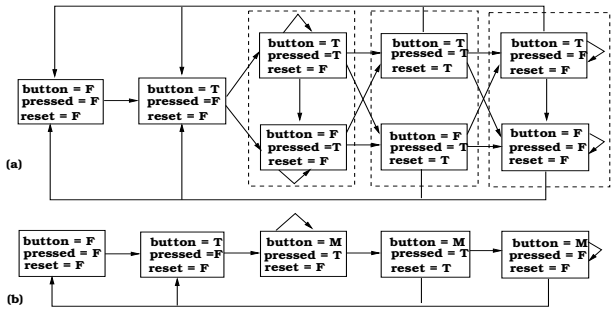


Figure 2: A model of the Button and the abstracted Button over the lattice 3

EXAMPLE 2. *The elevator model of Plath & Ryan consists of a single elevator which accepts requests made by agents (not shown in the figure) pressing buttons on the floor landings and from inside the elevator. The elevator moves up and down between floors and opens and closes its doors in response to these requests. The main module of the elevator is composed of several instances of the module Button (one per floor), which is shown in Figure 2 (a). Each of the states assigns values to the three propositional variables:* **button**, **pressed**, *and* **reset** *with the following meaning:* **button** $= T$ *- button available for pressing,* **pressed** $= F$*- button not latched, and* **reset** $= F$ *- request not fulfilled yet The arrows denote transitions between states. The transition relation is two-valued (so the transitions are not labeled) and corresponds to the possible changes between the states. The states indicated by the dashed lines (having symmetric transition relations) can be abstracted. This requires the use of a three-valued abstracted model (see Figure 2 (b)) in order to assign the value M (see Figure 5 (***3***)) to the variable* **button** *in the abstracted states.*

The semantics of mv $\mu\mathrm{K}^+$ is given by the function $[\cdot]^{\mathrm{M}}_{\rho}(s)$, which, for each formula φ of mv $\mu\mathrm{K}^+$, a model M, a state s in M, and a valuation of the fixed point variables $\rho : Var \longrightarrow L^S$, returns the value of φ at the state s of the model M for the valuation ρ, defined as follows:

- $[p]^M_\rho(s) = V(s,p)$ for $p \in PV$,
- $[Z]^M_\rho(s) = \rho(Z)(s)$ for $Z \in Var$,
- $[\neg p]^M_\rho(s) = \overline{[p]^M_\rho(s)}$,
- $[\varphi \vee \psi]^M_\rho(s) = [\varphi]^M_\rho(s) \cup [\psi]^M_\rho(s)$,
- $[\varphi \wedge \psi]^M_\rho(s) = [\varphi]^M_\rho(s) \cap [\psi]^M_\rho(s)$,
- $[\mathrm{EX}\varphi]^M_\rho(s) = \bigcup_{t \in S}(\mathcal{R}_1(s,t) \cap [\varphi]^M_\rho(t))$,

[4]We do not show $\mathcal{R}_2$ in the figures.

- $[\mathrm{AX}\varphi]^M_\rho(s) = \bigcap_{t\in S}(\overline{\mathcal{R}_1(s,t)} \cup [\varphi]^M_\rho(t))$,
- $[\nu Z.\varphi(Z)]^M_\rho(s) = \bigcup\{\mathcal{O} \in L^S \mid \mathcal{O} \leq' [\varphi]^M_{\rho[Z\to\mathcal{O}]}\}(s)$,
- $[\mu Z.\varphi(Z)]^M_\rho(s) = \bigcap\{\mathcal{O} \in L^S \mid [\varphi]^M_{\rho[Z\to\mathcal{O}]} \leq' \mathcal{O}\}(s)$, where $\rho[Z \to \mathcal{O}]$ coincides with ρ except for mapping Z to $\mathcal{O}$, and $\leq'$ is a pointwise extension of the corresponding operator of $\mathcal{L}$.
- $[\mathrm{K}_i\varphi]^M_\rho(s) = \bigcap_{\{s\in S|s\sim_i s'\}}[\varphi]^M_\rho(s')$,
- $[\mathrm{E}_\Gamma\varphi]^M_\rho(s) = \bigcap_{i\in\Gamma}[\mathrm{K}_i\varphi]^M_\rho(s)$
- $[\mathrm{D}_\Gamma\varphi]^M_\rho(s) = \bigcup_{i\in\Gamma}[\mathrm{K}_i\varphi]^M_\rho(s)$,
- $[\overline{\mathrm{K}}_i\varphi]^M_\rho(s) = \bigcup_{\{s\in S|s\sim_i s'\}}[\varphi]^M_\rho(s')$,
- $[\overline{\mathrm{E}}_\Gamma\varphi]^M_\rho(s) = \bigcup_{i\in\Gamma}[\overline{\mathrm{K}}_i\varphi]^M_\rho(s)$
- $[\overline{\mathrm{D}}_\Gamma\varphi]^M_\rho(s) = \bigcap_{i\in\Gamma}[\overline{\mathrm{K}}_i\varphi]^M_\rho(s)$,

where φ, ψ are formulas of mv $\mu\mathrm{K}^+$, $i \in A$ and $\Gamma \subseteq A$. Note that in fact we can get rid of negation (complement) completely. Indeed, negated propositions can be replaced by fresh propositions, following e.g. [1]. That is, for $p \in PV$ we can define $\overline{p}$ such that $V(s,\overline{p}) = \overline{V(s,p)}$[5]; the complements of the values of $\mathcal{R}_1$ can be represented by the second component of $\mathcal{R}$, set at $\mathcal{R}_2(s,t) = \overline{\mathcal{R}_1(s,t)}$; and $\overline{\mathcal{R}_1(s,t)}$ can be replaced by $\mathcal{R}_2(s,t)$ in the semantics of $\mathrm{AX}\varphi$. This way we can 'forget' about negation at the price of having to use two values for the transition relation. We show that the above 'trick' is superfluous not only when we do not translate formulas containing both universal and existential subformulas [11], but also when the translation preserves complement.

4. MOTIVATING EXAMPLES

Next, we show two examples. The first of them justifies the use of multiple logical values for expressing the knowledge possesed by agents, whereas the second one shows an mv model where the transition relation is also multivalued.

EXAMPLE 3. *Consider an agent B representing the bank, whose task is to decide if a customer C applying for a loan is to be granted that loan to not. For this purpose, the agent sends a query regarding the reliability of customer to a number of databases which contain information about bad debtors. From each database, B may obtain the following information:* **Yes**, *customer C is a bad debtor;* **No**, *customer C is not a bad debtor;* **Error** *- access denied or impossible. What is more, different databases can surely provide different responses — depending on whether customer C occurs in them or not. Clearly, in such a situation the two classical logical values are not enough, and the agent has to take into consideration all basic combinations of responses from the databases:* **F** *— no information on the customer;* **a** *— unanimous "***No***",* **b** *— unanimous "***Yes***", and* **T** *— inconsistent information from different databases (both "***No***" and "***Yes***" occur) Thus the natural logic for the agent would be a four-valued one, featuring the above values ordered into the lattice* $\mathbf{2}\times\mathbf{2}$ *shown in Figure 4. Let p_C represent the proposition "C is a bad debtor", and s — the agent's decision state. Then, given suitable formal details about the model* M[6]*, a resonable policy to be followed by the agent B is: Refuse the loan irrevocably if* $[K_B p_C]^{\mathrm{M}}(s) = \mathbf{b}$*, and grant it unconditionally if* $[K_B p_C]^{\mathrm{M}}(s) = \mathbf{a}$*. In turn, if* $[\overline{K}_B p_C]^{\mathrm{M}}(s) = \mathbf{F}$*, the agent could delay decision until getting some information from the databases, and if* $[\overline{K}_B p_C]^{\mathrm{M}}(s) = \mathbf{T}$ *(conflicting information about C's reliability) — demand a security for the credit from the customer.*

EXAMPLE 4. *In Figure 3 we show an example of a standard multi-valued model representing our requirements for a simple coffee machine (a modified example from [3]). The coffee machine can be operated by several agents. However, here we show one agent only, for simplicity and clarity sake. The model uses multiple truth values to distinguish between transitions and propositions that* **must**, **should**, **should not**, *or* **must not** *be true. Two types of unknown values are used:* **do not know** *for values controlled by the system, but not yet decided, and* **do not care** *for values controlled by the environment. Labelling a state s with $p = x$ means that $V(s,p) = x$, where p is a proposition and x is a truth value. Labelling a transition with x means that its truth value is x. In the figure we omit all the transitions labelled* **F**, *i.e., "prohibited". The coffee machine starts in the state* **OFF**, *where it is irrelevant whether a cup is provided or not, which is modeled by cup =* **DC**. *Since it has not been decided yet whether the machine should be able to go directly to* **IDLE**, *the corresponding transition is labelled* **DK**. *On the other hand, all the transitions labelled* **T** *represent the required changes of states. Note that the transitions from the states* **Delivers ...** *to* **OFF** *are labelled* **N**. *As they are dependent on the environment they cannot be prohibited, i.e., labelled with* **F**. *The value* **DK** *of the proposition power in the state* **IDLE** *represents the fact that it has not yet been decided whether the power should be ON or OFF on idle. Notice that the model abstracts away from the payments required from clients for simplicity sake.*

The transitions labelled with actions $BC, C + S, WC, WC + S, Ret$ are shared between the coffee machine and the agent. More precisely, a transition labelled with an action of the coffee machine can be only executed synchronously with a transition labelled with the same action of the drinking agent. Namely, the agent can select a type of coffee to be delivered by the machine and the machine returns to **READY** *only after the agent has finished drinking and moved back to the state* **HAPPY**.

There are several properties that can be specified and then checked in the model:

1. *The machine must eventually be able to deliver coffee,*
2. *Coffee is always delivered in a cup,*
3. *After coffee has been delivered, coffee cannot be immediately delivered again.*
4. *Always when coffee is delivered the agent knows that he is drinking,*
5. *Always when the agent is thirsty, he will eventually drink coffee,*
6. *Always when there is no coffee, the agent knows that he is not drinking.*

The properties of the coffee machine model can be specified by the following formulas:

[5]We consider only propositions that are used by the formula to be verified.

[6]We disregard ρ here for simplicity.

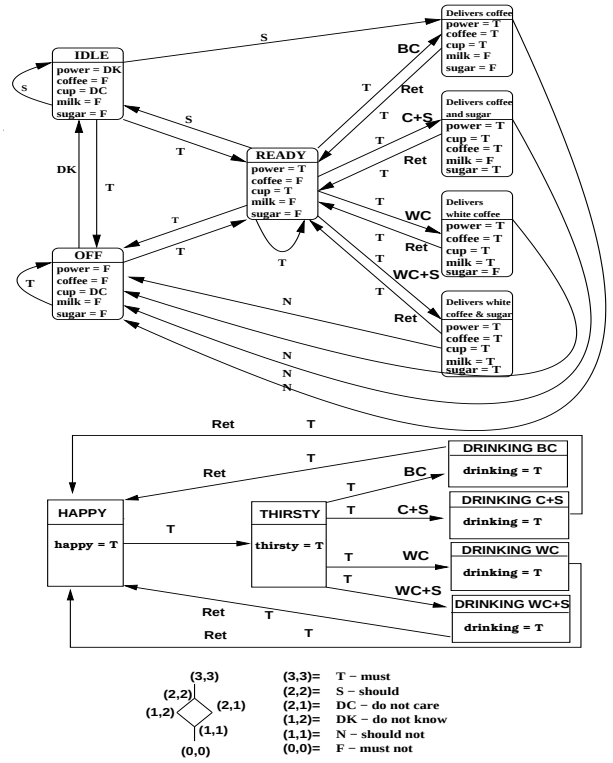


Figure 3: A coffee machine model with one drinking agent over the lattice $\mathbf{2 \times 2 + 2}$

1. EF*coffee*,
2. AG($\neg$*coffee* $\vee$ *cup*),
3. AG($\neg$*coffee* $\vee$ AX($\neg$*coffee*)).

The properties of the drinking agent model can be specified by the following formulas:

4. AG(*coffee* $\Rightarrow$ K$_1$*drinking*),
5. AG(*thirsty* $\Rightarrow$ EF *drinking*),
6. AG($\neg$ *coffee* $\Rightarrow$ K$_1\neg$*drinking*).

5. MODEL CHECKING FOR mv μK$^+$

Now we proceed with the basic results which will underlie all of the translations for the mv logic of knowledge and time with more than two logical values.

5.1 Translations preserving complement

THEOREM 1. *Let $\mathcal{L} = (L, \leq, \overline{})$ be an arbitrary quasi-boolean lattice* [7] *and let $f : L \to L$ be a function with $f(\mathbf{F}) = \mathbf{F}$ preserving complement and arbitrary bounds in $\mathcal{L}$, i.e.,*

$$f(\overline{x}) = \overline{f(x)}.$$
$$f(\bigcap_{i\in I} x_i) = \bigcap_{i\in I} f(x_i), \quad f(\bigcup_{i\in I} x_i) = \bigcup_{i\in I} f(x_i). \tag{1}$$

[7]Actually, the theorem holds for any lattice, but if $\mathcal{L}$ does not satisfy De Morgan laws, then the positive normal form mv μK$^+$ of mv μK we consider here is not equivalent to the full language mv μK allowing abritrary placement of negation.

where I is an arbitrary set of indices. Further, let $M = (S, \mathcal{R}, \sim_1, \ldots, \sim_n, V, \mathcal{L})$ be an L-valued model of mv μK^+, and let $M' = (S, \mathcal{R}', \sim_1, \ldots, \sim_n, V', f(\mathcal{L}))$ be a model[8] *with values in $f(L)$, obtained out of* M *by replacing each value $x \in L$ returned by V and $\mathcal{R}_1$ in* M *with $f(x)$, i.e., $\mathcal{R}'_1(s, s') = f(\mathcal{R}_1(s, s'))$ and $V'(s, p) = f(V(s, p))$, for $s, s' \in S$ and $p \in PV$. Then, for any formula φ of mv μK^+ over $\mathcal{L}$, any valuation ρ, any state s, and any $x \in f(L)$, we have*[9]

$$[\varphi]^{M'}_{f\rho}(s) = x \text{ iff } [\varphi]^M_\rho(s) \in f^{-1}(x) \tag{2}$$

The proof of the theorem is based on a key result for the extended language which we will define now. Assume the set of atomic formulas of our language mv μK$^+$ is extended by adding expressions of the form $\underline{\mathcal{R}_1}(\underline{s}, \underline{t})$ for $s, t \in S$, where, for any s, $\underline{s}$ is a constant representing the state s, and $\underline{\mathcal{R}_1}$ is a constant representing the first component of the mv-transition relation $\mathcal{R}$. Denote the resulting language by mv μK$_e$, and assume that the interpretation of mv μK$^+$ is accordingly extended to mv μK$_e$.

LEMMA 1. *Let a formula φ be such that $[\varphi]^M_\rho(s) = \bigcup_{i\in I}[\varphi_i]^M_\rho(g_i(s))$ or $[\varphi]^M_\rho(s) = \bigcap_{i\in I}[\varphi_i]^M_\rho(g_i(s))$ for any model M, any valuation ρ and any state s of M, where I is an arbitrary set, $\varphi_i \in$ mv μK_e, $g_i : S \to S$ for $i \in I$, and (2) holds for each φ_i. Then (2) holds for φ too.*

PROOF. We consider the case when $[\varphi]^M_\rho(s) = \bigcap_{i\in I}[\varphi_i]^M_\rho(g_i(s))$; proof of the other case is similar. As f preserves the bounds, by the assumption on φ we have $f([\varphi]^M_\rho(s)) = \bigcap_{i\in I} f([\varphi_i]^M_\rho((g_i(s)))$. Since each φ_i satisfies (2), this yields $f([\varphi]^M_\rho(s)) = \bigcap_{i\in I}[\varphi_i]^{M'}_{f\rho}(g_i(s)) = [\varphi]^{M'}_{f\rho}(s)$, whence $[\varphi]^{M'}_{f\rho}(s) = x$ iff $[\varphi]^M_\rho(s) \in f^{-1}(x)$, and (2) holds for φ. □

Now we can prove the theorem itself. To simplify notation, we drop the brackets in expressions denoting application of functions wherever possible. We use induction on the length of the formula. The hypothesis (2) clearly holds for propositional variables and their negations, as well as for $\underline{\mathcal{R}_1}(\underline{s}, \underline{t})$ and $\neg\underline{\mathcal{R}_1}(\underline{s}, \underline{t})$. It also holds for fixed point variables, since $f([Z]^M_\rho(s)) = f((\rho Z)(s)) = ((f\rho)(Z))(s) = [Z]^{M'}_{f\rho}(s)$. Next, we assume (2) holds for the formulas of length at most k, and prove it holds for any formula φ of length $k + 1$.

If $\psi = \varphi_1 \vee \varphi_2, \psi = \varphi_1 \wedge \varphi_2, \psi = \mathrm{EX}\varphi$, $\psi = \mathrm{AX}\varphi$, $\psi = \mathrm{K}_i\varphi$ for $i \in A$, where $\varphi, \varphi_1, \varphi_2$ are formulas of length at most k, then by the definition of semantics for $\vee, \wedge, \mathrm{EX}, \mathrm{AX}, \mathrm{K}_i, \overline{\mathrm{K}_i}$ the formula ψ satisfies the assumptions of Lemma 1, whence (2) holds for ψ by that lemma. Further, if $\psi \in \{\mathrm{D}_\Gamma\varphi, \mathrm{E}_\Gamma\varphi, \overline{\mathrm{D}_\Gamma}\varphi, \overline{\mathrm{E}_\Gamma}\varphi\}$, then by the definition of semantics for $\mathrm{D}_\Gamma, \mathrm{E}_\Gamma, \overline{\mathrm{D}_\Gamma}, \overline{\mathrm{E}_\Gamma}$ and the result obtained above for $\mathrm{K}_i\varphi, \overline{\mathrm{K}_i}\varphi$ the formula ψ satisfies the assumptions of the lemma too.

It remains to consider the case when $\psi \in \{\mu Z.\varphi(Z), \nu Z.\varphi(Z)\}$, where φ is of length at most k. We shall only give the proof for the operator μ, since the case of ν is just dual, and the case of $\mathrm{C}_\Gamma\varphi$ reduces to the latter. Let $[\varphi]^M_\rho$ denote the function which maps each $s \in S$ to $[\varphi]^M_\rho(s)$. First, note that (2) is equivalent to

$$f([\varphi]^M_\rho) = [\varphi]^{M'}_{f\rho} \tag{3}$$

[8]By $f(\mathcal{L})$ we denote the structure $(f(L), \leq \cap (f(L) \times f(L)), \overline{} \cap f(L))$.

[9]The composition of the functions f and ρ is denoted by $f\rho$ or by $f(\rho)$.

We shall assume the above holds for φ, and prove it also holds for $\psi = \mu Z.\varphi(Z)$.

By the fixed point properties of μ [29], we have

$$[\psi]^M_\rho = \bigcup_{i=0}^{\infty} [\varphi]^M_{\rho_i} \tag{4}$$

where $\rho_0 = \rho[Z \to \mathbf{FF}]$[10], $\rho_{i+1} = \rho_i[Z \to [\varphi]^M_{\rho_i}]$ for $i \geq 0$. Likewise,

$$[\psi]^{M'}_{f\rho} = \bigcup_{i=0}^{\infty} [\varphi]^{M'}_{\rho'_i} \tag{5}$$

where $\rho'_0 = (f\rho)[Z \to \mathbf{FF}], \rho'_{i+1} = \rho'_i[Z \to [\varphi]^{M'}_{\rho'_i}]$ for $i \geq 0$. We shall now prove by induction on i that

$$f\rho_i = \rho'_i,\ i \geq 0 \tag{6}$$

For $i = 0$, we have $f\rho_0 = f(\rho[Z \to \mathbf{FF}]) = (f\rho)[Z \to \mathbf{FF}] = \rho'_0$, since $f(\mathbf{F}) = \mathbf{F}$. Now assume the thesis holds for some $i \geq 0$. Then, $f\rho_{i+1} = f(\rho_i[Z \to [\varphi]^M_{\rho_i}]) = f\rho_i([Z \to [\varphi]^{M'}_{f\rho_i}]) = \rho'_i[Z \to [\varphi]^{M'}_{\rho'_i}] = \rho'_{i+1}$, where the second equality follows from the inductive assumption on φ, and the third - from the inductive assumption on ρ_i. This ends the proof of (6). As f preserves arbitrary bounds, from (4), the inductive assumption on φ, and finally (5) we get that $f([\psi]^M_\rho) =$

$$f(\bigcup_{i=0}^{\infty}([\varphi]^M_{\rho_i})) = \bigcup_{i=0}^{\infty} f([\varphi]^M_{\rho_i}) = \bigcup_{i=0}^{\infty}[\varphi]^{M'}_{f\rho_i} = \bigcup_{i=1}^{\infty}[\varphi]^{M'}_{\rho'_i} = [\psi]^{M'}_{f\rho}$$

Thus (3) holds for ψ too, which ends the proof.

First and foremost, the bounds and complement preservation conditions (1) imply clearly that $f(\mathcal{L}) = (f(L), \leq \cap\ (f(L) \times f(L)), \overline{} \cap f(L))$ is a De Morgan subalgebra of $\mathcal{L}$ - hence we can indeed consider a model M' taking values in $f(L)$. This theorem provides the basis for reducing the model-checking problem over an algebra $\mathcal{L}$ to the model-checking problems over its proper subalgebras of the form $f(\mathcal{L})$, where f is the mapping referred to in Theorem 1.

The general method defines k mappings (translations) $f_1, \ldots, f_k$ such that:

- $f_1(\mathcal{L}), \ldots, f_k(\mathcal{L})$ are algebras for which we already have model-checking algorithms,
- for any $x \in L$, there exist some $x_i \in f_i(L), i = 1, \ldots, k$, such that $\bigcap_{i=1}^{k} f_i^{-1}(x_i) = \{x\}$[11].

Indeed: in such a case, Theorem 1 implies that, for any formula φ of mv $\mu\mathrm{K}^+$, any model M over $\mathcal{L}$, any valuation ρ, and any state $s \in S$, $[\varphi]^M_\rho(s) = x$ iff $[\varphi]^{M_i}_{f_i\rho}(s) = x_i$ for all $i = 1, \ldots, k$, where each M_i is the model over $f_i(L)$ obtained out of M using the mapping (translation) f_i. Hence the model-checking problem for $\mathcal{L}$ reduces to the model-checking problems over $f_1(\mathcal{L}), \ldots, f_k(\mathcal{L})$.

Notice that so far we have not used the second component of $\mathcal{R}$, i.e., $\mathcal{R}_2$, which is not needed in the case where the translation preserves complement.

[10] $\mathbf{FF}$ denotes the function $S \longrightarrow \{\mathbf{F}\}$.

[11] Note that the condition imposed on the x_i's is satisfied by join-irreducible elements (see Definition 4). In that case, the condition is equivalent to Birkhoff's representation theorem for finite distributive lattices. To the authors' best knowledge, the above formulation is new.

5.2 Translations not preserving complement

When complement is not preserved by our translation f, we cannot directly reduce the validity over M to the validity over M' using single values for V and $\mathcal{R}$ only. The problem is that in order to compute $[\varphi]^{M'}_{f\rho}$ we need both the values $f(x)$ and $f(\overline{x})$ for $x = V(s,p)$ or $x = \mathcal{R}_1(s,s')$. Unfortunately, now $f(\overline{x})$ cannot be computed from $f(x)$. Therefore, to keep this information we need new names not only for the negated propositions, but also for the complement of the first component of the transition relation. The situation with the propositions is much easier to deal with as we can simply assign fresh propositions to the states. As far as the transition relation is concerned, there are several possibilities to deal with $\mathcal{R}_1(s,s')$. When translating to **2**, one could obviously get a standard 2-valued model by duplicating states (see [2] or [11] p. 275) of M, so that the values of $f(\mathcal{R}_1(s,s'))$ and $f(\overline{\mathcal{R}_1(s,s')})$ were available as corresponding to the transition between either the original states or their copies.

Here we suggest also another solution, which would not give us a standard model, but does not require duplicating states, and can be directly used by some model checkers which encode transition relation w.r.t. the formula to be verified. As we have already mentioned in the former section, the idea is to use the second component of $\mathcal{R}$ to store the complement of the first one in the model M, i.e., to have $\mathcal{R}_2(s,s') = \overline{\mathcal{R}_1(s,s')}$, and to replace $\overline{\mathcal{R}_1(s,s')}$ with $\mathcal{R}_2(s,s')$ in the semantics of $\mathrm{AX}\varphi$. Then the function f is applied to both the values, returning $\mathcal{R}'_1(s,s') = f(\mathcal{R}_1(s,s'))$ and $\mathcal{R}'_2(s,s') = f(\mathcal{R}_2(s,s'))$ in M'. In other words, we translate the complement of $\mathcal{R}_1(s,s')$ directly. In case of two-valued models, the transitions would be labelled by pairs $(\mathbf{F},\mathbf{F}), (\mathbf{F},\mathbf{T}), (\mathbf{T},\mathbf{F})$, and $(\mathbf{T},\mathbf{T})$, and depending on whether the value of an existential or an universal formula is computed, either the first or the second element of the pair would be used.

5.3 Translation to 2-valued algebra

In this section we show that the translation method based on join irreducible elements [11, 2] can be seen as a special instance of our general translation.

Definition 4. *An element $l \in L$ is called* join-irreducible *iff $l \neq \mathbf{F}$ and, for any $x, y \in L$, $l = x \cup y$ implies that either $l = x$ or $l = y$.*
The set of all the join-irreducible elements of a lattice $\mathcal{L}$ is denoted by $\mathcal{JI}(\mathcal{L})$.

In Figure 3, the join-irreducible elements in the lattices are marked with black dots.

It is well known that every element of a finite distributive lattice can be uniquely decomposed into the join of all the join-irreducible elements in its downard closure $\downarrow x = \{y \in L \mid y \leq x\}$ of x, i.e.,

$$x = \bigcup(\mathcal{JI}(\mathcal{L}) \cap \downarrow x) \tag{7}$$

Let S be a set and $\mathcal{L} = (L, \leq, \overline{})$ a De Morgan algebra. Denote by L^S the set of all the functions from S to L. Then, $(L^S, \leq', \overline{}')$ is a De Morgan algebra, where $\leq'$ and $\overline{}'$ are pointwise extensions of the corresponding operators of $\mathcal{L}$.

For $x \in L$ let $\uparrow x = \{y \in L \mid x \leq y\}$ denote the upward closure of x.

THEOREM 2. *Let $\mathcal{L}$ be a finite De Morgan algebra, and let $l \in \mathcal{JI}(\mathcal{L})$. Then, the function $f_l : L \longrightarrow \{\mathbf{T}, \mathbf{F}\}$ defined by $f_l(\uparrow l) = \mathbf{T}$ and $f_l(L \setminus \uparrow l) = \mathbf{F}$ preserves arbitrary bounds.*

PROOF. This proof uses two facts, namely that $l \in \mathcal{JI}(\mathcal{L})$ and that $\mathcal{L}$ is distributive. We should prove that

$$\text{(a) } f_l(\bigcup_{i\in I} x_i) = \bigcup_{i\in I} f_l(x_i), \qquad \text{(b) } f_l(\bigcap_{i\in I} x_i) = \bigcap_{i\in I} f_l(x_i)$$

(a): Obviously, $f_l(\bigcup_{i\in I} x_i) = \mathbf{T}$ implies $(\bigcup_{i\in I} x_i) \in \uparrow l$, whence $l = l \cap \bigcup_{i\in I} x_i$. As $\mathcal{L}$ is distributive, the latter implies $l = \bigcup_{i\in I}(l \cap x_i)$. Since $l \in \mathcal{JI}(\mathcal{L})$, one can easily see that this in turn yields $l \cap x_i = l$, i.e., $x_i \in \uparrow l$, for some $i \in I$, whence $f_l(x_i) = \mathbf{T}$, and in consequence $\bigcup_{i\in I} f_l(x_i) = \mathbf{T}$.

Conversely, if $\bigcup_{i\in I} f_l(x_i) = \mathbf{T}$, then $f_l(x_i) = \mathbf{T}$ for some $i \in I$. Thus $x_i \in \uparrow l$ for some $i \in I$, whence $\bigcup_{i\in I} x_i \in \uparrow l$ by the monotonicity of join and $f_l(\bigcup_{i\in I} x_i) = \mathbf{T}$, which proves (a).

(b): Clearly, $f_l(\bigcap_{i\in I} x_i) = \mathbf{T}$ iff $\bigcap_{i\in I} x_i \in \uparrow l$ iff $x_i \in \uparrow l$ for each $i \in I$ by the anti-monotonicity of join. This in turn holds iff $f_l(x_i) = \mathbf{T}$ for each $i \in I$ iff $\bigcap_{i\in I} f_l(x_i) = \mathbf{T}$. Thus (b) holds too. □

By (7), for each $x \in L$ we have $x = \bigcup(\mathcal{JI}(\mathcal{L}) \cap \downarrow x)$. Let M^l be the model obtained using the translation f_l. Therefore, according to Theorem 1 (provided f_l preserves complement)

$$[\varphi]^{M_l}_{f_l(\rho)}(s) = x \text{ iff } [\varphi]^M_\rho(s) \in f_l^{-1}(x)$$

whence

$$[\varphi]^{M_l}_{f_l(\rho)}(s) = \mathbf{T} \text{ iff } [\varphi]^M_\rho(s) \in \uparrow l$$

Thus,

$$[\varphi]^M_\rho(s) = \bigcup\{l \in \mathcal{JI}(\mathcal{L}) \mid [\varphi]^{M_l}_{f_l(\rho)}(s) = \mathbf{T}\} \qquad (8)$$

Now, we give a sufficient condition for f_l to preserve complement.

THEOREM 3. *Let f_l be the translation function defined as in Theorem 2 for $l \in \mathcal{JI}(\mathcal{L})$ satisfying the following condition: for each $x \in L$, $x \in \uparrow l$ iff $\overline{x} \in L \setminus \uparrow l$. Then, f_l preserves complement, i.e., $f_l(\overline{x}) = \overline{f_l(x)}$ for each $x \in L$.*

PROOF. By the definition of f_l, we have $f_l(\overline{x}) = \mathbf{T}$ iff $\overline{x} \in \uparrow l$. By the condition assumed, the latter holds iff $\overline{\overline{x}} \in L \setminus \uparrow l$. As $\overline{\overline{x}} = x$, this is true iff $f_l(x) = \mathbf{F}$, whence indeed $f_l(\overline{x}) = \overline{f_l(x)}$. □

EXAMPLE 5. *For three lattices and their join-irreducible elements l, we show which of the corresponding functions f_l preserve (Figure 4) and which do not preserve (Figure 5) complement. Notice that no translation from* **3** *preserves complement. This explains why all the existing translations require duplication of states in the translated models. For the lattice* **4** *only f_S preserves complement, for* $\mathbf{2} \times \mathbf{2}$ *both the functions f_a and f_b preserve complement, whereas for* $\mathbf{2} + \mathbf{4}$ *only f_{DK} and f_{DC} do so. The functions f_l for the remaining join-irreducible elements l of the above lattices do not preserve complement. To mark a translation graphically, we adopt the convention that the lattice elements lying above the extra line are superimposed on the lattice structure mapped to* **T**, *whereas the elements below the line are mapped to* **F**. *Notice that for the lattice* $\mathbf{2} + \mathbf{4}$ *two translations are shown in Figure 4.*

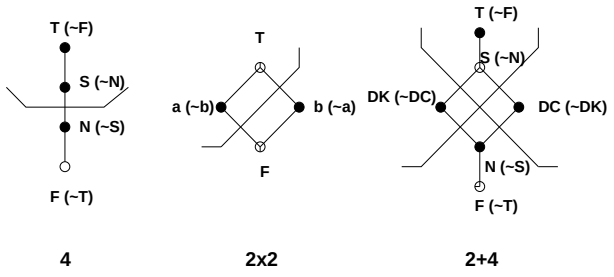


Figure 4: Translations preserving complement

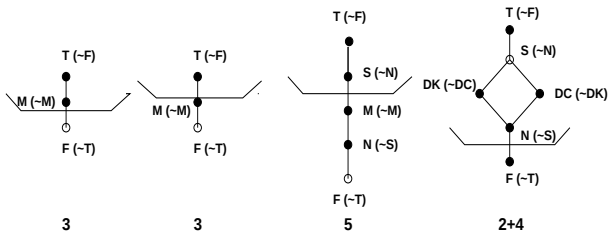


Figure 5: Translations not preserving complement

6. FINAL REMARKS

We have combined here three different logical frameworks: the logic of knowledge with interpreted system semantics, temporal logic and modal μ-calculus. Motivated by the cases of incomplete or inconsistent information occurring in real life applications, we have equipped the resulting cross-framework language in a logical domain being an arbitrary De Morgan lattice. We have obtained a new, unified mv logic of knowledge and time with a great expressive power, providing a universal framework for reasoning simultaneously about the phenomena related to knowledge and properties of concurrent systems in a multi-valued environment, and featuring additionally the fixpoint operators of μ-calculus. We have shown that the general method of model checking for mv temporal logic and mv μ-calculus via a translation to standard, two-valued models which preserves the lattice bounds, developed in our earlier papers, can be extended to the above unified logic, and is characterized by greater efficiency if the translation preserves complement in addition to the bounds. We have also given several examples illustrating the motivation for this type of framework and supporting practical applicability of our improvement.

7. REFERENCES

[1] G. Bruns and P. Godefroid. Generalized model checking: Reasoning about partial state spaces. In *Proc. of the 11th Int. Conf. on Concurrency Theory (CONCUR'00)*, volume 1877 of *LNCS*, pages 168–182. Springer-Verlag, 2000.

[2] G. Bruns and P. Godefroid. Model checking with multi-valued logics. In *ICALP*, pages 281–293, 2004.

[3] M. Chechik, B. Devereux, and S. Easterbrook. Implementing a multi-valued symbolic model checker. In *Proc. of the 7th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'01)*, volume 2031 of *LNCS*, pages 404–419.

Springer-Verlag, 2001.

[4] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[5] S. Easterbrook and M. Chechik. A framework for multi-valued reasoning over inconsistent viewpoints. In *Proc. of the 23rd Int. Conf. on Software Engineering (ICSE'01)*, pages 411–420. IEEE Computer Society, 2001.

[6] R. Fagin, J. Y. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, 1995.

[7] M. Fitting. Many-valued modal logics. *Fundamenta Informaticae*, 15(3-4):335–350, 1991.

[8] M. Fitting. Many-valued modal logics II. *Fundamenta Informaticae*, 17:55–73, 1992.

[9] P. Godefroid, M. Huth, and R. Jagadeesan. Abstraction-based model checking using modal transition systems. In *Proc. of the 12th Int. Conf. on Concurrency Theory (CONCUR'01)*, volume 2154 of *LNCS*, pages 426–440. Springer-Verlag, 2001.

[10] P. Godefroid and R. Jagadeesan. On the expressiveness of 3-valued models. In *Proc. of the 4th Int. Conf. on Verification, Model Checkig, and Abstract Interpretation (VMCAI'03)*, volume 2575 of *LNCS*, pages 206–222. Springer-Verlag, 2003.

[11] A. Gurfinkel and M. Chechik. Multi-valued model checking via classical model checking. In *Proc. of the 14th Int. Conf. on Concurrency Theory (CONCUR'03)*, volume 2761 of *LNCS*, pages 266–280. Springer-Verlag, 2003.

[12] W. Hoek and M. Wooldridge. Model checking knowledge and time. In *Proc. of the 9th Int. SPIN Workshop (SPIN'02)*, volume 2318 of *LNCS*, pages 95–111. Springer-Verlag, 2002.

[13] W. Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *Proc. of the 1st Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, volume III, pages 1167–1174. ACM, July 2002.

[14] M. Huth, R. Jagadeesan, and D. A. Schmidt. A domain equation for refinement of partial systems. *Mathematical Structures in Computer Science*, 14:469–505, 2004.

[15] M. Huth and S. Pradhan. Consistent partial model checking. In *Proc. of the Workshop on Domains VI*, volume 73 of *ENTCS*, pages 45–85. Elsevier, 2004.

[16] W. Jamroga and W. Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63(2-3):185–219, 2004.

[17] M. Kacprzak, A. Lomuscio, and W. Penczek. From bounded to unbounded model checking for temporal epistemic logic. *Fundamenta Informaticae*, 63(2-3):221–240, 2004.

[18] M. Kacprzak, A. Lomuscio, and W. Penczek. Verification of multi-agent systems via unbounded model checking. In *Proc. of the 3rd Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'04)*, volume II, pages 638–645. IEEE Computer Society, July 2004.

[19] M. Kacprzak and W. Penczek. Unbounded model checking for alternating-time temporal logic. In *Proc. of the 3rd Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'04)*, volume II, pages 646–653. IEEE Computer Society, July 2004.

[20] M. Kacprzak and W. Penczek. Fully symbolic unbounded model checking for alternating-time temporal logic. *Autonomous Agents and Multi-Agent Systems*, 11(1):69 – 89, 2005.

[21] B. Konikowska and W. Penczek. Model checking multi-valued modal μ-calculus revisited. In *Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'04)*, volume 170(2) of *Informatik-Berichte*, pages 307–318.

[22] B. Konikowska and W. Penczek. Model checking for multi-valued computation tree logics. In M. Fitting and E. Orłowska, editors, *Beyond Two: Theory and Applications of Multiple Valued Logic*, pages 193–210. Physica-Verlag, 2003.

[23] A. Lomuscio, T. Lasica, and W. Penczek. Bounded model checking for interpreted systems: Preliminary experimental results. In *Proc. of the 2nd NASA Workshop on Formal Approaches to Agent-Based Systems (FAABS'02)*, volume 2699 of *LNAI*, pages 115–125. Springer-Verlag, 2003.

[24] R. Meyden and N. V. Shilov. Model checking knowledge and time in systems with perfect recall. In *Proc. of the 19th Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'99)*, volume 1738 of *LNCS*, pages 432–445. Springer-Verlag, 1999.

[25] W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundamenta Informaticae*, 55(2):167–185, 2003.

[26] F. Raimondi and A. Lomuscio. A tool for specification and verification of epistemic properties of interpreted systems. In *Proc. of the Int. Workshop on Logic and Communication in Multi-Agent Systems (LCMAS'03)*, volume 85(2) of *ENTCS*. Elsevier, 2003.

[27] F. Raimondi and A. Lomuscio. Verification of multiagent systems via ordered binary decision diagrams: An algorithm and its implementation. In *Proc. of the 3rd Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'04)*, volume II, pages 630–637. IEEE Computer Society, July 2004.

[28] F. Raimondi and A. Lomuscio. Symbolic model checking of multi-agent systems using OBDDs. In *Proc. of the 3rd NASA Workshop on Formal Approaches to Agent-Based Systems (FAABS III)*, volume 3228 of *LNCS*, pages 213–221. Springer-Verlag, 2005.

[29] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[30] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons, 2002.

[31] B. Woźna, A. Lomuscio, and W. Penczek. Bounded model checking for deontic interpreted systems. *Electr. Notes Theor. Comput. Sci.*, 126:93–114, 2005.

[32] B. Woźna, A. Lomuscio, and W. Penczek. Bounded model checking for knowledge and real time. In *'Proc. of the 4th Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'05)*, pages 165–172. IEEE Computer Society, 2005.

Temporal Qualitative Coalitional Games

Thomas Ågotnes
Department of Informatics, University of Bergen
PB. 7800, N-5020 Bergen, Norway
agotnes@ii.uib.no

Wiebe van der Hoek
Department of Computer Science, University of Liverpool
Liverpool L69 7ZF, UK
wiebe@csc.liv.ac.uk

Michael Wooldridge
Department of Computer Science, University of Liverpool
Liverpool L69 7ZF, UK
mjw@csc.liv.ac.uk

ABSTRACT

Qualitative Coalitional Games (QCGs) are a version of coalitional games in which an agent's desires are represented as goals which are either satisfied or unsatisfied, and each choice available to a coalition is a set of goals, which would be jointly satisfied if the coalition made that choice. A coalition in a QCG will typically form in order to bring about a set of goals that will satisfy all members of the coalition. In this paper, we introduce and study *Temporal QCGs* (TQCGs), i.e., games in which a sequence of QCGs is played. In order to represent and reason about such games, we introduce a linear time temporal logic of QCGs, known as $\mathcal{L}(TQCG)$. We give a complete axiomatization of $\mathcal{L}(TQCG)$, use it to investigate the properties of TQCGs in a small example, identify its expressive power, establish its complexity, characterise classes of TQGCs with formulas from our logical language, and formulate several (temporal) solution concepts for TQCGs.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; I.2.4 [**Knowledge representation formalisms and methods**]

General Terms

Theory

Keywords

Coalitional games, repeated games, logic

1. INTRODUCTION

The study of repeated games now forms a major component of the game theory literature [9, pp.133–161]. Perhaps the best-known example of such a repeated game is the iterated prisoner's dilemma, which has for example been studied both analytically [3, pp.353-358] and by means of competitions [2].

Given the role of game theory as a theoretical underpinning to the multi-agent systems field [11], it seems that repeated games are of particular importance to the field. By-and-large, we are not interested in building multi-agent systems that will operate in a "one-shot" fashion: we typically want them to operate over time, often without a pre-defined termination time. Moreover, given the important role that *coalitional games* play in multi-agent systems [12, 13], it seems that repeated coalitional games are also likely to be of significance. However, comparatively little research has considered repeated coalitional games, or coalitional games played over time [7].

Our aim in this paper is to study an iterated version of *Qualitative Coalitional Games* (QCGs) [15], a variation of coalitional games in which agent's desires are represented as goals which are either satisfied or unsatisfied, and each choice available to a coalition is a set of goals, which would be jointly satisfied if the coalition made that choice. A coalition in a QCG will typically form in order to bring about a set of goals that will satisfy all members of the coalition. In this paper, we introduce and study *Temporal QCGs* (TQCGs): games in which QCGs are played repeatedly. In order to represent and reason about such games, we introduce $\mathcal{L}(TQCG)$, a linear time temporal logic of QCGs. We give a complete axiomatization of $\mathcal{L}(TQCG)$, demonstrate its expressive power with respect to a type of simulation between TQCGs, establish the computational complexity of satisfiability for TQCGs, investigate the properties of TQCGs by characterising them as formulae in $\mathcal{L}(TQCG)$ and finally characterise some solution concepts of TQCGs in $\mathcal{L}(TQCG)$.

We begin, in the following section, with a short introduction to QCGs. As part of this introduction, in section 2.1 we define a logic for expressing properties of individual QCGs: this logic will serve as the "state language" or "assertion language" for the temporal QCG logic. In section 3, we introduce TQCGs. We begin with a short informal motivation, then give the temporal language for expressing properties of TQCGs, present an example to illustrate the idea of TQCGs and the role of the temporal language, investigate the expressive power of the language by means of a simulation relation between TQCG structures, give a complete axiomatization of the temporal language, and then investigate the axiomatic characterisation of various classes of TQCGs.

2. QUALITATIVE COALITIONAL GAMES

We give a brief introduction to Qualitative Coalitional Games (QCGs): details may be found in [15]. A QCG contains a (non-empty, finite) set $\mathcal{A} = \{1, \ldots, m\}$ of *agents*. Each agent $i \in \mathcal{A}$ is assumed to have associated with it a (finite) set $\mathcal{G}_i$ of *goals*, drawn from a set of overall possible goals $\mathcal{G}$. The intended interpretation is that the members of $\mathcal{G}_i$ represent all the individual rational outcomes for i – intuitively, the outcomes that give it "better than zero utility". That is, agent i would be happy if *any* member of $\mathcal{G}_i$ were

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

achieved – then it has "gained something". But, in QCGs, we are not concerned with preferences over individual goals. Thus, at this level of modelling, i is *indifferent* among the members of $\mathcal{G}_i$: it will be *satisfied* if *at least one* member of $\mathcal{G}_i$ is achieved, and *unsatisfied* otherwise. Note that cases where more than one of an agent's goals are satisfied are not an issue – an agent's aim will simply be to ensure that at least one of its goals is achieved, and there is no sense of an agent i attempting to satisfy as many members of $\mathcal{G}_i$ as possible.

A *coalition*, typically denoted by C, is simply a set of agents, i.e., a subset of $\mathcal{A}$. The *grand coalition* is the set of all agents, $\mathcal{A}$. We assume that each possible coalition has available to it a set of possible *choices*, where each choice intuitively characterises the outcome of one way that the coalition could cooperate. We model the choices available to coalitions via a *characteristic function* with the signature $\mathcal{V} : 2^{\mathcal{A}} \rightarrow 2^{2^{\mathcal{G}}}$. Thus, in saying that $G \in \mathcal{V}(C)$ for some coalition $C \subseteq \mathcal{A}$, we are saying that one choice available to the coalition C is to bring about *exactly* the goals in G. At this point, the reader might expect to see some constraints placed on characteristic functions. For example, at first sight the following *monotonicity* constraint might seem natural: $C \subseteq C'$ implies $\mathcal{V}(C) \subseteq \mathcal{V}(C')$. Although such a constraint is entirely appropriate for many scenarios, there are cases where such a constraint is not appropriate[1].

Bringing these components together, a qualitative coalitional game (QCG) is a tuple:

$$\Gamma = \langle \mathcal{A}, \mathcal{G}, \mathcal{G}_1, \ldots, \mathcal{G}_n, \mathcal{V} \rangle \text{ where}$$

- $\mathcal{A}$ is a finite, non-empty set of *agents*;
- $\mathcal{G}$ is a finite, non-empty set of possible *goals*;
- $\mathcal{G}_i \subseteq \mathcal{G}$ is the set of goals for agent $i \in \mathcal{A}$; and
- $\mathcal{V} : 2^{\mathcal{A}} \rightarrow 2^{2^{\mathcal{G}}}$ is the characteristic function of the game.

EXAMPLE 1. *Let Γ_1 be the following QCG for a collection of agents and a collection of goals $\{g_1, \ldots\}$. Agent 1 is satisfied with g_1 and g_4, and agent 2 is satisfied with g_2 and g_3. The characteristic function, where C_1, C_2, C_3, C_4 are different coalitions:*
$\mathcal{V}(C_1) = \{ \{g_1, g_2\} \}$ $\mathcal{V}(C_2) = \{ \{g_2, g_3\}, \{g_1\} \}$
$\mathcal{V}(C_3) = \{ \{g_5, g_6\} \}$ $\mathcal{V}(C_4) = \{ \{g_2, g_3\}, \{g_1\}, \{g_4\} \}$
We will make use of Γ_1 in later examples.

2.1 A Logic for QCGs

A logic for expressing properties of individual QCGs has not been formalised before. We now introduce such a logic. This logic will later be used as the assertion language, or state language, for the temporal logic we develop in section 3. The language is defined in two parts: $\mathcal{L}_c$ is the *satisfaction language*, and is used to express properties of choices made by agents. The basic constructs in this language are of the form sat_i, meaning "agent i is satisfied". The overall language $\mathcal{L}(QCG)$ is used for expressing properties of QCGs themselves. The main construct in this language is of the form $\langle C \rangle \varphi$, where φ is a formula of the satisfaction language, and means that C have a choice such that this choice makes φ true. For example, $\langle 3 \rangle (sat_1 \wedge sat_4)$ will mean that 3 has a choice that simultaneously satisfies agents 1 and 4.

Formally, the grammar φ_c defines the *satisfaction language* $\mathcal{L}_c$, while φ_q defines the *QCG language* $\mathcal{L}(QCG)$.

$$\begin{array}{lcl} \varphi_c & ::= & sat_i \mid \neg\varphi_c \mid \varphi_c \vee \varphi_c \\ \varphi_q & ::= & \langle C \rangle \varphi_c \mid \neg\varphi_q \mid \varphi_q \vee \varphi_q \end{array}$$

[1]For example, consider a legal scenario in which certain coalitions are forbidden by monopoly or anti-trust laws.

where $i \in \mathcal{A}$ and $C \subseteq \mathcal{A}$. (We note some similarities between our logical language $\mathcal{L}(QCG)$ and Pauly's language for axiomatizing judgement aggregation procedures [10], although the motivation and use of the languages are quite different.)

We use the usual derived propositional connectives ($\wedge$, $\rightarrow$, $\leftrightarrow$) for both languages $\mathcal{L}_c$ and $\mathcal{L}(QCG)$, and in addition write $[C]\varphi$ to abbreviate $\neg\langle C \rangle \neg\varphi$. The formula $[C]\varphi$ will be defined to be true exactly when φ is a *necessary* consequence of the coalition C making a choice; φ will be true no matter *which* choice the coalition makes. When $C = \{a\}$ is a singleton, we sometimes write $\langle a \rangle$ and $[a]$ for $\langle C \rangle$ and $[C]$.

When $\Gamma = \langle \mathcal{A}, \mathcal{G}, \mathcal{G}_1, \ldots, \mathcal{G}_n, \mathcal{V} \rangle$ is a QCG, $G \subseteq \mathcal{G}$ and $\varphi \in \mathcal{L}_c$, $\Gamma, G \models_Q \varphi$ is defined as follows:

$\Gamma, G \models_Q sat_i$ iff $\mathcal{G}_i \cap G \neq \emptyset$
$\Gamma, G \models_Q \neg\psi$ iff not $\Gamma, G \models_Q \psi$
$\Gamma, G \models_Q \psi_1 \vee \psi_2$ iff $\Gamma, G \models_Q \psi_1$ or $\Gamma, G \models_Q \psi_2$

When $\Gamma = \langle \mathcal{A}, \mathcal{G}, \mathcal{G}_1, \ldots, \mathcal{G}_n, \mathcal{V} \rangle$ is a QCG and φ is a $\mathcal{L}(QCG)$ formula, $\Gamma \models_Q \varphi$ is defined as follows:

$\Gamma \models_Q \langle C \rangle \psi$ iff there is a $G \in \mathcal{V}(C)$ such that $\Gamma, G \models_Q \psi$
$\Gamma \models_Q \neg\psi$ iff not $\Gamma \models_Q \psi$
$\Gamma \models_Q \psi_1 \vee \psi_2$ iff $\Gamma \models_Q \psi_1$ or $\Gamma \models_Q \psi_2$

EXAMPLE 2. *Let Γ_1 be as in Example 1. Then:*

$\Gamma_1 \models_Q \langle C_1 \rangle (sat_1 \wedge sat_2)$
$\Gamma_1 \models_Q (\langle C_2 \rangle sat_1 \wedge \langle C_2 \rangle sat_2) \wedge \neg(\langle C_2 \rangle (sat_1 \wedge sat_2))$
$\Gamma_1 \models_Q \neg(\langle C_3 \rangle sat_1 \vee \langle C_3 \rangle sat_2)$

Summarising, the satisfaction of agents is evaluated against a set of goals, while Boolean combinations of expressions referring to choices of coalitions are evaluated on a QCG Game Γ. The latter combinations will be the atomic assertions in our temporal framework of Section 3.

2.2 Expressive Power of L(QCG)

We look at the properties of QCGs which are definable in our language. It is clear from our language definition that what $\mathcal{L}(QCG)$ can express is which coalition can satisfy which set of agents concurrently. Note that we are not interested in *how* the coalitions make certain sets of agents satisfied, nor *why* an agent is satisfied (i.e., which goal satisfied him). We will now demonstrate that the properties of QCGs we can express in the language $\mathcal{L}(QCG)$ are exactly the properties closed under a notion of *QCG-simulation*. In other words, the language can not differentiate two games Γ and Γ' iff they *QCG-simulate* each other.

Obviously, equivalence of models transcends mere isomorphism. In particular, the semantics of performing a *choice* seem to depend only on which agents are satisfied by the choice. For example, one could imagine a mapping between "equivalent" goals of two models, maybe collapsing two goals of one model into one goal of the other. However, such a relation between models does not capture all instances of equivalent models. What is needed is a relation between *sets* of goals. This motivates the following definition of a QCG-simulation as a *relation* between two models. It is only necessary to relate goals which can actually be chosen by some coalition. Furthermore, it only makes sense to relate models which are defined over the same set of agents.

A relation

$$Z \subseteq \bigcup_{C \subseteq \mathcal{A}} (\mathcal{V}(C) \times \mathcal{V}'(C))$$

is a *QCG-simulation* between two QCGs $\Gamma = \langle \mathcal{A}, \mathcal{G}, \mathcal{G}_1, \ldots, \mathcal{G}_n, \mathcal{V} \rangle$ and $\Gamma' = \langle \mathcal{A}, \mathcal{G}', \mathcal{G}'_1, \ldots, \mathcal{G}'_n, \mathcal{V}' \rangle$ iff the following conditions hold for all coalitions C.

1. If GZG' then $G \cap \mathcal{G}_i = \emptyset$ iff $G' \cap \mathcal{G}'_i = \emptyset$, for all i (the *satisfaction condition*)
2. For every $G \in \mathcal{V}(C)$ there is a $G' \in \mathcal{V}'(C)$ such that GZG' (Z is *total*)
3. For every $G' \in \mathcal{V}'$ there is a $G \in V(C)$ such that GZG' (Z is *surjective*)

If there exist a QCG-simulation between two games Γ and Γ', we write $\Gamma \rightleftharpoons \Gamma'$. If $\Gamma \rightleftharpoons \Gamma'$, we can simulate any choice in one model with a choice in the other, and vice versa. This notion of simulation is somewhat similar to the notion of "alternating simulation" between alternating transition systems in [1].

EXAMPLE 3. *Let Γ_2 be the QCG with the same agents as in Γ_1 (Example 1), goals $f_1, f_2, \ldots$ such that agent 1 is satisfied in f_1 and f_3 and agent 2 is satisfied in f_2, f_3 and f_4, and the following characteristic function:*

$$\mathcal{V}(C_1) = \{\ \{f_3\}\ \} \quad \mathcal{V}(C_2) = \{\ \{f_2\}, \{f_1\}\ \}$$
$$\mathcal{V}(C_3) = \{\ \{f_5\}\ \} \quad \mathcal{V}(C_4) = \{\ \{f_1\}, \{f_2\}, \{f_4\}\ \}$$

Then $\Gamma_1 \rightleftharpoons \Gamma_2$. The relation Z consisting of the following pairs is a QCG-simulation between Γ_1 and Γ_2.

$$\begin{array}{ccc} \langle\{g_1, g_2\}, \{f_3\}\rangle & \langle\{g_2, g_3\}, \{f_2\}\rangle & \langle\{g_1\}, \{f_1\}\rangle \\ \langle\{g_5, g_6\}, \{f_5\}\rangle & \langle\{g_2, g_3\}, \{f_2\}\rangle & \langle\{g_2, g_3\}, \{f_4\}\rangle \\ \langle\{g_1\}, \{f_1\}\rangle & \langle\{g_4\}, \{f_1\}\rangle & \end{array}$$

Note that Z is not a function, nor the inverse of a function.

We write $\Gamma \equiv \Gamma'$ iff $\forall_{\varphi \in \mathcal{L}(QCG)}[\Gamma \models_Q \varphi \Leftrightarrow \Gamma' \models_Q \varphi]$.

THEOREM 1. *Satisfaction is invariant under QCG-simulation:*

$$\Gamma \rightleftharpoons \Gamma' \quad \Rightarrow \quad \Gamma \equiv \Gamma'$$

PROOF. Let $\Gamma = \langle \mathcal{A}, \mathcal{G}, \mathcal{G}_1, \ldots, \mathcal{G}_n, \mathcal{V}\rangle$ and $\Gamma' = \langle \mathcal{A}, \mathcal{G}', \mathcal{G}'_1, \ldots, \mathcal{G}'_n, \mathcal{V}'\rangle$ with $\Gamma \rightleftharpoons \Gamma'$. First, we show that

$$GZG' \Rightarrow (\Gamma, G \models_Q \psi \Leftrightarrow \Gamma', G' \models_Q \psi) \qquad (1)$$

for any ψ by induction over ψ. For the base case, let $\psi = sat_i$. $\Gamma, G \models_Q \psi$ iff $\mathcal{G}_i \cap G \neq \emptyset$ iff, by the satisfaction condition, $\mathcal{G}'_i \cap G' \neq \emptyset$ iff $\Gamma', G' \models_Q \psi$. The inductive step (negation and disjunction) is straightforward. We now show that

$$\Gamma \models_Q \varphi \Leftrightarrow \Gamma' \models_Q \varphi$$

for any φ by induction on φ. For the base case, let $\varphi = \langle C\rangle\psi$. For the direction to the right, if $\Gamma \models_Q \varphi$ then there is a $G \in \mathcal{V}(C)$ such that $\Gamma, G \models_Q \psi$. By totality of Z, there is a $G' \in \mathcal{V}'(C)$ such that GZG'. By (1), $\Gamma', G' \models_Q \psi$, and thus $\Gamma' \models_Q \varphi$. The direction to the left is symmetric: if $\Gamma' \models_Q \varphi$ there is a $G' \in \mathcal{V}'(C)$ such that $\Gamma', G' \models_Q \psi$; by surjectivity of Z there is a $G \in \mathcal{V}(C)$ such that GZG'; and by (1) $\Gamma, G \models_Q \psi$ and thus $\Gamma \models_Q \varphi$. The inductive step (negation and disjunction) is straightforward. □

The obvious question now is whether every pair of equivalent models are connected by a QCG-simulation. The answer is "yes".

THEOREM 2. *Let Γ, Γ' be defined over the same set of agents:*

$$\Gamma \rightleftharpoons \Gamma' \quad \Leftarrow \quad \Gamma \equiv \Gamma'$$

PROOF. Let $\Gamma = \langle \mathcal{A}, \mathcal{G}, \mathcal{G}_1, \ldots, \mathcal{G}_n, \mathcal{V}\rangle$ and $\Gamma' = \langle \mathcal{A}, \mathcal{G}', \mathcal{G}'_1, \ldots, \mathcal{G}'_n, \mathcal{V}'\rangle$ with $\Gamma \equiv \Gamma'$. With any coalition C and any choice $G \in \mathcal{V}(C)$, associate the set $S^C_G = \{i : G \cap \mathcal{G}_i \neq \emptyset\}$ of agents satisfied if C chooses G. Similarly for Γ': $T^C_{G'} = \{i : G' \cap \mathcal{G}'_i \neq \emptyset\}$ for any $G' \in \mathcal{V}'(C)$.

We define a QCG-simulation $Z : \Gamma \rightleftharpoons \Gamma'$ as follows: for every coalition C and pair of choices $G \in \mathcal{V}(C)$, $H \in \mathcal{V}'(C)$,

$$GZH \Leftrightarrow S^C_G = T^C_H$$

We must show that Z is total, i.e., that if $G \in \mathcal{V}(C)$, then there is a $H \in \mathcal{V}'(C)$ such that $S^C_G = T^C_H$. Suppose not: assume that $i \in S^C_G$ and $i \notin T^C_H$ for all $H \in \mathcal{V}'(C)$ (the argument is similar when $i \notin S^C_G$ and $i \in T^C_H$ for some $H \in \mathcal{V}'(C)$). Then $\Gamma \models_Q \langle C\rangle sat_i$ and $\Gamma' \models_Q \neg\langle C\rangle sat_i$, which contradicts the fact that $\Gamma \equiv \Gamma'$.

Similarly, we must show that Z is surjective, i.e., that if $H \in \mathcal{V}'(C)$, then there is a $G \in \mathcal{V}(C)$ such that $S^C_G = T^C_H$. Suppose not: assume that $i \in T^C_H$ and $i \notin S^C_G$ for all $G \in \mathcal{V}(C)$ (the argument is similar when $i \notin T^C_H$ and $i \in S^C_G$ for some $G \in \mathcal{V}(C)$). Then $\Gamma' \models_Q \langle C\rangle sat_i$ and $\Gamma \models_Q \neg\langle C\rangle sat_i$, which contradicts the fact that $\Gamma \equiv \Gamma'$.

Finally, we show that the satisfaction condition holds. If GZH, then $G \cap \mathcal{G}_i \neq \emptyset$ iff $i \in S^C_G$ iff, by the definition of Z, $i \in T^C_H$ iff $H \cap \mathcal{G}'_i \neq \emptyset$. □

2.3 Axiomatisation for QCGs

We define a Hilbert style axiomatisation of qualitative coalitional games, and prove its soundness and completeness. We name our axiomatisation for QCGs $\mathbf{K}(QCG)$. This name emphasises the close resemblance to the modal system $\mathbf{K}$, which also indicates that our logic, is in a sense, a weakest basic system for QCGs, to which more sophisticated constraints can easily be added — such extensions are the topic of Section 4. The system $\mathbf{K}(QCG)$ over the language $\mathcal{L}(QCG)$ is defined as follows, where φ, ψ are arbitrary $\mathcal{L}(QCG)$ formulae, α, β are arbitrary $\mathcal{L}_c$ formulae and C an arbitrary coalition:

$Prop^-$	If φ is an $\mathcal{L}(QCG)$-instance of a propositional tautology, then φ is provable
K^-	$[C](\alpha \to \beta) \to ([C]\alpha \to [C]\beta)$ is provable
MP^-	If $\varphi, \varphi \to \psi$ are provable, then ψ is provable
Nec^-	If α is an ($\mathcal{L}_c$) instance of a propositional tautology, then $[C]\alpha$ is provable

It is easy to see that the deduction theorem holds for $\mathbf{K}(QCG)$.

We will need the following properties of $\mathbf{K}(QCG)$. The proofs are straightforward for readers familiar with modal logic.

LEMMA 1. *Let $\alpha, \beta \in \mathcal{L}_c$:*

1. $\vdash_{\mathbf{K}(QCG)} \langle C\rangle(\alpha \wedge \beta) \to \langle C\rangle\alpha$
2. $\vdash_{\mathbf{K}(QCG)} \langle C\rangle(\alpha \vee \beta) \to (\langle C\rangle\alpha \vee \langle C\rangle\beta)$
3. $\vdash_{\mathbf{K}(QCG)} (\langle C\rangle\alpha \wedge [C](\alpha \to \beta)) \to \langle C\rangle\beta$

THEOREM 3 (SOUNDNESS & COMPLETENESS). *For any $\Phi \subseteq \mathcal{L}(QCG)$, $\varphi \in \mathcal{L}(QCG)$: $\Phi \models_Q \varphi \Leftrightarrow \Phi \vdash_{\mathbf{K}(QCG)} \varphi$*

PROOF. For soundness (the direction to the left), it is easy to see that the axioms are valid, and that the rules preserve logical consequence.

For completeness, let $\Psi \subseteq \mathcal{L}(QCG)$ be $\mathbf{K}(QCG)$ consistent. We show that Ψ is satisfied by some QCG. Let $\mathcal{A}$ be the set of agents and let $n = |\mathcal{A}|$. Let Δ be a $\mathcal{L}(QCG)$ maximal and $\mathbf{K}(QCG)$ consistent set containing Ψ (the proof of existence of such a set is the standard proof of Lindenbaum's lemma). We now construct $\Gamma = \langle \mathcal{A}, \mathcal{G}, \mathcal{G}_1, \ldots, \mathcal{G}_n, \mathcal{V}\rangle$, intended to satisfy Ψ, as follows:

- $\mathcal{G} = \{sat_1, \ldots, sat_n\}$
- $\mathcal{G}_i = \{sat_i\}$, for each i

- $X \in \mathcal{V}(C) \Leftrightarrow \langle C\rangle\xi_X \in \Delta$, for any $X \subseteq \mathcal{G}$, where

$$\xi_X \equiv \bigwedge_{sat_i \in X} sat_i \wedge \bigwedge_{i \in \mathcal{A}, sat_i \notin X} \neg sat_i$$

We show that

$$Gamma \models_Q \gamma \Leftrightarrow \gamma \in \Delta$$

for any γ by structural induction over γ. For the base case, $\gamma = \langle C\rangle\alpha$ for some $\alpha \in \mathcal{L}_c$. Again, we use induction on the structure of α. For the (nested) base case, let $\alpha = sat_i$. For the direction to the right, if $\Gamma \models_Q \gamma$ then there is an $X \in \mathcal{V}(C)$ such that $\Gamma, X \models_Q \alpha$, i.e., there is an $X \subseteq \mathcal{G}$ such that $\langle C\rangle\xi_X \in \Delta$ and $X \cap \{sat_i\} \neq \emptyset$. Thus, $sat_i \in X$, and by Lemma 1.1, $\gamma = \langle C\rangle sat_i \in \Delta$. For the direction to the left, let $\langle C\rangle sat_i \in \Delta$. Let

$$\chi_i = \bigvee_{S \subseteq \mathcal{A}} \xi_{(S \cup \{sat_i\})}$$

$sat_i \rightarrow \chi_i$ is a $\mathcal{L}_c$ instance of a propositional tautology, so $[C](sat_i \rightarrow \chi_i) \in \Delta$ by *Nec*. By Lemma 1.3, $\langle C\rangle\chi_i \in \Delta$. By Lemma 1.2,

$$\bigvee_{S \subseteq \mathcal{A}} \langle C\rangle\xi_{(S \cup \{sat_i\})} \in \Delta$$

and thus $\langle C\rangle\xi_{S \cup \{sat_i\}} \in \Delta$ for some $S \subseteq \mathcal{A}$. It follows that $S \cup \{sat_i\} \in V(C)$, and since $\Gamma, (S \cup \{sat_i\}) \models_Q sat_i$ we get that $\Gamma \models_Q \langle C\rangle sat_i$ which concludes the proof of the direction to the left in the innermost induction proof. Both the inner and the outer induction steps (negation and disjunction) are straightforward. □

Note that the completeness proofs demonstrates that we do not need to deal with multiple satisfaction of an agent's goal: in fact, one (symbol for a) goal for each agent is enough to reason about abilities for satisfaction!

3. TEMPORAL QCGS

In principle there are many ways to temporalise QCGs. As a first investigation, we assume a linear time model, in which, at each time point, a (possibly different) QCG Γ is played. A *temporal qualitative coalitional game* (TQCG) is then a triple

$$M = \langle S, \sigma, Q\rangle \text{ where:}$$

- S is a set of *states*;
- $\sigma : \mathbb{N} \rightarrow S$ associates a state $\sigma(u)$ with every natural number time point $u \in \mathbb{N}$; and
- $Q : S \rightarrow \mathbf{Q}$, where $\mathbf{Q}$ is the class of all QCGS, is a function associating a qualitative coalitional game $Q(s) = \langle \mathcal{A}^s, \mathcal{G}^s, \mathcal{G}^s_1, \ldots, \mathcal{G}^s_n, \mathcal{V}^s\rangle$ with every state s.

We will make just one requirement of TQCGs: that the set of agents and overall goals remains the same in all states. Formally, $\forall s, t \in S$: $\mathcal{A}^s = \mathcal{A}^t$ and $\mathcal{G}^s = \mathcal{G}^t$. This does not mean that an agent's goals must remain fixed, however: we allow for the possibility that an agent has different goals in different states. We also admit the possibility of a coalition having different choices in different states. Since the sets of agents and overall goals are fixed across all states, we will simply denote these by $\mathcal{A}$ and $\mathcal{G}$ respectively, omitting the state index.

3.1 A Logic for TQCGs

To express properties of TQCGs, we extend the QCG language $\mathcal{L}(QCG)$ with the standard temporal operators of linear-time temporal logic: $\bigcirc$ – "next", $\Diamond$ – "eventually", $\Box$ – "always in the future", and $\mathcal{U}$ – "until" [8]. Formally, the language $\mathcal{L}(TQCG)$ is defined by the grammar φ_t.

$$\varphi_t \quad ::= \quad \langle C\rangle\varphi_c \mid \neg\varphi_t \mid \varphi_t \vee \varphi_t \mid \varphi_t \,\mathcal{U}\, \varphi_t \mid \bigcirc\varphi_t$$

We again assume the usual derived propositional connectives, in addition to $\Diamond\varphi$ for $\top\,\mathcal{U}\,\varphi$ and $\Box\varphi$ for $\neg\Diamond\neg\varphi$. Moroever, we define $\Box^*\varphi$ as $(\varphi \wedge \Box\varphi)$ (φ is true now and always in the future), and $\Diamond^*\varphi = \neg\Box^*\neg\varphi$ (φ is true now or sometime in the future).

When $M = (S, \sigma, Q)$ is a TQCG, $u \in \mathbb{N}$, and φ is a $\mathcal{L}(TQCG)$ formula, the satisfaction relation $M, u \models_T \varphi$ is defined as follows (the cases for negation and disjunction are defined as usual):

$M, u \models_T \varphi$ iff $Q(\sigma(u)) \models_Q \varphi$, when $\varphi \in \mathcal{L}(QCG)$

$M, u \models_T \bigcirc\psi$ iff $M, u+1 \models_T \psi$

$M, u \models_T \psi_1 \,\mathcal{U}\, \psi_2$ iff there is some i such that $M, u+i \models_T \psi_2$ and for all $0 < j < i$ $M, u+j \models_T \psi_1$

For instance, the following formula of $\mathcal{L}(TQCG)$ means that eventually, agent 3 can always choose to satisfy agents 1 and 4 simultaneously: $\Diamond\Box\langle 3\rangle(sat_1 \wedge sat_4)$.

We will henceforth use $\mathcal{L}(TQCG)$ to refer to both the language, and the logic we have defined over this language.

3.2 An Example

We illustrate the logic by a small example. We focus here on temporal properties of goal satisfaction, rather than on contrasting the power of different coalitions (i.e., on which coalitions are likely to form). The latter is discussed in detail in Section 4.

We model the following situation by a temporal qualitative coalitional game. Two agents 1 and 2 both need to use the same resource, say a web service, from time to time. Sometimes an agent needs *read* access, and sometimes it needs *write* access. The integrity of the web service is violated if at the same time either i) both read and write access are granted (inconsistent reads), ii) two write accesses are granted (inconsistent writes) or iii) no read access and no write access are granted (inefficiency).

Let $M = (S, \sigma, Q)$ be a TQCG where S is some infinite set of states, and σ and Q are such that the following holds for $Q(\sigma(k)) = \langle \mathcal{A}, \mathcal{G}, \mathcal{G}_1^{\sigma(k)}, \mathcal{G}_2^{\sigma(k)}, \mathcal{G}_{sys}^{\sigma(k)}, \mathcal{V}^{\sigma(k)}\rangle$ for any $k \geq 0$:

- $\mathcal{A} = \{1, 2, sys\}$. We model the agents as players 1 and 2, and the web service as player sys ("the system").
- $\mathcal{G} = \{r, w_1, w_2, ok\}$. That each of these goals are achieved means that right now:

  r : every client is granted read access
  w_i : agent i is granted write access
  ok : the integrity of the system is not violated

- $\mathcal{G}_1^{\sigma(k)} = \begin{cases} \{w_1\} & \text{if } k \mod 5 = 0 \\ \{r, w_1\} & \text{otherwise} \end{cases}$
  Agent 1 needs to have write access at every fifth point in time. At any other point in time, it is happy as long as it is not left idle, i.e., if it has either read or write access.
- $\mathcal{G}_2^{\sigma(k)} = \begin{cases} \{w_2\} & \text{if } k \mod 3 = 0 \\ \{r, w_2\} & \text{otherwise} \end{cases}$
  Agent 2's goals are similar to agent 1's, except that it needs write access at every third instead of fifth time point.
- $\mathcal{G}_{sys}^{\sigma(k)} = \{ok\}$. The system is satisfied if the integrity is not violated. Note that $\mathcal{G}_{sys}^{\sigma(k)}$ does not depend on k; the system's goal does not vary over time.

- $\mathcal{V}^{\sigma(k)}(sys) = \left\{ \begin{array}{c} \emptyset, \{w_1, ok\}, \{w_2, ok\}, \{r, ok\}, \\ \{w_1, w_2\}, \{w_1, r\}, \{w_2, r\}, \{w_1, w_2, r\} \end{array} \right\}$. The web service can satisfy certain sets of goals. These sets does not necessarily include the goal that the integrity is not violated. We have implicitly defined what is the desired behaviour of the system: each choice involving ok implements a choice in which the integrity invariant is not violated. Note that the choices available to the system do not vary over time. For reasons of space, in this example we are not bothered about $\mathcal{V}^s(C)$ when C is a coalition different from $\{sys\}$.

The following properties hold in $M, 1$.

1. $\Box\langle sys\rangle sat_{sys}$. The system can maintain integrity.
2. $\Box(\langle sys\rangle sat_1 \wedge \langle sys\rangle sat_2)$. Agent 1 can always be satisfied by the system, and the same for agent 2.
3. $\Box\langle sys\rangle(sat_1 \wedge sat_2)$. Agents 1 and 2 can always be simultaneously satisfied by the system.
4. $\Diamond\!\!\!\!\Box\neg\langle sys\rangle(sat_1 \wedge sat_2 \wedge sat_{sys})$. The system cannot always satisfy agents 1 and 2 simultaneously without violating the integrity of the system.
5. $\Box\langle sys\rangle\neg sat_1$. The system can keep agent 1 unsatisfied forever.
6. $\Box\Diamond\!\!\!\!\Box\langle sys\rangle(\neg sat_1 \wedge \neg sat_2 \wedge sat_{sys})$. It is infinitely often the case that the system can make agents 1 and 2 unsatisfied at the same time without violating integrity (this happens at multiples of fifteen).
7. $\langle sys\rangle(\neg sat_1 \wedge \neg sat_2 \wedge sat_{sys})\,\mathcal{U}\,\neg\langle sys\rangle(sat_1 \wedge sat_2 \wedge sat_{sys})$. At some point in the future (i.e., $u = 15$), the system is unable to jointly satisfy agents 1 and 2 without violating integrity. Up until that time, *sys* is always able to make agents 1 and 2 jointly *un*satisfied (note that we evaluate the formula in $M, 1$).
8. $\psi \wedge \Box(\psi \rightarrow \overbrace{\bigcirc(\neg\psi \wedge \cdots \bigcirc(\neg\psi \wedge \bigcirc\psi)\cdots)}^{\text{“}\bigcirc(\neg\psi\wedge\text{” 14 times}})$ where $\psi = \langle sys\rangle(\neg sat_1 \wedge \neg sat_2 \wedge sat_{sys})$. The system can make agents 1 and 2 jointly unsatisfied without violating integrity at time points which are multiples of fifteen, and at no other time points.

As a final point, observe that from a logical point of view, the situations at time points 3 and 5 are indistinguishable:

$$Q(\sigma(3)) \rightleftharpoons Q(\sigma(5))$$

This once again demonstrates that our logic abstracts away from *how* a coalition satisfies individuals: obviously, to satisfy agent 1 for instance, *sys* has to make different choices in $\sigma(3)$ from those in $\sigma(5)$.

3.3 Expressive Power of TQCGs

The notion of simulation for QCGs (Section 2.2) can be naturally lifted to the temporal case. When $M = (S, \sigma, Q)$ and $M' = (S', \sigma', Q')$ are TQCGS and $k \geq 0$, we define

$$\begin{array}{lcl} M, k \rightleftharpoons_T M', k & \Leftrightarrow & Q(\sigma(k)) \rightleftharpoons Q'(\sigma'(k)) \\ M \rightleftharpoons_T M' & \Leftrightarrow & \forall_{n\geq 0} M, n \rightleftharpoons_T M', n \end{array}$$

The notion of elementary equivalence for TQCGS over the language $\mathcal{L}(TQCG)$ can be defined as follows. $M, k \equiv M', k$ iff, for every $\varphi \in \mathcal{L}(TQCG)$, $M, k \models_T \varphi$ iff $M', k \models_T \varphi$. $M \equiv M'$ iff $M, k \equiv M', k$ for every $k \geq 0$.

THEOREM 4. *For all TQCGs M, M': $M \rightleftharpoons_T M' \Leftrightarrow M \equiv M'$*

Note that in the temporal case, the fact that $M, k \rightleftharpoons_T M', k$ is not sufficient for $M, k \equiv M', k$ to hold.

3.4 Satisfiability

The *satisfiability* problem for $\mathcal{L}$(TQCG) is as follows: given a formula $\varphi \in \mathcal{L}$(TQCG), does there exist a TQCG M and $u \in \mathbb{N}$ such that $M, u \models \varphi$?

THEOREM 5. *The sat. probl. for $\mathcal{L}(TQCG)$ is* PSPACE*-complete.*

PROOF. Membership of PSPACE follows from the fact that satisfiability for LTL+K_n (the fusion of LTL and multi-modal K) is PSPACE-complete [5]. Any $\mathcal{L}$(TQCG) formula is also a formula of LTL+K_n, interpreting sat_i as Boolean variable. (The reverse is not the case, of course.) But the relationship is more than merely syntactic: for all $\varphi \in \mathcal{L}$(TQCG):

$$\varphi \text{ is } \mathcal{L}\text{(TQCG)-satisfiable iff } \varphi \text{ is LTL+}K_n \text{ satisfiable}$$

(Notice that we are here quantifying over $\mathcal{L}$(TQCG), formulae, not LTL+K_n formulae.) Given an LTL+K_n interpretation that satisfies $\varphi \in \mathcal{L}$(TQCG), it is straightforward to extract from this a TQCG that satisfies φ.

For PSPACE-hardness, we reduce LTL satisfiability [14]. First, let $\varphi^\dagger$ denote the result of systematically replacing each Boolean variable p that occurs in LTL formula φ with a symbol sat_p. Next, we define a transformation τ, from LTL formulae to $\mathcal{L}$(TQCG), as follows:

$$\tau(\varphi) = \left\{ \begin{array}{ll} [1](\varphi^\dagger) & \text{where } \varphi \text{ is propositional} \\ \#\tau(\psi) & \text{where } \varphi = \#\psi \text{ and } \# \in \{\neg, \bigcirc\} \\ \tau(\psi)\#\tau(\chi) & \text{where } \varphi = \psi\#\chi \text{ and } \# \in \{\vee, \mathcal{U}\} \end{array} \right.$$

Finally, given an LTL formula φ, the $\mathcal{L}$(TQCG) instance φ^τ we create is:

$$\varphi^\tau = (\langle 1\rangle\top) \wedge (\Box\langle 1\rangle\top) \wedge \tau(\varphi)$$

We claim that φ is LTL satisfiable iff φ^τ is $\mathcal{L}$(TQCG) satisfiable; the proof is an easy induction. The key point is that the choice sets of agent 1 in any TQCG satisfying φ^τ define an appropriate valuation for propositional variables in a corresponding LTL interpretation satisfying φ, and *vice versa* (remember that $[1]\varphi$ iff φ holds for all of 1's choices). The first two conjuncts in the definition of φ^τ ensure that such a choice set always exists. □

3.5 Axiomatisation for TQGCs

The system $\mathbf{K}(TQCG)$ over the language $\mathcal{L}(TQCG)$ is defined as follows, where φ, ψ are arbitrary $\mathcal{L}(TQCG)$ formulae, A, B are arbitrary $\mathcal{L}(QCG)$ formulae, α, β are arbitrary $\mathcal{L}_c$ formulae and C an arbitrary coalition. For simplicity, we write $\vdash_{\mathbf{T}}$ instead of $\vdash_{\mathbf{K}(TQCG)}$ for derivability in $\mathbf{K}(TQCG)$.

$Prop^-$	If A is an ($\mathcal{L}(QCG)$) instance of a propositional tautology, then $\vdash_{\mathbf{T}} A$
K^-	$\vdash_{\mathbf{T}} [C](\alpha \rightarrow \beta) \rightarrow ([C]\alpha \rightarrow [C]\beta)$
MP^-	If $\vdash_{\mathbf{K}(QCG)} A$ and $\vdash_{\mathbf{K}(QCG)} A \rightarrow B$, then $\vdash_{\mathbf{T}} B$
Nec^-	If α is an ($\mathcal{L}_c$) instance of a propositional tautology, then $\vdash_{\mathbf{T}} [C]\alpha$
$A1$	$\vdash_{\mathbf{T}} \Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$
$A2$	$\vdash_{\mathbf{T}} \bigcirc\neg\varphi \leftrightarrow \neg\bigcirc\varphi$
$A3$	$\vdash_{\mathbf{T}} \bigcirc(\varphi \rightarrow \psi) \rightarrow (\bigcirc\varphi \rightarrow \bigcirc\psi)$
$A4$	$\vdash_{\mathbf{T}} \Box\varphi \rightarrow (\bigcirc\varphi \wedge \bigcirc\Box\varphi)$
$A5$	$\vdash_{\mathbf{T}} \Box(\varphi \rightarrow \bigcirc\varphi) \rightarrow (\bigcirc\varphi \rightarrow \Box\varphi)$
$U1$	$\vdash_{\mathbf{T}} \varphi\,\mathcal{U}\,\psi \rightarrow \Diamond\!\!\!\!\Box\psi$
$U2$	$\vdash_{\mathbf{T}} \varphi\,\mathcal{U}\,\psi \leftrightarrow \bigcirc\psi \vee (\bigcirc\varphi \wedge \bigcirc(\varphi\,\mathcal{U}\,\psi))$
$Prop$	If φ is an ($\mathcal{L}(TQCG)$) instance of a propositional tautology, then $\vdash_{\mathbf{T}} \varphi$
MP	If $\vdash_{\mathbf{T}} \varphi$ and $\vdash_{\mathbf{T}} \varphi \rightarrow \psi$, then $\vdash_{\mathbf{T}} \psi$
Nec	If $\vdash_{\mathbf{T}} \varphi$ then $\vdash_{\mathbf{T}} \Box\varphi$

Axioms $Prop^-$ and K^- and rules MP^- and Nec^- say that every $\mathbf{K}(QCG)$-theorem is also a $\mathbf{K}(TQCG)$-theorem. The subsystem consisting of axioms $A1$–$U2$ and rules *Prop–Nec* is a version (with $\mathcal{L}(QCG)$ formulae in place of atomic propositions) of an axiomatisation of linear time logic proved to be be sound and complete in [6].

THEOREM 6 (SOUNDNESS & COMPLETENESS). *For any* $\varphi \in \mathcal{L}(TQCG)$: $\vdash_{\mathbf{T}} \varphi \quad \Leftrightarrow \quad \models_T \varphi$

PROOF. The logic $\mathbf{K}(TQCG)$ is what Finger and Gabbay [4] calls a *temporalisation* of $\mathbf{K}(QCG)$: the language of $\mathbf{K}(TQCG)$ has atomic $\mathbf{K}(QCG)$ formulae in place of atomic propositions; the semantic structures of $\mathbf{K}(TQCG)$ identifies a semantic structure for $\mathbf{K}(QCG)$ at each time point used to interpret $\mathbf{K}(QCG)$ formulae; and the axioms/rules of $\mathbf{K}(TQCG)$ are the axioms/rules of the temporal logic for temporal formulae in addition to axioms/rules of $\mathbf{K}(QCG)$ for $\mathbf{K}(QCG)$ formulae.

Finger and Gabbay show that the temporalisation of a sound and complete system is sound and complete. It should be noted that our definition of $\mathbf{K}(TQCG)$ differs from the definition of a temporalisation in [4] by the following. First, we do not have past-time operators in our language. The expressive power is nevertheless the same [6]. Second, we use a slightly different temporal axiomatisation. Neither of these differences change the soundness and completeness proof in [4] in any significant degree. The theorem thus follows immediately from Theorem 3. □

4. CHARACTERIZING TQCGS

In this section, we investigate the axiomatic characterisation of various classes of TQCG. As usual, in saying that a formula scheme φ characterises a property P of models, we mean that φ is valid in a model M iff M has property P; if only the right-to-left part of this biconditional holds, then we say property P implies φ. Also note that for an $\mathcal{L}(TQCG)$ formula φ, to say that φ is valid in a class of models, is the same as saying that $\Box^*\varphi$ is valid in that class.

Basic Correspondences

Let $h^s(C)$ denote the set of all agents that could possibly be satisfied (not necessarily jointly) by coalition C in state s:

$$h^s(C) = \{i : i \in \mathcal{A} \,\&\, \exists G \in \mathcal{V}^s(C), \mathcal{G}_i^s \cap G \neq \emptyset\}$$

The "h" here is for "happpiness": we think of $h^s(C)$ as all the agents that C could possibly make happy in s. Thus the semantic property $i \in h^s(C)$ is a counterpart to the syntactic expression $\langle C\rangle sat_i$.

The first property on models that we consider is the *persistence of happiness* (PH): if coalition C can make i happy in a state s, they can make i happy in the state immediately following s.

$$\forall u \in \mathbb{N}, (i \in h^{\sigma(u)}(C)) \rightarrow (i \in h^{\sigma(u+1)}(C)) \qquad (PH)$$

We have the following characterisation.

LEMMA 2. $\langle C\rangle sat_i \rightarrow \bigcirc\langle C\rangle sat_i$ *characterises* PH.

In the same way, we can characterise the persistence of *un*happiness: property PU says that if C cannot make i happy in a state s, then they cannot make i happy in the state t that immediately follows s.

$$\forall u \in \mathbb{N}, (i \notin h^{\sigma(u)}(C)) \rightarrow (i \notin h^{\sigma(u+1)}(C)) \qquad (PU)$$

LEMMA 3. $\neg\langle C\rangle sat_i \rightarrow \bigcirc\neg\langle C\rangle sat_i$ *characterises* PU.

Now consider the following two constraints. The first, EH, says that eventually, C will be able to make i happy.

$$\exists u \in \mathbb{N}, (i \in h^{\sigma(u)}(C)) \qquad (EH)$$

Notice that in the terminology of reactive systems, this is a *fairness* or *response* property [8, p.288]: it implies that something (i being made happy) can happen infinitely often. (Of course, the fact that C *can* make i happy infinitely often does not mean they will do so.)

LEMMA 4. $\diamondsuit^*\langle C\rangle sat_i$ *characterises* EH.

The obvious counterpart to EH is of course the property EU, which states that, eventually, C will be unable to satisfy i.

$$\exists u \in \mathbb{N}, (i \notin h^{\sigma(u)}(C)) \qquad (EU)$$

LEMMA 5. $\diamondsuit^*\neg\langle C\rangle sat_i$ *characterises* EU.

Combining these properties, we get the following.

LEMMA 6. *PH and EH together imply $\diamondsuit^*\Box^*\langle C\rangle sat_i$, while properties PU and EU together imply $\diamondsuit^*\Box^*\neg\langle C\rangle sat_i$.*

Finally, we consider *safety* properties. The constraint AH says that C can always make i happy, while the constraint AU says that C can never make i happy.

$$\forall s \in S, (i \in h^s(C)) \;\; (AH) \quad \forall s \in S, (i \notin h^s(C)) \;\; (AU)$$

The characterizations are as follows. (Note that there are some obvious implications between these and other properties that we do not list explicitly – e.g., AH implies both EH and PH.)

LEMMA 7. *$\langle C\rangle sat_i$ characterises AH, and $\neg\langle C\rangle sat_i$ char. AU.*

Basic Properties of Choice Sets

Three obvious constraints that we might consider relate to whether or not a particular coalition C has any "real" choices. The first, ECS, says that C never has any choices.

$$\forall s \in S, \mathcal{V}^s(C) = \emptyset \qquad (ECS)$$

The second says that C always has a meaningful choice.

$$\forall s \in S, \exists G \in \mathcal{V}^s(C), G \neq \emptyset \qquad (NECS)$$

The third says that C can choose everything .

$$\forall s \in S, \mathcal{G} \in \mathcal{V}^s(C) \qquad (CCS)$$

LEMMA 8. *Any model that satisfies ECS also satisfies AU, and so ECS implies $\neg\langle C\rangle sat_i$, while any model that satisfies CCS also satisfies AH, and so CCS implies $\langle C\rangle sat_i$.*

Note that $NECS$ alone does not have any characterization: however, when combined with other properties, below, we will see that it has a role.

Static Goal Sets and Choices

Another two simple properties are that the goal sets for each agent and the choice sets for each coalition are guaranteed to remain unchanged. We get the following two constraints, stating that agent i's goal sets are static (constraint SGS) and that coalition C's choices remain static (SC).

$$\forall s, s' \in S, (\mathcal{G}_i^s = \mathcal{G}_i^{s'}) \qquad \text{(SGS)}$$

$$\forall s, s' \in S, (\mathcal{V}^s(C) = \mathcal{V}^{s'}(C)) \qquad \text{(SC)}$$

Taken separately, there does not seem too much we can say about static goal sets and static choice sets. However, taken together, we get the following.

LEMMA 9. *Any model satisfying both SGS and SC also satisfies PH and PU, and as a consequence, SGS and SC together imply* $\langle C\rangle sat_i \leftrightarrow \bigcirc\langle C\rangle sat_i$.

Note that we do not immediately derive a characterisation here. It is perfectly well possible that $\langle C\rangle sat_i \leftrightarrow \bigcirc\langle C\rangle sat_i$ is true in a model M not just because all agents' goals and all coalitions' choices stay fixed, but because there is an intricate interplay going on between for instance an agent changing some of his goals, while at the same time, the coalition C 'synchronously' changing its options. Note that in our example of Section 3.2 for instance, both (SGS) and (SC) are true for $C = \{sys\}$ and $i = sys$, so that, indeed, $\langle\{sys\}\rangle sat_{sys} \leftrightarrow \bigcirc\langle\{sys\}\rangle sat_{sys}$. On the other hand, taking $C = \{sys\}$ and $i = 1$, we don't have (SGS) and (SC), although we still have $\langle\{sys\}\rangle sat_1 \leftrightarrow \bigcirc\langle\{sys\}\rangle sat_1$.

Dynamic Goal Sets

There are several properties we can investigate with respect to goal sets. First, suppose that agent i's goal set is guaranteed to *monotonically decrease* over time. Roughly, this condition means that every agent is guaranteed to get *no easier* to satisfy over time. Formally, this condition on a model M is defined by the following property.

$$\forall u \in \mathbb{N}\,(\mathcal{G}_i^{\sigma(u+1)} \subseteq \mathcal{G}_i^{\sigma(u)}) \qquad (MDGS)$$

LEMMA 10. *Any model satisfying SC and MDGS will satisfy PU, and hence SC and MDGS together imply* $\neg\langle C\rangle sat_i \rightarrow \bigcirc\neg\langle C\rangle sat_i$.

Suppose we this condition is *strict*, so that an agent i is guaranteed to get strictly harder to satisfy over time. This condition is defined by the following further constraint, in addition to $MDGS$.

$$\forall u \in \mathbb{N} \quad \begin{array}{l}(\mathcal{G}_i^{\sigma(u)} = \emptyset) \vee \\ (\exists v \in \mathbb{N} : (v > u) \wedge (\mathcal{G}_i^{\sigma(v)} \subset \mathcal{G}_i^{\sigma(u)}))\end{array} \qquad (SMDGS)$$

We get the following.

LEMMA 11. *Any model satisfying SC, MDGS, and SMDGS will also satisfy PU and EU, and so SC, MDGS, and SMDGS together imply* $\Diamond^*\Box^*\neg\langle C\rangle sat_i$.

Now suppose agent i has monotonically *increasing* goal sets: that is, agent i gets *no harder* to satisfy over time.

$$\forall u \in \mathbb{N}, (\mathcal{G}_i^{\sigma(u)} \subseteq \mathcal{G}_i^{\sigma(u+1)}) \qquad (MIGS)$$

LEMMA 12. *Any model satisfying both SC and MIGS will satisfy constraint PH, and hence SC and MIGS together imply* $\langle C\rangle sat_i \rightarrow \bigcirc\langle C\rangle sat_i$.

The associated strictness constraint is as follows.

$$\forall u \in \mathbb{N} \quad \begin{array}{l}(\mathcal{G}_i^{\sigma(u)} = \mathcal{G}) \vee \\ (\exists v \in \mathbb{N} : (v > u) \wedge (\mathcal{G}_i^{\sigma(u)} \subset \mathcal{G}_i^{\sigma(v)}))\end{array} \qquad (SMIGS)$$

We might expect that SC, $MIGS$, and $SMIGS$ together imply the validity of the formula scheme $\Diamond^*\Box^*\langle C\rangle sat_i$, but this is not the case. A counter example is given by a model that satisfies the empty choice set property (ECS) for coalition C, as described above. If we add the constraint that the choices for C are non-empty $(NECS)$, however, then we get the following.

LEMMA 13. *Any model that satisfies NECS, SC, MIGS, and SMIGS also satisfies PH and EH, and hence the following formula scheme will be valid in any model satisfying NECS, SC, MIGS, and SMIGS:* $\Diamond^*\Box^*\langle C\rangle sat_i$.

Dynamic Choices

We can also consider the ways in which the choices available to coalitions may change over time. Analogously to $MIGS$ and $MDGS$, we can define properties $MICS$ and $MDCS$, which say that the sets of choices available to coalition C monotonically increase and decrease respectively.

$$\forall u \in \mathbb{N}, (\mathcal{V}^{\sigma(u)}(C) \subseteq \mathcal{V}^{\sigma(u+1)}(C)) \qquad \text{(MICS)}$$

$$\forall u \in \mathbb{N}, (\mathcal{V}^{\sigma(u+1)}(C) \subseteq \mathcal{V}^{\sigma(u)}(C)) \qquad \text{(MDCS)}$$

Notice that taken together, these two conditions imply static choice sets (SC). Alone, the properties do not have any characterisation, but axioms emerge when we make assumptions about goal sets.

LEMMA 14. *(1) Any model satisfying MICS and SGS will satisfy constraint PH, and hence MICS and SGS together imply* $\langle C\rangle sat_i \rightarrow \bigcirc\langle C\rangle sat_i$.
(2) Any model satisfying MDCS and SGS will satisfy constraint PU, and hence MICS and SGS together imply $\neg\langle C\rangle sat_i \rightarrow \bigcirc\neg\langle C\rangle sat_i$.

The associated strictness condition for increasing choice sets is:

$$\begin{array}{l}\forall u \in \mathbb{N}, \forall G_1 \in \mathcal{V}^{\sigma(u)}(C)(G_1 = \mathcal{G}) \vee \\ \quad (\exists v \in \mathbb{N}, \exists G_2 \in \mathcal{V}^{\sigma(v)}(C), (v > u) \wedge (G_1 \subset G_2))\end{array} \qquad (SMICS)$$

LEMMA 15. *Any model satisfying MICS, SGS, and SMICS or MICS, MIGS, and SMICS will also satisfy constraints PH and EH, and hence MICS, SGS, and SMICS together imply* $\Diamond^*\Box^*\langle C\rangle sat_i$.

We omit here the analysis for the case of monotonically *de*creasing choice sets.

Solution Concepts

In [15], a range of different solution concepts were defined for QCGs. It should be clear that many of the solution concepts of [15] can be characterised via formulae of $\mathcal{L}(QCG)$. For example, a basic solution concept is that of a *successful* coalition – one that has a choice available such that this choice satisfies all its members [15, p.47]. We can characterise this via a predicate $succ(C)$, as follows.

$$succ(C) \equiv \langle C\rangle(\bigwedge_{i\in C} sat_i)$$

Similarly, the notion of a minimal coalition (one such that no subset of the coalition is successful [15, p.51]) may be captured as follows.

$$min(C) \equiv \bigwedge_{C'\subseteq C} \neg succ(C')$$

Thus the core of a coalition being non-empty [15, p.54] may be captured as follows:

$$cne(C) \equiv (succ(C) \wedge min(C))$$

The idea of agent i being a veto player for agent j [15, p.57] is defined by:

$$veto(i,j) \equiv \bigwedge_{C\subseteq\mathcal{A}} (\langle C\rangle sat_j \rightarrow \neg\langle C \setminus \{i\}\rangle sat_j)$$

And finally, the idea of a coalition being mutually dependent [15, p.58] is captured as follows:

$$md(C) \equiv \bigwedge_{i\neq j\in C} veto(i,j)$$

How might these concepts be extended into the temporal dimension of TQCGs and $\mathcal{L}(TQCG)$? It should first be clear that each concept has four different temporal versions, corresponding to prefixing the formula characterising it with one of the following four, increasingly powerful temporal operators:

$$\Diamond \qquad \Box\Diamond \qquad \Diamond\Box \qquad \Box$$

Thus, for example, $\Box\Diamond succ(C)$ means that coalition C are successful *infinitely often* – no matter which time point we pick, there will be a subsequent time point at which C are successful. (Using the terminology of reactive systems [8], we might then say that C are hence *fairly successful*.) Similarly, a *temporally strong* form of coalitional stability is captured by the formula $\Box cne(\mathcal{A})$: if this formula is satisfied in a TQCG, then, it can be argued, the only coalition that will ever form is the grand coalition.

It is potentially more interesting, however, to study a richer interplay between temporal and QCG dimensions. For example, from agent is point of view, perhaps the only really interesting issue is whether at every time point there is some stable coalition, containing this agent.

$$tstable(i) \equiv \Box \bigvee_{C \subseteq \mathcal{A}: i \in C} cne(C)$$

From the point of view of a coalition C, which seeks to form, the notion of a *stable government* seems relevant: a stable government is a coalition that can always satisfy its "electorate".

$$sg(C) \equiv \Box\langle C\rangle(\bigwedge_{i \in \mathcal{A}} sat_i)$$

This can of course be strengthened, requiring C to in addition be an internally stable coalition.

$$sg'(C) \equiv \Box(cne(C) \wedge \langle C\rangle(\bigwedge_{i \in \mathcal{A}} sat_i))$$

With respect to mutual dependence, one possibility, captured by the formula $\Box md(C)$, is that a coalition is *always* mutually dependent. However, we can capture a weaker type of mutual dependence as follows:

$$wmd(C) \equiv \bigwedge_{i \neq j \in C} \Diamond veto(i,j)$$

We draw two conclusions. The first is that the language $\mathcal{L}(TQCG)$ is well suited to capturing such solution concepts: it makes it possible to express elegantly concepts that would be difficult to understand were they expressed at the semantic level. The second is that extending QCGs into the temporal dimension adds an entirely new level of richness to their structure, which as these examples suggest, demands further study.

5. CONCLUSION

Qualitative Game Structures were introduced in [15] as a model to deal with one of the driving questions in cooperation: 'Which coalition should I join?'. The design of such games is motivated by a principle of economy: rather than associationg a utility to every choice, the emphasis is on *satisfaction* of agents, which is triggered or not by *choices* of coalitions.

The logical analysis of such games, as underlines the idea that we have a basic and simple notion of coalitional games with QCGs: a natural language for it gives rise to an axiomatisation almost identical to the simplest modal logic **K**. We established several technical results for this language, and were then able to lift them to the era of temporal QCGs.

The possible directions for further research are multiple. First of all, the properties of TQGCs that we characterised in Section 4 are only the most straightforward. Even in static games, there are interesting conditions to be investigated (see the *monotonicity* property mentioned in Section 2, for example). Second, our way of temporalising QCGs also only reflects a simple case. It would be interesting to add temporal structure to the games themselves, and reason about what agents can achieve over time, by applying suitable *strategies*, rather than making 'one-shot choices'. In addition, *finite horizon* versions of TQCGs might also be worth investigating: for example, if an agent is only concerned about being satisfied *once*, then it might be prepared to join a coalition that does not satisfy it throughout a game, as long as, in the final state of the game, the coalition *does* satisfy it. Such strategising is not possible or appropriate in infinite horizon games.

Acknowledgements

The research reported in this paper was carried out when the first author was visiting the Department of Computer Science, University of Liverpool, supported by grant 166525/V30 from the Norwegian Research Council.

6. REFERENCES

[1] R. Alur, T.A. Henzinger, O. Kupferman, and M.Y. Vardi. Alternating refinement relations. In *Proc. of CONCUR'98*, vol. 1466 of *LNCS*, pp. 163–178. Springer-Verlag, 1998.

[2] R. Axelrod. *The Evolution of Cooperation*. Basic Books: New York, 1984.

[3] K. Binmore. *Fun and Games: A Text on Game Theory*. D. C. Heath and Company: Lexington, MA, 1992.

[4] M. Finger and D. M. Gabbay. Adding a temporal dimension to a logic system. *Journal of Logic, Language, and Information*, 1:203–233, 1992.

[5] D. M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*. Elsevier, 2003.

[6] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *POPL '80*, pp. 163–173, 1980. ACM Press.

[7] M. Klusch and A. Gerber. Dynamic coalition formation among rational agents. *IEEE Int. Syst.*, 17(3):42–47, 2002.

[8] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verl.: Berlin, Germany, 1992.

[9] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press: Cambridge, MA, 1994.

[10] M. Pauly. *Axiomatising Judgement Aggregation Procedures in a Mininal Logical Language*. Unpublished paper.

[11] T. Sandholm. Distributed rational decision making. In G. Weiß, editor, *Multiagent Systems*, pages 201–258. The MIT Press: Cambridge, MA, 1999.

[12] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1–2):209–238, 1999.

[13] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, 1998.

[14] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[15] M. Wooldridge and P. E. Dunne. On the computational complexity of qualitative coalitional games. *Artificial Intelligence*, 158(1):27–73, 2004.

Reasoning about Action and Cooperation

Luigi Sauro
Jelle Gerbrandy
Dipartimento di Informatica
University of Torino, Italia
jelle@gerbrandy.com
sauro@di.unito.it

Wiebe van der Hoek
Michael Wooldridge
Department of Computer Science
University of Liverpool, United Kingdom
wiebe@csc.liv.ac.uk
mjw@csc.liv.ac.uk

ABSTRACT

We present a logic for reasoning both about the ability of agents to cooperate to execute complex actions, and how this relates to their ability to reach certain states of affairs. We show how the logic can be obtained in a modularised way, by combining a model for reasoning about actions and their effects with a model that describes what actions an agent can perform. More precisely, we show how one can combine an action logic which resembles Propositional Dynamic Logic with a cooperation logic which resembles Coalition Logic. We give a sound and complete axiomatisation for the logic, illustrate its use by means of an example, and discuss possible future extensions to it.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*coherence and coordination, multiagent systems*

General Terms

Theory

Keywords

Agent and multi-agent architectures, Cooperation and coordination among agents, Logics for agent systems, (Multi-)Agent planning

1. INTRODUCTION

When reasoning about cooperation in multi-agent systems, at least two questions seem important: *Who* can achieve some state of affairs and *How* can they obtain it? Historically, the latter question has received far more attention than the former. In the AI planning community, for example, it is known as the means-ends reasoning problem, and indeed the AI planning project is directed at developing computer programs that can answer it efficiently [4]. In the logic community, the question has been studied largely through the medium of action and program logics, of which PDL is perhaps the best known example [7]. A basic issue in such logics is the construction of complex programs from simpler ones. Typical constructs to build complex programs include sequential composition, non-deterministic choice and tests. However, there are limits to what one can say in such logics. For example, in standard PDL, one cannot express that an action is *not* executed; nor is not straightforward to say that one complex action *subsumes*, or *implies* another. Last but not least, although there is a notion of concurrency in PDL, all processes are expected to behave as specified by the program under consideration, and there is no notion of competition or cooperation, and hence no need to express *who* can achieve something. Building on work in PDL and related formalisms, some early attempts in the multi-agent systems community to formalise the ability of agents to execute plans and bring about states of affairs include [8, 14, 9].

The question of *who* can achieve some state of affairs has received much less attention within the AI and logic community. Although Moore presented a modal analysis of ability as long ago as 1977 [11], and a number of other authors built on this work over the intervening three decades, it is only recently that the logical analysis of ability has entered the mainstream of multi-agent systems research. Two closely related formalisms have received particular attention: the Alternating-time Temporal Logic (ATL, [1]), in which one can express that a coalition can enforce a certain temporal property, and Coalition Logic ([12]), in which the basic expression states that a coalition is effective for some outcome. Another closely related formalism is the Coalition Logic for Propositional Control (CL-PC, [15]), where the propositional atoms under the control of an agent determine what he, or any coalition he is a member of, can achieve. All these approaches revolve around a model in which a set of agents, who may have conflicting interests, may or may not work together to obtain a certain state of affairs. Central is the notion of so-called α-ability: the capability of a group of agents to enforce a certain state of affairs, no matter what actions the other agents take. As in [15], we will call such logics *Cooperation Logics*.

Notice that in all of these approaches, the ability of groups of agents to obtain a state of affairs is modeled without an explicit representation of *how* these agents obtain them. In other words, these logics do not represent actions and plans in the object language, even though in some cases, actions or choices are present in the underlying semantics. However, such an *explicit* representation of action in the object language is important in many domains. For example, certain actions may be more costly than others, and the costs may be distributed among the agents helping to achieve the goal according to the actions they perform. Also, in some situa-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

tions, it seems a good heuristic to formulate a plan before organising a coalition to execute it [16, 13]. Finally, using such a language for verification of a MAS would not only provide the user — or indeed, the agents themselves— with a proof *that* a desired goal of the system can be achieved, but it would also explicitly capture a *plan* of how to achieve it.

In this paper, we are interested in how the ability to obtain a state of affairs relates to the ability to execute certain, possibly complex, and possibly collective, actions. We develop a logic that combines operators from cooperation logics with an explicit representation of actions. We achieve this by dividing the problem in two parts: a logic for reasoning about actions and their effects (*à la* dynamic logic) together with a variant of coalition logic to reason about (groups of) agents and the actions that they can perform. In more detail, we study multiagent systems that consist of two separate parts: an *environment module* that describes actions and their effects, and an *agents module* describing how the agents can interact with the environment by performing certain actions.

We use the agents module to represent the ability of a coalition G to *execute* certain type of actions. Where β is a joint action (i.e., it denotes a set of actions to be executed concurrently), we write $\langle\langle G\rangle\rangle\beta$ to indicate that coalition G can execute the complex action β. In the environment, then, we can reason about the effect of actions using our environment logic, and write, for example $[\beta]\phi$ for 'every successful execution of β results in state where ϕ is true' (cf. PDL).

The cooperation modality of Coalition Logic expresses the ability of a group to obtain a certain state of affairs. This kind of expression can now be analysed using the two modules described above: a group G of agents has the ability to obtain a state of affairs ϕ just in case there exists a plan β that G is able to execute, which is guaranteed to result in ϕ.

Such an approach presumes that the effect of actions can be modeled more or less independently from the behaviour of the agents that perform these actions. This approach seems applicable to many real life scenarios. For example, an environment may be a communication channel via which agents send and receive messages. Or, the environment may be a Database Management System, defining actions and constraints in a general way, and the agents are users querying and updating the database.

Separating actions from the agents that perform them is not only practical from an operative viewpoint, but from a logical standpoint as well. It allows us to break up the problem of reasoning about planning in a multi-agent environment into two sub-problems, and to study the logic of each of these 'modules' in isolation before putting them together. This approach is also interesting, because it allows us to 'reuse' existing logics and exploit them as 'modules' in a richer system.

In Section 2 we will present the two building blocks of our framework. In section 2.1 we define a logic for the environment, in which one can reason about the effect of executing joint actions. The language can express that an action is not taken, but also, in a natural way, compares joint actions in relative 'strength': if the joint action β is stronger than α, then we also have that whatever is true as a result of α, is true when executing β. We then (in section 2.2) define our agents' module, in which the α-ability of coalitions can be expressed — but now, not with respect to states of affairs that can be reached, but for joint actions that can be guaranteed.

Section 2.3 puts these blocks together in a module called a multi-agent system. Here, we define $\langle\langle G\rangle\rangle\phi$, the α-ability of coalition G to bring about state of affairs ϕ, in terms of $\langle\langle G\rangle\rangle\alpha$ and $[\alpha]\phi$. Apart from the completeness results for the separate building blocks, which we judge to provide insight in the systems themselves, we believe that one of the interesting technical developments of our work relates to the fact that the completeness result for the *combined* logic is straightforward given the completeness of the two logics of the underlying modules. We see this as indicative of the success of the modularisation approach we adopt in this paper, and a good indication that it might be usefully applied elsewhere.

Section 3 is devoted to an example, illustrating how the logic can be used to represent properties of a multi-agent system. The example is a scenario in which agents in a coalition can collectively set variables which they are unable to change on their own. Section 4 indicates a few of the many possible extensions to our multi-agent system, and Section 5 concludes.

2. ACTION AND COOPERATION

In our setting, a multiagent system is composed of two *modules*: one module describes the *agents* in terms of the actions they can execute, another describes the *environment* that is populated by these agents, and in particular the effects of performing certain actions.

2.1 The environment module

There are many ways of modeling an environment. One very general way to model actions and their effects is by using a Labeled Transition System (LTS) [6, 2, 10]. An LTS is a directed graph in which the edges are labeled by elements from a set of atomic actions and the vertices are labeled with sets of propositional variables. Edges are called transitions, the vertices are called states. The idea is that if a state s is connected to another state s' by a transition labeled with an action a, this means that s' is a possible result of executing action a in state s. The sets of variables at each end of the transition tell us how the performance of the action changes the state of the environment, in terms of the propositions that change value.

One of the drawbacks of this basic LTS model is that it does not allow for modeling concurrency directly: there is no straightforward way to model the fact that performing certain actions together may have a very different effect than performing each of them separately.

In the present context, we are interested in the way that the choice of actions of different agents interact, so we propose to model the effect of actions being performed concurrently directly in the model. We do this by labeling the transitions not with single actions, but with *sets* of actions. Intuitively, a transition labeled with a set of actions A represents the effect of executing exactly those actions together, and no others. This approach to deal with concurrency is similar to that taken in [5].

DEFINITION 1 (ENVIRONMENT MODULE).
An environment module Env is a Set-Labeled Transition System *(SLTS) is a tuple* $\langle S, Ac, (\rightarrow)_{A\subseteq Ac}, P, \pi\rangle$ *where:*

1. S *is a set of states.*
2. Ac *is a finite set of actions.*
3. *For each* $A \subseteq Ac$, $\rightarrow_A$ *is a relation over* S.
4. P *is a set of propositional variables*
5. $\pi : (S \times P) \rightarrow \{1, 0\}$ *is an interpretation function that assigns to each state and each propositional variable a truth value.*

We require that the accessibility relations $\rightarrow_A$ *are* serial *for each set of actions* A*, i.e., that for each state* s *there is always an* s' *with* $s \rightarrow_A s'$. □

A state s is characterised by the propositional variables that are true and false in it, as given by $\pi(s)$, and by the possible effects of executing combinations of actions A concurrently, as defined by the accessibility relations $\rightarrow_A$. We will refer to a set of actions A as a *concurrent action*. Actions may be *non-deterministic*, in the sense that the outcome of performing a set of actions may not always determine a unique result.

To reason about actions and their effect in an SLTS, we use a language that is similar to that of Propositional Dynamic Logic. PDL has a language that consists of two parts: an action language to describe actions, and a 'static' language to describe properties of states. However, rather than directly incorporating the program constructs of PDL, we use *Boolean combinations* of atomic actions to reason about sets of actions.

DEFINITION 2 (ACTION EXPRESSIONS).
Given the set of atomic actions Ac, the set of action expressions *is defined by the grammar:*

$$\alpha ::= a \mid \alpha \wedge \alpha \mid \neg\alpha$$

where $a \in Ac$.

To interpret this action language, we define a relation $\models^p$ between subsets of Ac and action expressions. $A \models^p \alpha$ is to be read as 'concurrent action A is of type α', or 'α is true of A'.

DEFINITION 3 (INTERPRETATION OF ACTIONS).

1. $A \models^p a$ *iff* $a \in A$
2. $A \models^p \neg\alpha$ *iff* $A \not\models^p \alpha$
3. $A \models^p \alpha \wedge \beta$ *iff* $A \models^p \alpha$ *and* $A \models^p \beta$.

So, for example, the concurrent action $\{a, b\}$ is of type $a \wedge \neg c$, but it is not of type $\neg b$. Formally, the action language is of course simply classical Boolean propositional logic, with Boolean variables corresponding to atomic actions. We can define additional connectives, such as $\vee$ or $\rightarrow$, in the usual way.

DEFINITION 4 (LANGUAGE OF ENVIRONMENT LOGIC).
Given a set of actions Ac and a set of propositional variables P, the language of environment logic is given by:

$$\phi ::= p \mid \phi \wedge \phi \mid \neg\phi \mid [\alpha]\phi$$

where $p \in P$, and α is an expression of the action language.

In effect, the resulting language is Boolean Modal Logic [3]. The truth of sentences of environment logic is defined in the standard way, as follows.

DEFINITION 5 (SEMANTICS OF ENVIRONMENT LOGIC). *Let $Env = \langle S, Ac, (\rightarrow)_{A \subseteq Ac}, P, \pi\rangle$ be a SLTS, and s is a state in S. Truth of sentences of the language of environment logic is defined inductively as follows:*

1. $Env, s \models^e p$ *iff* $\pi(s, p) = 1$
2. $Env, s \models^e \phi_1 \wedge \phi_2$ *iff* $Env, s \models^e \phi_1$ *and* $Env, s \models^e \phi_2$.
3. $Env, s \models^e \neg\phi$ *iff* $Env, s \not\models^e \phi$
4. $Env, s \models^e [\alpha]\phi$ *iff for all* $A \subseteq Ac$ *and* $s' \in S$, *if* $A \models^p \alpha$ *and* $s \rightarrow_A s'$ *then* $Env, s' \models^e \phi$.

As usual we write $Env \models^e \phi$ when for all states $s \in S$, $Env, s \models^e \phi$. For example, if a is an atomic action, then $[a]\phi$ is true just in case executing a will always result in a state where ϕ is true, no matter what other actions are performed concurrently.

A sentence $[\neg a]\phi$ expresses that *not* doing a will always result in a state where ϕ is true – performing any concurrent action that does not involve a will results in a state where ϕ is true.

In a precise sense, the formula $[\alpha]\phi$ expresses that being of type α is a sufficient condition for a set of actions to guarantee that ϕ be true; the formula $[\neg\alpha]\neg\phi$ expresses that doing an action of type α is a *necessary* condition for reaching a state where ϕ is true – if the action is not of type α, for all the states in which the system will result ϕ is false.

The logic that results from this semantics is given by the following set of axioms and rules.

DEFINITION 6 (ENVIRONMENT LOGIC).
The environment logic *($\vdash^e$) has the following axioms*

1. *All axioms of propositional logic*
2. $[\alpha](\phi \rightarrow \psi) \rightarrow ([\alpha]\phi \rightarrow [\alpha]\psi)$ *for each α (distribution)*
3. *All axioms of the form $[\alpha]\phi \rightarrow [\beta]\phi$, if $\vdash \beta \rightarrow \alpha$ in propositional logic.*
4. $([\alpha]\phi \wedge [\beta]\phi) \rightarrow [\alpha \vee \beta]\phi$ *(additivity)*
5. $\neg[\alpha]\neg\top$ *if α is consistent in propositional logic (seriality)*
6. $[\bot]\ \bot$

Rules:

1. *from $\vdash^e \phi$ and $\vdash^e \phi \rightarrow \psi$, derive $\vdash^e \psi$ (modus ponens)*
2. *from $\vdash^e \phi$, derive $\vdash^e [\alpha]\phi$ (necessitation for each action expression α)*

We write $\Sigma \vdash^e \phi$ if ϕ can be derived from the set of sentences Σ using these rules and axioms.

Axiom 2 and the rule of necessitation state that the $[\alpha]$-operators are normal modal operators. Axiom 3 says that if β implies α in propositional logic, then, if an α-action always results in a ϕ state, then a β-action will as well (since any β-action is an α-action). Axiom 4 says that if all α-actions and all β-actions result in a state where ϕ is true, then also all $\alpha \vee \beta$-actions will. Finally, the seriality axiom reflects that any consistent combination of actions will lead to some resulting state.

This logic is sound and complete with respect to the semantics.

PROPOSITION 7 (SOUNDNESS AND COMPLETENESS).
Environment logic is sound and complete with respect to environment models.

$$\vdash^e \phi \quad \textit{iff} \quad Env \models^e \phi \textit{ for each environment model } Env.$$

proof: We give the main steps of a proof. Construct a canonical model in the usual way, by taking maximal consistent sets of the logic as states in the model. The following *truth lemma* holds for each maximal consistent Σ:

$$\phi \in \Sigma \text{ iff } \Sigma \models^e \phi \text{ in the canonical model}$$

So, suppose $[\alpha]\psi \in \Sigma$. We need to show that $\Sigma \models^e [\alpha]\psi$. If α is not consistent this holds trivially. So, assume that $\Sigma \rightarrow_A \Gamma$ for some A such that $A \models^p \alpha$. Then $\psi \in \Gamma$ by definition of the

canonical model, and by induction hypothesis, $\Gamma \models^e \psi$. Since Γ was arbitrary, it follows that $\Sigma \models^e [\alpha]\psi$.

For the other direction. Before proving the general case, we first consider the case where our action expression is of the form $\alpha_A = \bigwedge A \wedge \bigwedge\{\neg a \mid a \in Ac \text{ and } a \notin A\}$ for some set of actions A.

Suppose $[\alpha_A]\psi \notin \Sigma$ for some A. With axiom 2 and the consistency of Σ, this means that $\{\chi \mid [\alpha_A]\chi \in \Sigma\} \cup \{\neg\psi\}$ is consistent. This set can be extended to a maximal consistent Γ. We show that $\Sigma \rightarrow_A \Gamma$. To see this, take an arbitrary $[\alpha]\chi \in \Sigma$ such that $A \models^p \alpha$. Then, in propositional logic, $\vdash \alpha_A \rightarrow \alpha$, and by axiom 3, it must hold that $[\alpha_A]\chi \in \Sigma$. But then, by construction of Γ, $\chi \in \Gamma$. Putting everything together, we have that $\Sigma \rightarrow_A \Gamma$, $A \models^p \alpha_A$, and, by induction hypothesis, $\Gamma \not\models^e \psi$. We conclude that $\Sigma \not\models^e [\alpha_A]\psi$.

Now we move to the general case. Suppose that $\Sigma \models^e [\alpha]\psi$ for arbitrary α. If α is not consistent, then we can use axioms 6 and 3 to conclude that $[\alpha]\psi$ is in Σ. If α is consistent, then the sentence α is equivalent in propositional logic to the sentence $\alpha^\vee := \bigvee\{\alpha_A \mid A \models^p \alpha\}$. Clearly, we have that $\Sigma \models^e [\alpha^\vee]\psi$; and also that, for axiom 3, $\Sigma \models^e [\alpha_A]\psi$ for each A such that $A \models^p \alpha$. By the previous argument, we have that $[\alpha_A]\psi \in \Sigma$ for each A such that $A \models^p \alpha$. With axiom 4, it follows then that $[\alpha^\vee]\psi \in \Sigma$. We can now use axiom 3 and the fact that $\alpha^\vee \rightarrow \alpha$, and conclude that $[\alpha]\psi \in \Sigma$. □

So, what we have now is a dynamic logic in which we can reason about *concurrent* actions. That is, we have a way of reasoning about the effect of performing *combinations* of actions, as well as about the effect of *not* performing certain actions. This dynamic logic will serve as the environment module of our multi-agent system. We will now proceed to define the agents that will populate this environment.

2.2 Agents Module

Agents are identified with the set of actions that they can perform.

DEFINITION 8 (AGENTS MODULE).
Given a set of agents Ag and a set of actions Ac, an agents *module is a tuple $\langle Ag, Ac, \mathsf{act}\rangle$, where* act *is a function $Ag \mapsto \mathcal{P}(Ac)$ that assigns to each agent i a subset $\mathsf{act}(i)$ of actions from Ac.*
We require that $\bigcup_{i\in Ag} \mathsf{act}(i) = Ac$.

We will abuse notation somewhat, and write 'act' as a *pars pro toto* for $\langle Ag, Ac, \mathsf{act}\rangle$. We denote with $\mathsf{act}(G)$ the set $\bigcup_{i\in G} \mathsf{act}(i)$ of actions that the group G of agents can perform.

To represent the ability of agents to perform actions of a certain type, we will use some notation from cooperation logic. For an action expression α, a sentence of the form $\langle\!\langle G\rangle\!\rangle\alpha$ expresses the fact that G has the power to bring it about that a set of actions of type α will be performed; we say that G is *effective for* α, that G is able to enforce α, or simply that G is able to do α. It is important to note that α here is an action type: we are dealing with the ability to *carry out a complex action*, not the ability to *bring about some state of affairs*.

A group G is able to enforce α exactly if there is a subset from the actions under their control that, no matter what actions the other agents choose, is such that the resulting concurrent action is of type α.

The way the semantics is defined is similar to that of the Coalition Logic of Propositional Control of [15], except that we explicitly allow for different agents to 'control' the same action.

DEFINITION 9 (ABILITY FOR ACTIONS).
Let $\langle Ag, Ac, \mathsf{act}\rangle$ be an agents module, let $G \subseteq Ag$ be a set of agents.

$\langle Ag, Ac, \mathsf{act}\rangle \models^a \langle\!\langle G\rangle\!\rangle\alpha$ *iff there is an $A \subseteq \mathsf{act}(G)$ such that for all $B \subseteq \mathsf{act}(Ag \setminus G)$ it holds that $A \cup B \models^p \alpha$*

So, for example, if act is an agents module in which an agent i can execute action a, then $\mathsf{act} \models^a \langle\!\langle\{i\}\rangle\!\rangle a$: i can enforce a. However, i can only enforce $\neg a$ if he is the only one that can execute a: $\mathsf{act} \models^a \langle\!\langle\{i\}\rangle\!\rangle\neg a$ only if $a \notin \mathsf{act}(j)$ for any j other than i. Similarly, $\mathsf{act} \models^a \langle\!\langle G\rangle\!\rangle(a \wedge \neg b)$ exactly when at least one agent in G can execute a, while no agent outside of G can do b.

DEFINITION 10 (COALITION LOGIC FOR ACTIONS).
The coalition logic for actions is given by the following axioms, together with all axiom schemes of propositional logic, and modus ponens. We denote derivability in this logic with $\vdash^a$.

1. $\langle\!\langle G\rangle\!\rangle\top$, *for all* $G \subseteq Ag$
2. $\langle\!\langle G\rangle\!\rangle\alpha \rightarrow \neg\langle\!\langle Ag \setminus G\rangle\!\rangle\neg\alpha$
3. $\langle\!\langle G\rangle\!\rangle\alpha \rightarrow \langle\!\langle G\rangle\!\rangle\beta$ *if* $\vdash \alpha \rightarrow \beta$ *in propositional logic.*
4. $(\langle\!\langle G_1\rangle\!\rangle\alpha \wedge \langle\!\langle G_2\rangle\!\rangle\beta) \rightarrow \langle\!\langle G_1 \cup G_2\rangle\!\rangle(\alpha\wedge\beta)$ *for all* $G_1, G_2 \subseteq Ag$ *such that* $G_1 \cap G_2 = \emptyset$
5. $\langle\!\langle G\rangle\!\rangle a \rightarrow \bigvee_{i\in G} \langle\!\langle\{i\}\rangle\!\rangle a$, *for all* $G \subseteq Ag$ *and atomic* $a \in Ac$
6. $(\langle\!\langle G\rangle\!\rangle\alpha \wedge \langle\!\langle G\rangle\!\rangle\beta) \rightarrow \langle\!\langle G\rangle\!\rangle(\alpha\wedge\beta)$ *if* α *and* β *have no atomic actions in common.*
7. $\langle\!\langle G\rangle\!\rangle\neg a \rightarrow \langle\!\langle G\rangle\!\rangle a$ *for atomic* $a \in Ac$.
8. $\langle\!\langle G\rangle\!\rangle\alpha \rightarrow \bigvee\{\langle\!\langle G\rangle\!\rangle \bigwedge\Phi \mid \Phi$ *is a set of literals such that* $\bigwedge\Phi \rightarrow \alpha\}$

An explanation of some of the axioms is in order.

The first axiom states that any group of agents is able to enforce some trivial action. Axiom 2 is typical of cooperation logics, and states that if a group G is able to enforce an action α, then the agents outside that group cannot enforce that an action that is not of type α will happen. This follows directly from the definitions.

Axiom 3 says that if a group of agents is able to enforce an α-action, and all α-actions are also of type β, then G is also able to enforce a β-action. Note that the axioms 1 and 3 give us a kind of necessitation, even for actions: If α is a classical tautology, then we also have $\vdash^a \langle\!\langle G\rangle\!\rangle\alpha$.

Axiom 4 states that if two independent groups of agents have the ability to enforce, respectively, actions of type α and of type β, then they can combine forces to enforce an $\alpha\wedge\beta$-action. This holds only if the two groups are disjoint: if this is not the case, and the two groups have an agent in common, say agent i, then it may be that i needs to execute action a to (help) enforce α, while he needs to refrain from doing a to enforce β: there is no single choice of actions that guarantees that $\alpha \wedge \beta$ will happen.

Axiom 5 says that if a group of agents has the ability to enforce an atomic action, then at least one of them must have that ability by himself.

Axiom 6 says that if a group can enforce actions α and β, and α and β involve completely different atomic actions, then the group can also enforce $\alpha\wedge\beta$ (by choosing the actions needed to guarantee α and performing them together with those that guarantee β).

The validity of axiom 7 is a consequence of the assumption in definition 8 that any atomic action can be executed by at least one agent. It says that if a group of agents can guarantee that an *atomic*

action is *not* performed, then one of the agents in that group must be able to execute that action.

Axiom 8 says that if a group of agents can perform an α-action, then there must be a set of literals (that is, formulas of the form a or $\neg a$) that together imply α, that they can enforce – that is, there must be a 'recipe' that tells them which actions to execute and which not to execute, that guarantees that α results.

PROPOSITION 11 (SOUNDNESS AND COMPLETENESS).
Coalition logic for actions is sound and complete with respect to agent models.

proof: Soundness is relatively straightforward. To prove completeness, we define, given a maximal $\vdash^a$-consistent set of sentences Σ, a function act for which the following truth lemma holds:

$$\mathsf{act} \models^a \phi \text{ iff } \phi \in \Sigma$$

which gives us an immediate completeness result. We define act by setting $\mathsf{act}(i) = \{a \mid \langle\langle i \rangle\rangle a \in \Sigma\}$. Note that with axioms 4 and 5, we have for atomic actions a that $a \in \mathsf{act}(G)$ iff $\langle\langle G \rangle\rangle a \in \Sigma$, a fact that we will use in the proof.

Before proving the truth lemma, we define, for a group of agents G and a set of actions $A \subseteq \mathsf{act}(G)$, the set $\Phi(A, G)$ to consist of all literals $A \cup \{\neg a \mid a \in \mathsf{act}(G), a \notin A \text{ and } a \notin \mathsf{act}(Ag \setminus G)\}$. The sentence $\Phi(A, G)$ characterises exactly those possible courses of action that might result when G chooses to do A.

We prove the truth lemma by induction on ϕ, where the only interesting case is the one where ϕ is of the form $\langle\langle G \rangle\rangle \alpha$.

Suppose first that $\mathsf{act} \models^a \langle\langle G \rangle\rangle \alpha$. That means that there is an $A \subseteq \mathsf{act}(G)$ such that for all $B \subseteq \mathsf{act}(Ag \setminus G) : A \cup B \models^p \alpha$. Clearly, then, $\vdash \bigwedge \Phi(A, G) \rightarrow \alpha$ in propositional logic.

By definition of act, it holds that $\langle\langle G \rangle\rangle a \in \Sigma$ for each $a \in A$. Similarly, we have for each $a \notin \mathsf{act}(Ag \setminus G)$ that $\langle\langle Ag \setminus G \rangle\rangle a \notin \Sigma$, from which it follows with maximality and axiom 2 that $\langle\langle G \rangle\rangle \neg a \in \Sigma$. Axiom 6 allows us to conclude that $\langle\langle G \rangle\rangle \bigwedge \Phi(A, G) \in \Sigma$, and by the monotony axiom 3 it follows that $\langle\langle G \rangle\rangle \alpha \in \Sigma$.

For the other direction, assume that $\langle\langle G \rangle\rangle \alpha \in \Sigma$. Since Σ includes axiom 8, there must be a set of literals Φ such that $\bigwedge \Phi \vdash \alpha \in \Sigma$ and $\langle\langle G \rangle\rangle \bigwedge \Phi \in \Sigma$. With axiom 3, we know that for each of the literals l in Φ it holds that $\langle\langle G \rangle\rangle l \in \Sigma$.

For positive literals a, we immediately have that $\langle\langle G \rangle\rangle a$ implies that $a \in \mathsf{act}(G)$.

For negative literals $\neg a$, we have with axiom 7 that $\langle\langle G \rangle\rangle \neg a$ implies that $a \in \mathsf{act}(G)$. Moreover, for the negative literals we can use axiom 2 to conclude that $a \notin \mathsf{act}(Ag \setminus G)$.

But this means that $\mathsf{act} \models^a \langle\langle G \rangle\rangle \bigwedge \Phi$, and since $\bigwedge \Phi \rightarrow \alpha$, that $\mathsf{act} \models^a \langle\langle G \rangle\rangle \alpha$. □

2.3 A multi-agent system – agents acting in an environment

In the previous two subsections we defined a module describing an environment, and a separate module describing the actions that agents can choose to perform, either by themselves or together. These two modules can be related by identifying the set of actions in the two respective modules. Such a combination provides us with a semantics for reasoning about agents that act in an environment by way of performing certain actions.

Formally, a multi-agent system is an environment together with an agent module that shares its action repertoire.

DEFINITION 12 (MULTI-AGENT SYSTEM).
A multi-agent system MaS is a tuple

$$\langle S, Ac, (\rightarrow)_{A \subseteq Ac}, P, \pi, Ag, \mathsf{act} \rangle$$

where:

1. $\langle S, (\rightarrow_{A'})_{A \subseteq Ac}, P, \pi \rangle$ *is an environment (in the sense of definition 1).*
2. $\langle Ac, Ag, \mathsf{act} \rangle$ *is an agent module (in the sense of definition 8).*

Of course, everything that we could express with the languages we used to talk about the environment and about the capabilities of the agents remains interesting in the combined system as well, so a logic for reasoning about a multi-agent system should include both. But in the new, more complex system, there are other notions of interest that did not make sense in the separate modules.

In particular, we are interested in the more standard operators of coalition logic that reason not about ability to enforce complex *actions*, but ability to enforce certain *results*. We will overload the operators $\langle\langle G \rangle\rangle$ for this purpose, and add sentences of the form $\langle\langle G \rangle\rangle \phi$ to the language, where ϕ can be any sentence. Intuitively, $\langle\langle G \rangle\rangle \phi$ means that the agents in G have the power to obtain a state where ϕ is true.

This leads to the following definition of a language for multi-agent systems:

$$\phi ::= p \mid \phi \wedge \phi \mid \neg \phi \mid [\alpha]\phi \mid \langle\langle G \rangle\rangle \alpha \mid \langle\langle G \rangle\rangle \phi$$

where $p \in P$, and α is an expression of the action language, as in definition 2. The satisfaction of formulae of the type $\langle\langle G \rangle\rangle \phi$ is defined as follows.

DEFINITION 13 (ABILITY).
MaS, $s \models^m \langle\langle G \rangle\rangle \phi$ *iff there is a set of actions* $A \subseteq \mathsf{act}(G, s)$ *such that for all* $B \subseteq \mathsf{act}(Ag \setminus G)$ *and for all states* t, *if* $s \rightarrow_{A \cup B} t$, *then MaS*, $t \models^m \phi$.

So, $\langle\langle G \rangle\rangle \phi$ is true in a certain state just in case the agents in G are able to execute a concurrent action that guarantees, independently from what the other agents do, the truth of ϕ. As for the environment module, *MaS* $\models^m \phi$ means that for all $s \in S$, *MaS*, $s \models^m \phi$.

The following proposition provides a precise link between the ability to obtain a certain state of affairs, and the ability to execute an action that is guaranteed to result in such a state. It states that a group has the ability to enforce the truth of a sentence just in case it is able to enforce a concurrent action that is guaranteed to result in a state in which that sentence is true:

OBSERVATION 14. *Given a multi-agent system MaS and a state* s *of its environment:*
MaS, $s \models^m \langle\langle G \rangle\rangle \phi$ *iff there exists an action expression* α *with MaS*, $s \models^m \langle\langle G \rangle\rangle \alpha$ *and MaS*, $s \models^m [\alpha]\phi$

proof: Assume that *MaS*, $s \models^m \langle\langle G \rangle\rangle \phi$, and let A be a witness for this fact. The action expression $\bigwedge \Phi(A, G)$ (the set of sentences we used in the proof of proposition 11) is the witness we are looking for: it holds that *MaS*, $s \models^m \langle\langle G \rangle\rangle \bigwedge \Phi(A, G)$, and that *MaS*, $s \models^m [\bigwedge \Phi(G, A)]\phi$.

For the other direction, let α be such that *MaS*, $s \models^m \langle\langle G \rangle\rangle \alpha$ and *MaS*, $s \models^m [\alpha]\phi$. By definition it follows from *MaS*, $s \models^m \langle\langle G \rangle\rangle \alpha$ that there must be a set of actions $A \in \mathsf{act}(G, s)$ such that $A \cup B \models^m \alpha$ for each $B \subseteq \mathsf{act}(Ag \setminus G, s)$. Take this A, and take any $B \subseteq \mathsf{act}(Ag \setminus G)$ and t such that $s \rightarrow_{A \cup B} t$. Since $A \cup B \models^p \alpha$ and *MaS*, $s \models^m [\alpha]\phi$, it must hold that $t \models^m \phi$. So *MaS*, $s \models^m \langle\langle G \rangle\rangle \phi$. □

The logic for the multiagent system obviously contains all axioms we already had from the two separate models. To obtain a complete axiom system, it suffices to add just two axioms that relate the ability to obtain a state of affairs with the ability to execute concurrent actions.

DEFINITION 15 (COOPERATION LOGIC WITH ACTIONS). *The cooperation logic with actions, $\vdash^m$, is given by:*

1. *All axioms and rules from the environment logic of section 2.1*
2. *All axioms and rules from the agent logic of section 2.2*
3. $(\langle\langle G\rangle\rangle\alpha \wedge [\alpha]\phi) \rightarrow \langle\langle G\rangle\rangle\phi$ *for each* α
4. $\langle\langle G\rangle\rangle\phi \rightarrow \bigvee\{\langle\langle G\rangle\rangle\alpha \wedge [\alpha]\phi \mid \alpha$ *is the conjunction of a set of action literals* $\}$

where 'action literals' are atomic actions a or their negations $\neg a$.

The two new axioms relate the ability to execute actions with the ability to enforce the truth of propositions, in the same way as is done in observation 14.

PROPOSITION 16 (SOUNDNESS AND COMPLETENESS). *Cooperation logic with actions is sound and complete with respect to action models:*

$$\vdash^m \phi \quad \textit{iff} \quad \textit{MaS} \models^m \phi \textit{ for all models MaS}$$

proof: Soundness follows from the soundness of the two modules and the previous observation 14.

Completeness is relatively straightforward with the completeness results of sections 2.1 and 2.2. We first construct a model *MaS* combining the constructions of the previous proofs in the straightforward way.

We need to prove a truth lemma stating that $\textit{MaS}, \Sigma \models^m \phi$ iff $\phi \in \Sigma$. This is proven by induction on ϕ, leaving the proofs of propositions 11 and 7 practically unchanged. The only interesting case is the new operator. So suppose that

ϕ is of the form $\langle\langle G\rangle\rangle\psi$

Suppose $\textit{MaS}, \Sigma \models^m \langle\langle G\rangle\rangle\psi$. With observation 14, there is an action expression α such that $\Sigma \models^m \langle\langle G\rangle\rangle\alpha$ and $\Sigma \models^m [\alpha]\psi$. By induction hypothesis and the previous two cases, we have that $\langle\langle G\rangle\rangle\alpha \in \Sigma$ and $[\alpha]\phi \in \Sigma$, and therefore, with our axiom 3 and the fact that Σ is maximal, that $\langle\langle G\rangle\rangle\phi \in \Sigma$.

For the other direction, assume that $\langle\langle G\rangle\rangle\psi \in \Sigma$. Then, with the axiom 4 and maximality of Σ, there is a set of action literals Φ such that $[\bigwedge \Phi]\psi \in \Sigma$ and that $\langle\langle G\rangle\rangle \bigwedge \Phi$ in Σ. We can conclude (by induction hypothesis and the previous cases) that $\Sigma \models^m [\bigwedge \Phi]\psi$ and $\Sigma \models^m \langle\langle G\rangle\rangle \bigwedge \Phi$. We then use observation 14 again to conclude that $\Sigma \models^m \langle\langle G\rangle\rangle\psi$. □

Summarising, we have defined an environment as a Set-Labelled Transition System, and defined a PDL-like logic for representing the performance of complex actions in such an environment. We gave a sound and complete logic for this logic. We then defined agents as actors that can choose to do certain sets of actions, and defined a sound and complete logic for reasoning about the ability of groups of agents to enforce that a certain type of concurrent action is executed. We then defined a notion of cooperative ability. This *modularisation* of a multi-agent system paid off: it turned out to be relatively easy to find a sound and complete axiomatisation for the combination of these two logics.

3. AN EXAMPLE

Often in a multi-agent system, certain variables can be changed only when two or more agents cooperate, while other variables can be changed by the agents acting by themselves. For example, a database may contain crucial data that can be changed only with the consent of all operators, while data of a lesser importance can be changed by any agent. For another example, a website may accept a request for information without further ado, but for accepting a transaction with a credit card, it will need the consent of both the customer and the credit card company. Similarly, a website such as Ebay will accept advertisements that are submitted by a single agent acting alone, but will not accept a transaction if this is not confirmed by all parties involved.

All of these examples are instances of a case in which certain variables are under control of single agents, while other variables can only be changed by groups of agents acting together. We will consider an instance of this schema, in which two agents control two variables. To make the example concrete, we will talk about prison guards that control two doors (see Figure 1). A prison is composed of two interiors, a dormitory and a courtyard. One door opens to the outside, and it is important that it is not opened by accident. It can therefore only be opened if both guards act together; if it is open, each of the agents can close it at will. The other door separates the cell block from the courtyard, and each guard can open that door by himself. The fact that the inner gate is open is expressed by a proposition $\mathsf{in\_open}$; if the outer gate is open, $\mathsf{out\_open}$ is true.

The state of the gates is ruled by four buttons: a_1, b_1, a_2 and b_2. The gate $\mathsf{in\_open}$ toggles its state if and only if at least one of a_1 and a_2 is pressed. The outer gate $\mathsf{out\_open}$ toggles its state if and only if both b_1 and b_2 are pressed.

This description can be captured by an environment model *Env* with four states determined by the fact whether the gates are open or not. For example, we will write 01 for the state in which the inner door is closed and the outer door is open. Each of the buttons corresponds to an atomic action in the model, and the transitions are as described (so, for example, it will hold that $01 \rightarrow_{\{a_2,b_2\}} 11$).

With our environment language, we can express properties of the environment such that the fact that the action $a_1 \wedge b_1 \wedge b_2$ ensures, in the state 00, that in the next state both the gates will be open:

$$\textit{Env}, 00 \models^e [a_1 \wedge b_1 \wedge b_2](\mathsf{in\_open} \wedge \mathsf{out\_open})$$

Similarly, the fact that if the outer gate is closed, the action $b_1 \wedge b_2$ is both a sufficient and necessary condition to open the outer gate is captured by:

$$\textit{Env} \models^e \neg\mathsf{out\_open} \rightarrow ([b_1 \wedge b_2]\mathsf{out\_open} \wedge [\neg(b_1 \wedge b_2)]\neg\mathsf{out\_open})$$

Two guards, gd_1 and gd_2, control the buttons. The first guard gd_1 has access to the buttons a_1 and b_1, whereas gd_2 has access to a_2 and b_2. This situation can be captured by an agents model with the same set of atomic actions as the environment, and a function act such that $\mathsf{act}(gd_1) = \{a_1, b_1\}$ and $\mathsf{act}(gd_2) = \{a_2, b_2\}$.

We can express that the action b_1 can be executed only by the guard gd_1 as $\langle\langle gd_1\rangle\rangle b_1 \wedge \neg\langle\langle gd_2\rangle\rangle b_1$ (strictly speaking, we should have written $\langle\langle gd_1\rangle\rangle$ instead of $\langle\langle\{gd_1\}\rangle\rangle$, but we are omitting the curly brackets for readability).

Together, the agents can push both b_1 and b_2, that is, $\mathsf{act} \models^a \langle\langle gd_1, gd_2\rangle\rangle(b_1 \wedge b_2)$. Analogously, both of them can, individually, execute the disjunction of a_1 and a_2: $\mathsf{act} \models^a \langle\langle gd_1\rangle\rangle(a_1 \vee a_2) \wedge \langle\langle gd_2\rangle\rangle(a_1 \vee a_2)$.

Combining the environment and the agents modules in multiagent system *MaS*, we obtain a model of the guards controlling the prison.

For example, since $b_1 \wedge b_2$ is a necessary condition to open the outer gate, and each guard gd_i can prevent the execution of b_i, it follows that both guards are needed to open the outer gate. Indeed, it holds that $\textit{MaS} \models^m \neg\mathsf{out\_open} \rightarrow (\neg\langle\langle gd_1\rangle\rangle\mathsf{out\_open} \wedge$

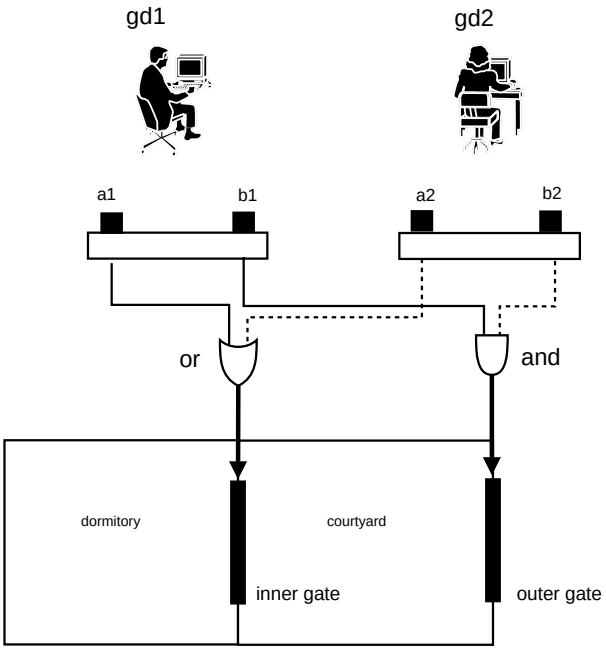


Figure 1: ***The prison example.***

$\neg\langle\langle gd_2\rangle\rangle$out_open)). Together, they can, indeed, open the door: *MaS* $\models^m \langle\langle gd_1, gd_2\rangle\rangle$out_open. To escape, the prisoner will need to take both of the guards hostage to open the outer door.

On the other hand, we want to be sure that when either one of the guards notes that there's a fire in the dormitory, he can open the inner gate to let the prisoners into the courtyard. Since both the guards are able to execute at least one of the actions a_1 and a_2, act $\models^a \langle\langle gd_1\rangle\rangle(a_1 \vee a_2) \wedge \langle\langle gd_2\rangle\rangle(a_1 \vee a_2)$, and an action of type $a_1 \vee a_2$ opens the inner gate, *Env* $\models^e \neg$in_open $\rightarrow [a_1 \vee a_2]$in_open, each of the guards has the power to open the inner gate if required:

$$\textit{MaS} \models^m \neg\mathsf{in\_open} \rightarrow (\langle\langle gd_1\rangle\rangle\mathsf{in\_open} \wedge \langle\langle gd_1\rangle\rangle\mathsf{in\_open}).$$

4. RICHER SYSTEMS

We have studied a system in which 'plans' of agents to obtain certain goals (i.e., complex action expressions) never look more than one transition ahead. There are different ways of enriching the system. First of all, one may enrich the model itself, and, for example, take a more sophisticated view of agents, taking the information they have and/or their goals into account as well. But also if we leave the semantical model the way it is, there are ways to enrich the language to allow us to express other interesting properties of a system.

One obvious choice is to enrich the language on the level of the multi-agent system with other operators borrowed from ATL [1].

Let a strategy for a group of agents G be a function that assigns to each state a subset of actions from act(G). A strategy for G together with a strategy for the agents not in G defines a unique path through the model, and we can define logical operators that not only talk about those states that can be reached by performing a single action, but also about strategies that guarantee that a sentence will always remain true, or will eventually become so:

1. *MaS*, $s \models \langle\langle G\rangle\rangle\Box\phi$ iff there is a strategy for the agents in G such that, no matter what the other agents do, ϕ will be true in all the states along the resulting paths.
2. *MaS*, $s \models \langle\langle G\rangle\rangle\Diamond\phi$ iff there is a strategy for the agents in G such that, no matter what the other agents do, will eventually result in a state where ϕ is true.

What we want to do is 'decompose' these notions into independent statements about the environment and about the agent module, to obtain a theorem similar to that of observation 14. We will focus on the second construct: the ability of a coalition to *eventually* bring about some state of affairs.

To express the type of actions that guarantee that some state will obtain at some point arbitrarily far in the future, we need a richer action language. A more program-like language is suitable here, in which we can express things like 'continue doing α until ϕ is true.' One well-studied set of operators for expressing these kinds of properties are those of Propositional Dynamic Logic [7]. We obtain an action repertoire like the following:

$$\alpha, \beta ::= \gamma \mid ?\phi \mid \alpha;\beta \mid \alpha \cup \beta \mid \alpha^*$$

where γ is a simple action expression (a Boolean combination of atomic actions), as in definition 2 and ϕ is a sentence as in definition 4. These operators can be given their standard semantics, which we do not repeat here.

We will call these types of action expression 'plans', in conformance with the usage we make of them. Now, what we are looking for is the following type of 'decomposition' of the meaning of $\langle\langle G\rangle\rangle\Diamond\phi$ — which is defined in terms of strategies — into a statement about the ability to execute a certain action and a statement about the effect of this action, in a way that is similar to observation 14:

Let MaS consist of an agent module act *and an environment module Env. Then:*

MaS, $s \models \langle\langle G\rangle\rangle\Diamond\phi$ *iff there is a plan* α *such that* G *has the ability to perform* α *(in the action module* act*), and* α *is guaranteed to reach a state where* ϕ *is true (in the environment module Env).*

With action expressions to express complex plans, then, we need a way of expressing that a plan is guaranteed to result in a state where ϕ is true. Somewhat disappointingly, PDL does not provide us with a way of doing that: the expression $[\alpha]\phi$ captures that all halting executions end up in a ϕ-state. The problem is that α might not halt at all, in which case ϕ may never become true, even if $[\alpha]\phi$ is.

One way of remedying this lack of expressive power is to add a new predicate to the language to obtain sentences of the form halt(α), that are true in a state s exactly when all executions of α are guaranteed to halt.

We also need to extend the language of the agent module of the previous section with expressions that express the ability to perform these complex actions. If we assume that all agents can observe all static sentences, the following definitions are reasonable:

1. act $\models \langle\langle G\rangle\rangle?\phi$ (since agents can observe anything)
2. act $\models \langle\langle G\rangle\rangle\alpha;\beta$ iff act $\models \langle\langle G\rangle\rangle\alpha$ and act $\models \langle\langle G\rangle\rangle\beta$
3. act $\models \langle\langle G\rangle\rangle\alpha \cup \beta$ iff act $\models \langle\langle G\rangle\rangle\alpha$ and act $\models \langle\langle G\rangle\rangle\beta$
4. act $\models \langle\langle G\rangle\rangle\alpha^*$ iff act $\models \langle\langle G\rangle\rangle\alpha$

Putting all of this together, we obtain the following conjecture:

CONJECTURE 17. *MaS*, $s \models \langle\langle G\rangle\rangle F\phi$ *iff there is a plan* α *(in the language extended with PDL-operators) such that MaS*, $s \models$

$[\alpha]\phi$ *(if* α *halts, it is in a* ϕ*-state, MaS,* $s \models \langle\alpha\rangle\top$ *(*α *is executable), MaS,* $s \models \mathsf{halts}(\alpha)$ *(*α *is guaranteed to halt), and MaS,* $s \models \langle\langle G\rangle\rangle\alpha$ *(the agent in* G *are able to enforce* α*).*

5. CONCLUSIONS

Many logic-based formalisms – cooperation logics – have been developed in order to represent the power of groups of agents to achieve certain states of affairs.

Pauly [12] provides a modal logic, Coalition Logic, which axiomatises α-ability – a notion of power developed in game theory. A state of affairs is denoted as a set of states that satisfy a propositional formula ϕ and, informally, a group of agents G is α-effective for ϕ if and only if there exists a joint strategy for G which, no matter what the other agents do, ensures the satisfaction of ϕ. Another example is CL-PC, van der Hoek et al. [15]. In CL-PC each agent controls the truth value of a set of variables. In this way, the notion of α-ability can be formalised a standard multi-modal Kripke model and the usual semantics for box and diamond operators can be used. Finally another well-known logic to reason about the α-ability is ATL, in which the expressivity of the previous approaches is enriched with CTL temporal formulas [1].

All these approaches directly describe the achievement power of groups of agents without explicitly representing what they actually do in order to achieve a certain state of affairs. This fact has two drawbacks. First, it may not be easy to reason about meta-information such as the costs required to achieve a certain state of affairs or how these costs are distributed. Second, even if these logics correctly describe whether a group of agents has the power achieve a state of affairs, they do not provide, as required in cooperative problem solving or social reasoning mechanism, any clue about which group of agents has the power to achieve a desired state of affairs.

Thus, inspired by work on cooperative problem solving [16] and social reasoning mechanisms [13], we have formalised a notion of power in two modules: in the first module, the environment module, we addressed the problem to describe a causal model of an environment in which several agents act at the same time. In this module it is possible to describe the fact that a plan, intended as a set of concurrent actions, assures a certain state of affairs. In the second module, the agents module, the capabilities of the agents are modeled and in particular the possibility for a group of agents to execute a plan no matter what the other agents do. Combining these two modules provides a straightforward way to describe the achievement power a group of agents: a group of agents has the power to achieve a state of affairs ϕ if there exists a plan α assuring ϕ and the group can execute α without being obstructed by the other agents.

We have provided a complete axiomatisation for the two separated modules as well as for their combination. We noted that the axiomatisation of the combined module is straightforward, given the completeness of the two logics of the underlying modules. This is evidence that the modularisation of the notion of power is possible.

Finally we also provided an indication of how to enrich the expressiveness of our framework, maintaining the modularisation uninjured, with operators borrowed from Dynamic Logic. Our future work is devoted to provide formal results in this sense.

6. REFERENCES

[1] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of ACM*, 49(5):672–713, 2002.

[2] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*, 1982.

[3] G. Gargov and S. Passy. A note on boolean modal logic. In P. Petkov, editor, *Mathematical Logic, Proc. of Heyting88*, pages 311–321. Plenum Press, 1990.

[4] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers: San Mateo, CA, 2004.

[5] Laura Giordano, Alberto Martelli, and Camilla Schwind. Dealing with concurrent actions in modal action logic. In Henri Prade, editor, *ECAI 98. 13th European Conference on Artificial Intelligence*, 1998.

[6] D. Harel. Dynamic logic. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic Volume II — Extensions of Classical Logic*, pages 497–604. D. Reidel Publishing Company: Dordrecht, The Netherlands, 1984.

[7] David Harel, Jerzy Tiuryn, and Dexter Kozen. *Dynamic Logic*. MIT Press, Cambridge, MA, USA, 2000.

[8] D. Kinny, M. Ljungberg, A. S. Rao, E. Sonenberg, G. Tidhar, and E. Werner. Planned team activity. In C. Castelfranchi and E. Werner, editors, *Artificial Social Systems — Selected Papers from the Fourth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-92 (LNAI Volume 830)*, pages 226–256. Springer-Verlag: Berlin, Germany, 1992.

[9] B. van Linder, W. van der Hoek, and J.J-Ch. Meyer. Formalizing abilities and opportunities of agents. *Fundamenta Informaticae*, 34(1,2):53–101, 1998.

[10] K.L. McMillan. *Symbolic Model Checking*. PhD thesis, CMU University, 1992.

[11] R. C. Moore. Reasoning about knowledge and action. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, Cambridge, MA, 1977.

[12] M. Pauly. A Modal Logic for Coalitional Power in Games. *Journal of Logic and Computation*, 12:146–166, 2002.

[13] J. S. Sichman and Y. Demazeau. On social reasoning in multi-agent systems. *Revista Iberoamericana de Inteligencia Artificial*, 13:68–84, 2001.

[14] M. P. Singh. *Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications (LNAI Volume 799)*. Springer-Verlag: Berlin, Germany, 1994.

[15] W. van der Hoek and M. Wooldridge. On the logic of cooperation and propositional control. *Artificial Intelligence, 64:1-2, pp. 81–119.*, 64(1-2):81–119, 2005.

[16] M. Wooldridge and J. Jennings. Towards a theory of cooperative problem solving. In *Procs. of the Sixth European Workshop on Modelling Autonomous Agents in Multi-Agent Worlds (MAAMAW-94)*, 1994.

Logics of Propositional Control

Jelle Gerbrandy
Dipartimento di Informatica
Torino, Italia
jelle@gerbrandy.com

ABSTRACT

The 'Cooperation Logic of Propositional Control', CL-PC, of van der Hoek and Wooldridge is a logic for reasoning about the ability of agents and groups of agents to obtain a certain state of affairs in a situation in which each of the agents controls a number of propositional variables.

We present a number of generalizations of this model, to represent situations in which agents only partially control the value of a variable, or cases in which agents share the control of a variable. We discuss and axiomatize some of these logics of 'partial control.' In addition, we show how this family of logics are closely connected to a body of work in mathematical logic: Cylindric Modal Logic.

Categories and Subject Descriptors

F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*modal logic, model theory*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*coherence and coordination, multiagent systems*

General Terms

Theory

Keywords

agent and multi-agent architectures, cooperation and coordination among agents, logics for agent systems

1. INTRODUCTION

Cooperation logics are logics for reasoning about what agents and groups of agents in a multi-agent system can obtain. Formal models for cooperation logics have recently gained some attention; examples are Coalition Logic [10, 9], and ATL [1]. These logics provide ways of talking about what agents, alone or in groups, can obtain with or without the cooperation of the other agents.

A cooperation logic for a specific kind of multi-agent system is the *Coalition Logic of Propositional Control* (CL-PC) from [13]. CL-PC is a logic for reasoning about a situation in which a number of binary control variables is allocated to a group of agents. Given a certain *state* – that is, given that the values of the variables are set in a certain way – agents can then change the variables allocated to them to change the state into a new one.

One of the observations we make in this paper is that the CL-PC model is naturally seen as a *game* (in the game-theoretical sense): the model concerns a set of agents that each can choose between a number of *strategies* (a strategy is for an agent is a valuation for the propositional variables allocated to it), and a choice of strategy for each of the agents determines the final result (a valuation over all variables). Seen as a game, it is quite natural to generalize the kind of model used in CL-PC, which is exactly what we will do.

The model for CL-PC proposed by [13] is about a particular case of propositional control: each variable is controlled by exactly one agent, and all variables can be controlled independently of each other. It is natural to weaken one or both of these assumptions, and consider models in which several agents share the control over the same variables, or cases in which one cannot change a certain variable without changing another as well.

The paper consists of two parts. In section 2, we introduce a general model for reasoning about situations in which propositions get allocated to a number of agents, whose choices with regard to the values of 'their' variables determine their final value. We will call such situations 'games of propositional control.' We show how the logical formalism used in CL-PC can be extended to apply to this larger class of games in a straightforward way. The result of this exercise gives us a modal logic that has been studied in mathematical logic as 'cylindric modal logic' This is a useful observation as it gives us a number of properties that are shared by all logics of propositional control.

In section 3, we discuss a number of particular types of propositional games. One of these is, of course, the model for CL-PC from [13], but we will also consider other types of games, in which agents may *share* the control over certain variables, or games in which the choices of the agents are restricted in certain ways. We will give a number of sound and complete axiomatizations for reasoning for this type of games.

2. GAMES OF PROPOSITIONAL CONTROL

There are different ways of defining a model for representing the control of agents over a given set of propositional variables. One general schema is to define such a situation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

as a *game*. A specification of a game of propositional control, then, starts with an *allocation* of propositional variables to each of the agents. The agents each choose a valuation for the variables that are allocated to them; the *strategies* of an agent are valuations over the variables that he controls.

Because the valuations chosen by each of the agents may not determine the final values in an unambiguous way (because the choices may conflict), there is an *outcome function* that gives us, for each combination of choices of strategies, an *outcome*, which is a valuation for all propositional variables. Formally, we define a game of propositional control as:

DEFINITION 1. *A* game of propositional control *is a tuple*

$$\mathcal{M} = \langle \mathsf{Ag}, \mathsf{At}, \mathsf{alloc}, \mathsf{strategy}, \mathsf{outcome} \rangle$$

Where:

- $\mathsf{Ag} = \{1, 2, \ldots n\}$ *is a finite, non-empty set of agents*
- At *is a set of propositional variables (or 'atoms').*
- alloc *is an* allocation function *that assigns to each agent from* Ag *a subset of variables from* At.
- *A* strategy *for a player i is a valuation over the propositional variables in* $\mathsf{alloc}(i)$ *– that is, a function that assigns to each propositional variable in* $\mathsf{alloc}(i)$ *a truth value.*
- *The function* strategy *assigns a set of available strategies to each agent i.*
- *We say that a* state *of the game is a tuple $s = (s_1, \ldots s_n)$ consisting of a strategy s_i from* $\mathsf{strategy}(i)$ *for each agent i.*
- outcome *is an* outcome function *that assigns to each state s a valuation over all the propositional variables in* At.

As an example, consider a case with two agents, a website and a telephone operator, that together control a database with customer data of some company. Say that a variable credit_ok represents the information that the credit card rating of mr. X is fine, and another variable, webcustomer, represents the fact that mr. X has placed an order over the internet.

The variable credit_ok is controlled by both agents, in the sense that if either the telephone operator or the website get the information that there is something wrong with mr X's credit rating, they can change the value of credit_ok to 0. The credit rating can only be set to 1 if both agents agree. The variable webcustomer is completely under control of the website.

This situation can be represented as a game of propositional control. A game with two players is represented naturally as a matrix, in the following way:

website\ operator	credit_ok : 1	credit_ok : 0
credit_ok : 1 webcustomer : 1	credit_ok : 1 webcustomer : 1	credit_ok : 0 webcustomer : 1
credit_ok : 1 webcustomer : 0	credit_ok : 1 webcustomer : 0	credit_ok : 0 webcustomer : 0
credit_ok : 0 webcustomer : 1	credit_ok : 0 webcustomer : 1	credit_ok : 0 webcustomer : 1
credit_ok : 0 webcustomer : 0	credit_ok : 0 webcustomer : 0	credit_ok : 0 webcustomer : 0

Here, the top row describes the strategy choices of the telephone operator, while the left column describes the options for the website. The outcome of their choices is represented by the values in the cells of the matrix.

Suppose, for example, that the present state of the environment is given by the top-right box in the example above — the telephone operator did not approve the credit card rating of a web customer, but the website does — then the operator is able to change the value of credit_ok to 1 by changing her 'strategy.' However, she does not have this possibility if the present state is the one given by the outcome on the third row on the right: even if the values in the database are the same here, the telephone operator cannot, by herself, change the rating of the customer, because the website disapproves. They can, however, change the credit card rating *together* – by both changing the value of credit_ok to 1.

To reason about these kind of scenarios in a formal way, we use formulas of a simple modal language. Formulas of coalition logic are constructed from a given set of agents Ag and a given set of atomic variables At.

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi \wedge \psi \mid \langle C \rangle \phi$$

Where $p \in \mathsf{At}$ and $C \subseteq \mathsf{Ag}$. The expression $\langle C \rangle \phi$ is a cooperation modality. The idea is that $\langle C \rangle \phi$ is true at some state just in case the agents in C can, by manipulating the values of the variables under their control, reach a state in which ϕ is true. We write $[C]$ for the dual of $\neg\langle C \rangle \neg$ of $\langle C \rangle$. The sentence $[C]\phi$ is true in case the agents in C cannot manipulate their variables in such a way that $\neg\phi$ becomes true: in other words, when the agents in C have no way of changing the truth value of ϕ from true to false.

DEFINITION 2 (INTERPRETATION).

$$\begin{array}{rcl}
\mathcal{M}, s \models p & \textit{iff} & \mathsf{outcome}(s)(p) = \mathsf{t} \\
\mathcal{M}, s \models \neg\phi & \textit{iff} & \mathcal{M}, s \not\models \neg\phi \\
\mathcal{M}, s \models \phi \wedge \psi & \textit{iff} & \mathcal{M}, s \models \phi \textit{ and } \mathcal{M}, s \models \psi \\
\mathcal{M}, s \models \langle C \rangle \phi & \textit{iff} & \textit{there is a state } t \textit{ such that} \\
 & & s(i) = t(i) \textit{for each } i \notin C, \textit{ and} \\
 & & \mathcal{M}, t \models \phi
\end{array}$$

So, a sentence of the form $\langle C \rangle \phi$ is true at a state s when it holds that the agents in C can unilaterally change their choice of strategy in such a way that in the resulting state, ϕ is true. So, in a precise sense, a sentence $\langle C \rangle \phi$ expresses al *local ability* of a group of agents in a state s: it is true in a state s just in case the agents in C are able to change the situation so that ϕ becomes true.

One of the interesting insights in [13] is that we also express 'global' properties of ability of agents that have been studied in Pauly's Coalition Logic ([9], [10]) and in ATL ([1]) using these 'local' operators.

We say that a group C is α-effective for an outcome ϕ, and write $\langle\langle C \rangle\rangle \phi$, when the players in the group can orchestrate their strategies in such a way that no matter what the other players do, ϕ will be true in the outcome.

$$G, s \models \langle\langle C \rangle\rangle \phi \text{ iff } G, s \models \langle C \rangle [\mathsf{Ag} \setminus C] \phi$$

This definition says that a group C is α-effective for ϕ just in case there is a combination of strategies for the agents in C such that, no matter what the agents outside of C (that is, those in the set $\mathsf{Ag} \setminus C$) choose to do, the resulting state will always be one where ϕ is true.

The notion of α-ability is a *global* notion, in the sense that the truth of a sentence $\langle\langle C\rangle\rangle\phi$ is independent of the state in which it is evaluated — it either is true in all states in a game, or in none of them. So, the fact whether $\langle\langle G\rangle\rangle\phi$ is true expresses a property of the whole game rather than of a particular state in it.

To turn back to our example: we saw that the website had the power to change the value of the variables credit_ok at will. Indeed, the sentence $\langle\langle$website$\rangle\rangle$webcustomer and the sentence $\langle\langle$website$\rangle\rangle\neg$webcustomer are both true. Neither the website nor the operator is α-effective to approve the credit card rating of mr. X: $\neg\langle\langle$website$\rangle\rangle$credit_ok holds, and so does $\neg\langle\langle$operator$\rangle\rangle$credit_ok, even if, for example, $\langle$website$\rangle$credit_ok is true in some states. However, together they are effective for setting credit_ok to 1, which is captured by the truth of $\langle\langle\{$website, operator$\}\rangle\rangle$credit_ok.

2.1 The logic of games of propositional control

It turns out that the logic of games of propositional control is located at an intersection point between different areas of mathematical logic: it is the logic of finite variable fragments of first order logic without identity, which is closely related to diagonal-free cylindric algebra [15, 14, 8], while its models are products of $S5$-models [3]. For this reason, it is relatively well-studied.

From this work, we know that the logic of propositional games is not finitely axiomatizable. Moreover, the logic is not decidable. There is, however, a sound and complete axiomatization, but it uses a non-standard rule. These are negative results, but the logic of propositional games is still of interest, because it can serve as a kind of 'minimal logic of propositional control': we know that the logic of any particular subclass of models (such as CL-PC) will include this this logic, so we will discuss its main features.

The following set of axioms and rules is shown to be sound and complete for the class of all propositional games from [8] for the language of cylindric algebra, which [15] translates into modal logic. (In the following, if C is a singleton set $\{i\}$, we will write $\langle i\rangle$ instead of $\langle\{i\}\rangle$.)

DEFINITION 3 (LOGIC OF PROPOSITIONAL GAMES).

propositional logic *all validities of propositional logic*

reduction $\langle\{i_0,\ldots,i_m\}\rangle\phi \leftrightarrow \langle i_0\rangle\ldots\langle i_m\rangle\phi$

asynchronicity $\langle i\rangle\langle j\rangle\phi \rightarrow \langle j\rangle\langle i\rangle\phi$

normality $[i](\phi\rightarrow\psi)\rightarrow([i]\phi\rightarrow[i]\psi)$

reflexivity $\phi\rightarrow\langle i\rangle\phi$

symmetry $\phi\rightarrow[i]\langle i\rangle\phi$

transitivity $\langle i\rangle\langle i\rangle\phi\rightarrow\langle i\rangle\phi$

Plus the following rules:

$$\frac{\vdash\phi}{\vdash[i]\phi}\qquad\frac{\vdash\phi,\vdash\phi\rightarrow\psi}{\vdash\psi}$$

Define $\tau(\chi)$ to be the sentence $\neg\langle A\rangle[(\bigwedge_{i\in A}\langle i\rangle\chi)\wedge\neg\chi]$. The logic contains the following non-standard rule:

$$\frac{(p\wedge\tau(p\wedge\neg\phi))\rightarrow\phi}{\phi}$$

provided that p does not occur in ϕ

When there are only two agents, this rule can be replaced with the following axiom

confluence $\langle i\rangle[j]p\rightarrow[j]\langle i\rangle p$.

The reduction axiom says that the modality $\langle C\rangle$ can be defined in terms of $\langle i\rangle$-operators for $i\in C$ – the ability of a set of agents can be decomposed in the abilities of the agents separately. The asynchronicity axiom says that the order in which the $\langle i\rangle$-operators are applied does not matter.

The reflexivity axiom states that if ϕ is true, then any agent can 'change' the current state (by doing nothing) to make ϕ true. Note that symmetry, reflexivity and transitivity are known as the $S5$ axioms that are familiar from classical modal logic as the axioms that characterize an equivalence relation.

More complicated is the non-standard rule. A full explanation would lead too far for the purposes of this paper. We refer the reader to the works quoted – and in any case, we will not need the rule for axiomatizing the logics we introduce below.

PROPOSITION 4 (COMPLETENESS [8, 14]). *The logic of definition 3 is sound and complete with respect to the class of games of propositional control defined in definition 1.*

proof: This follows from the completeness results for cylindric modal logic. For some details of how to connect those results to games of propositional control, see the appendix. □

We repeat the following results

PROPOSITION 5 ((UN)DECIDABILITY RESULTS [7, 5]).

- *If the number of agents is less than or equal to two, basic game logic is decidable.*
- *However, if there are more than 2 agents, basic game logic is not decidable.*

2.2 The Pareto Principle

The class of models defined in definition 1 is quite large. Many of the models that fall under the definition are hardly worthy of the name 'propositional control', as there need not be any systematic relation at all between the valuations that the agents choose and the actual outcome that results.

To do justice to the idea that the allocation function in a game of propositional control specifies some kind of control, we need some further constraints. There are many possible properties that are candidates for minimal conditions for a game to be called a game of propositional control, but one in particular stands out: the condition that if all agents that are allocated a given variable p choose the same value for p, then the outcome should reflect that choice.

DEFINITION 6 (PARETO PRINCIPLE).
Let $\mathcal{M}=\langle$Ag, At, alloc, strategy, outcome$\rangle$ be a game of propositional control.

The Pareto principle holds for $\mathcal{M}$ iff for each state $s=(s_0,\ldots,s_n)$ and for any i and j such that $p\in$ alloc(i) and $p\in$ alloc(j) it holds that if $s_i(p)=s_j(p)$, then outcome$(s)=s_i(p)$.

In the rest of this paper, we will only consider games for which the Pareto principle holds.

The Pareto principle is a property that is well known in social choice theory (which is where the name comes from). Social choice theory is concerned with finding a 'fair' way of

defining a preference order among a given set of alternatives, given the preferences of a group of agents (typical examples are political elections and auctions). It is perhaps useful to remark here that we are dealing with a situation that in certain respects is more simple than the topic usually dealt with in social choice theory. Here, we are dealing with variables that can each have only one out of two possible truth values. In particular, Arrow's Theorem, a central result that establishes that a 'fair' social choice function does not exist, does not arise as such within our framework. There are, however, interesting results from social choice theory that apply to games of propositional control such as they are defined here (see, for example, [6] or [11]) that show that given certain assumptions about 'fairness', fair outcome functions do not exist. It seems, however, that these kind of concerns are tangential to the kind of problem that we are concerned with in this paper.

3. LOGICS OF PROPOSITIONAL CONTROL

We have talked about games of propositional control in a very general way. It is now time to be more specific. In the rest of the paper, we will study a number of subclasses of games of propositional control, and provide them with sound and complete axiomatizations. Even if the proofs are fairly involved (they can be found in the appendix), the axiomatizations themselves are relatively straightforward, and have axioms that correspond directly with certain properties of the models.

3.1 Logic of Exclusive control

One particular subclass of games of propositional control is one in which no single variable is allocated to two different players. We will call such games *games of exclusive control*. The kind of game that is studied in CL-PC, which was the main inspiration for this paper, is an example of such a game.

DEFINITION 7 (GAMES OF EXCLUSIVE CONTROL). *Let $\mathcal{M} = (\mathsf{Ag}, \mathsf{At}, \mathsf{alloc}, \mathsf{strategy}, \mathsf{outcome})$ be a game.*

- *An allocation function* alloc *is* exclusive *when no two agents have control over the same variable, i.e. when* $\mathsf{alloc}(i) \cap \mathsf{alloc}(j) = \emptyset$ *whenever* $i \neq j$.
- *A* game of exclusive control *is a game for in which the Pareto principle holds, and that based on an exclusive allocation function* alloc.
- *A* game of actual control *is a game in which the allocation of a variable p to an agent i implies that i has a choice available where p is false and one where p is true: for each i and $p \in \mathsf{alloc}(i)$, a strategy s_i such that $s_i(p) = 1$, and a strategy s'_i such that $s'_i(p) = 0$.*
- *A* game of full control *is a game where* $\mathsf{strategy}(i)$ *contains a strategy for* each *valuation of* $\mathsf{alloc}(i)$.

A game of actual control is a game in which each agent, for each propositional variable p allocated to him, has at least one choice available where p is true and one where p is false. In a game of full control each player has all combinations of truth values of propositional variables available to him — this reflects an assumption that the variables can all be set independently of each other.

Note that the outcome function outcome of an exclusive game is determined completely by the choices of the players and the assumption that the Pareto principle holds – since no variable is allocated to two agents, obviously, their choices cannot conflict. Exclusive games of full propositional control are studied in [4] as 'distributed evaluation games', and are the models of CL-PC that are studied in [13] and [12].

EXAMPLE 8. *Suppose a player i can control two bits of a register, represented by the values of the bits p and q. She can set either bit to 0 or to 1, as she likes, but she cannot set both bits to 0. A second player j controls r. We can represent the game in a matrix:*

$i \setminus j$	r	$\neg r$
p, q	p, q, r	$p, q, \neg r$
$p, \neg q$	$p, \neg q, r$	$p, \neg q, \neg r$
$\neg p, q$	$\neg p, q, r$	$\neg p, q, \neg r$

This is an exclusive game of actual control. The players do not have full control, since i has no strategy for $\neg p, \neg q$.

It turns out that logic of games of exclusive control, and their extensions to real and full exclusive control, are fairly straightforward extensions of the minimal game logic of the previous section.

Before we introduce these axioms an abbreviation will be useful. We follow [13] and say that an agent *locally controls* a sentence ϕ iff it holds (in a state) that $\langle i \rangle \phi \wedge \langle i \rangle \neg \phi$. This sentence is true in a state just in case i can unilaterally change her strategy in such a way that ϕ becomes true, but also has a strategy available where ϕ becomes false. We write $\mathsf{controls}(i, \phi)$ for the sentence $\langle i \rangle \phi \wedge \langle i \rangle \neg \phi$

The property of exclusive control is characterized by the following axiom, that states that if an agent i can change the truth value of a propositional variable p, then no other player can do so.

Exclusive Control (EC) $\mathsf{controls}(i, p) \rightarrow \neg \mathsf{controls}(j, p,)$ for $i \neq j$ and $p \in P$.

In games of actual control, there is, for each propositional variable p, at least one player that controls it:

Actual Control (AC) $\bigvee_{i \in A} \mathsf{controls}(i, p)$ for each $p \in P$.

Similarly, in games of full control, if an agent controls all variables from a given set X, then she has the ability to change the outcome in such a way that any consistent boolean combination of the variables in X is satisfied:

Full Control (FC) $\bigwedge_{p \in X} \mathsf{controls}(i, p) \rightarrow \langle i \rangle \phi_v$, where ϕ_v is the conjunction of literals true in a valuation v for X.

We will write, for example, that $EC, FC \vdash \phi$ just in case ϕ is derivable in the logic of games of propositional control of definition 3 from any instance of the axiom schemas EC and FC.

PROPOSITION 9 (COMPLETENESS RESULTS).

$EC \vdash \phi$	*iff*	$G, s \models \phi$ *for all games G of exclusive control.*
$EC, AC \vdash \phi$	*iff*	$G, s \models \phi$ *for all games G of exclusive, actual control.*
$EC, FC \vdash \phi$	*iff*	$G, s \models \phi$ *for all games G of exclusive, full control.*

proof: See the appendix. It is perhaps important to note at this point that neither for this proposition, nor in any of the following ones, we need the full power of the logic of definition 3: we may drop the non-standard rule, and use the confluence axiom instead. □

3.2 Logic of Positive Control

If certain variables are allocated to more than one agent, we need a way of deciding what to do if one agent wants to set a variable to a different value than another. There are lots of options open. If there is an odd number of agents to which the variable is allocated, we can let the majority decide. In other scenarios we may have a constraint that a variable can only be set to a certain value with the consensus of all other players, such as in example in 2 in which the validity of the credit of a customer needs to be judged. In other situations, a constraint may be that if one of the players chooses a strategy in which p is true, then p is true in the outcome of the game.

Let us first consider games of the latter type:

DEFINITION 10 (GAMES OF POSITIVE CONTROL). *A game $\mathcal{M}$ is a* game of positive control *iff the Pareto principle holds, and it holds that if $s_i(p) = 1$ for some i, then $\mathsf{outcome}(s)(p) = 1$.*

Note that also in games of positive control, the Pareto principle implies that the outcome is completely determined by the strategies of the players.

Games of positive control have a particularly natural interpretation if we interpret the elements of the control variables $\mathcal{P}$ as *actions* that may be performed or not. Agents each have a repertoire of actions available that may overlap (or even be equal), and may choose to execute certain subsets of these actions (if we consider games of full control, they can execute any subset). The outcome such a game, then, consists of exactly the actions that have been executed. This behavior corresponds precisely to the definition of a game of positive control.

For example, if i chooses to execute p but not q, and j executes q, then the result will be that both p and q are executed. Such complex or concurrent actions can be described by a propositional formula; for example, $p \wedge \neg q$ is true of an outcome in which p is executed, but not q. The formula $\langle\langle i, j\rangle\rangle(p \wedge \neg q)$ then describes the game in which the group consisting of i and j has the ability to ensure that p is being executed, and to ensure that q is *not* done. In a game of positive control, it holds that $\langle\langle i, j\rangle\rangle(p \wedge \neg q)$ exactly when p is allocated to either i or j, while q is not allocated to any agent other than i or j.

Also positive control is characterized by a straightforward *axiom of positive control*:

Positive Control (POS) $p \rightarrow \bigvee_{i \in \mathsf{Ag}} [\mathsf{Ag} \setminus \{i\}]p$

That this axiom is sound can be seen as follows. If p is true, then by the Pareto principle, there must be an agent, say i, that has chosen a strategy in which p is true. Then, by the definition of a game of positive control, there is nothing that the other players can do about it, so $[\mathsf{Ag} \setminus \{i\}]p$ must be true.

PROPOSITION 11 (COMPLETENESS). *$POS \vdash \phi$ iff $\mathcal{M}, s \models \phi$ in all games of positive control.*

proof: See the appendix. □

3.3 Logic of Consensus Games

The mirror image of the games of positive control are games in which the truth-value of a propositional variable is determined by *consensus*: a propositional variable is true only if all players choose that proposition to be true.

DEFINITION 12 (CONSENSUS GAMES). *A game is a* consensus game *iff $\mathsf{outcome}(s)(p) = 1$ iff $s_i(p) = 1$ for each i such that $p \in \mathsf{alloc}(i)$*

A consensus game might also be called a game of 'negative' control, in which a variable is false in the outcome when at least one of the agents wants it to be false.

Consensus Axiom (CON) $\neg p \rightarrow \bigvee_{i \in N} [\mathsf{Ag} \setminus \{i\}]\neg p$.

The idea is similar to the axiom of positive control: if the outcome is $\neg p$, then, because of the Pareto principle, at least one of the players must have chosen a strategy in which p is false, which implies, by definition of the consensus game, that no matter what the other players will do, p will be false in the outcome.

EXAMPLE 13. *A simple safety mechanism is that two players must both give their consent for some action to be carried out (a customer and a seller confirm some cash transaction; two people lift a heavy table togethere). One very simple way is to describe this by the following game, where each player can either confirm p or not, and the result will be p only if both players give their consent:*

$a \setminus b$	p	$\neg p$
p	p	$\neg p$
$\neg p$	$\neg p$	$\neg p$

The minimal game logic together with the axiom of consensus is sound and complete with respect to all consensus games.

PROPOSITION 14. *$CON \vdash \phi$ iff $\mathcal{M}, s \models \phi$ in all consensus games $\mathcal{M}$*

proof: Analogous to the proof of proposition 11, which can be found in the appendix. □

4. CONCLUSIONS

We have defined a general model of 'propositional control' inspired by the Coalition Logic of Propositional Control of [13], and connected this work with work on mathematical logic (in particular with Cylindric Modal Logic) and with work on 'judgment aggregation' in social choice theory.

We then studied in more detail some particular examples of families of games of propositional control – in particular, we studied different forms of games of exclusive control – in which agents either have some or all strategies available, and provided sound and complete axiomatization for these logics. We also looked at games of 'positive control', in which a vote of one player suffices to make a proposition true, and at 'consensus games' in which a vote of a single player suffices to make a propositional variable false, and provides sound and complete axiomatization for these logics.

A number of open problems and questions are raised by this paper. We mention a few. First of all, there is a general question. Cylindrical Modal Logic, which we proposed as a kind of 'minimal logic of propositional control' is not

decidable, and has a 'strange' axiomatization with a non-standard rule. In contrast, the particular examples of logics we studied in section 3 are much better behaved – they are clearly decidable, and do not need special kinds of axiomatization. It is not really clear why this is the case, or where the 'borderline' is between decidability and undecidability. The answer to this question might lie with the class of games for which the Pareto principle holds true.

A more specific question regards the complexity of the logics discussed in this paper; we only know that the satisfiability and the model checking problems of the logics for games of full exclusive control are PSPACE-complete ([13]). (Satisfiabily and model-checking have the same complexity because of the strong restrictions on the kind of models that are allowed, which means that there are relatively few models to consider in the satisfaction problem. We do not expect the same results for our logics.) At last, there may be other types of logics of propositional control over and above the ones discussed in this paper that might be interesting to study.

To conclude, we list the main completeness results of this paper in a table:

class of games	characterizing axioms
exclusive	EC
exclusive, actual control	EC, AC
exclusive, full control	EC, AC, FC
games of positive control	POS
consensus games	CON

5. REFERENCES

[1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Lecture Notes in Computer Science*, 1536:23–60, 1998.

[2] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Theoretical Tracts in Computer Science. Cambridge University Press, 2001.

[3] D. Gabbay and V. B. Shehtman. Products of modal logics, part i. *Logic Journal of the IGPL*, 6(1):73–146, 1998.

[4] P. Harrenstein. *Logic in Conflict. Logical Explorations in Strategic Equilibrium*. PhD thesis, Utrecht University, 2004.

[5] R. Hirsch, I. Hodkinson, and A. Kurucz. On modal logics between KxKxK and S5xS5xS5. *Journal of Symbolic Logic*, 67:221–234, 2002.

[6] C. List and P. Pettit. Aggregating sets of judgements: An impossibility result. *Economics and Philosophy*, 18:89–110, 2002.

[7] R. Maddux. The equational theory of ca_3 is undecidable. *Journal of Symbolic Logic*, 45(2):311–316, 1980.

[8] S. Mikulas. Gabbay-style calculi. In H. Wansing, editor, *Proof Theory of Modal Logic*, volume 2 of *Applied Logic Series*, pages 243–252. Kluwer Academic Publishers, 1996.

[9] M. Pauly. *Logic for Social Software*. PhD thesis, University of Amsterdam, 2001. ILLC Dissertation Series 2001-10.

[10] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002.

[11] M. Pauly and M. van Hees. Logical constraints on judgment aggregation. *Journal of Philosophical Logic*, 2003. forthcoming.

[12] W. van der Hoek and M. Wooldridge. On the dynamics of delegation, cooperation, and control: a logical account. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 701–708, New York, NY, USA, 2005. ACM Press.

[13] W. van der Hoek and M. Wooldridge. On the logic of co-operation and propositional control. *Artificial Intelligence*, 64(1–2):81–119, 2005.

[14] Y. Venema. Cylindric modal logic. *Journal of Symbolic Logic*, 60:591–623, 1995.

[15] Y. Venema. Rectangular games. *Journal of Symbolic Logic*, 63(4):1549–1564, 1998.

Appendix

In this appendix, we provide some formal background about the relation between games of propositional control and standard models of modal logic, and proofs of the main theorems.

Games of propositional control can be seen as a kind of Kripke models. The following definition makes the connection precise.

DEFINITION 15 (KRIPKE MODEL INDUCED BY A GAME). *Given a game* $\mathcal{M} = \langle \mathsf{Ag}, \mathsf{At}, \mathsf{alloc}, \mathsf{strategy}, \mathsf{outcome} \rangle$ *The* Kripke model induced by M *is a tuple* $(W, (\longrightarrow_i)_{i \in \mathsf{Ag}}, V)$, *where:*

- W *is the set of states of* $\mathcal{M}$ *(that is, the Cartesian product of the sets* $\mathsf{strategy}(i)$*)*
- $s \longrightarrow_i s'$ *iff* $s_j = s'_j$ *for each* $j \neq i$, *and*
- $V(s) = \mathsf{outcome}(s)$.

In the following proofs, we will often confuse a game and the Kripke model induced by it, and write, for example, that $s \longrightarrow_i s'$ in a game $\mathcal{M}$ when s and s' are two states that differ at most in the strategy of i.

This connection between propositional games and Kripke models preserves the truth of sentences.

OBSERVATION 16. *If* s *is a state of a propositional game* $\mathcal{M}$, *and* $\mathcal{K}$ *is the Kripke model induced by* $\mathcal{M}$, *then* $\mathcal{M}, s \models \phi$ *iff* $\mathcal{K}, s \models \phi$ *(using the standard definition of modal logic).*

With these observations, we have a way of transforming games into Kripke models that preserves the truth value of sentences. This gives us a way of applying known results about Kripke models to games of propositional control. In particular, we can tie the logic of games to the following definition and proposition come from [15], [8].

DEFINITION 17 (CUBES).
A cube *is a Kripke model* $(U^{|\mathsf{Ag}|}, (\longrightarrow_i)_{i \in \mathsf{Ag}}, V)$, *with* $s \longrightarrow_i s'$ *iff* $s_j = s'_j$ *for each* $j \neq i$.

PROPOSITION 18 ([15], [8]). *The set of cubes is axiomatized by the minimal logic of propositional control of definition 3.*

To show that the logic of cubes is the same as the logic of games of propositional control, we need to show that each cube corresponds to a game, and each game corresponds to

a cube. The Kripke models induced by a game of propositional control are not precisely cubes, however: they are more like 'rectangles' (if the players number two) or like '(hyper)-bricks' (if there are more than 2 players). 'Cubes' are a subclass of these kind of models; those induced by games in which all players have the same number of strategies available. We will not provide the formal details here, but it is fairly straightforward to transform a Kripke model that is shaped like a 'brick' into an equivalent model that is shaped like proper 'cube' (at least, if the number of agents is finite). Perhaps abusing the geometrical metaphor, one possible way to make a cube from a brick is to add, where needed, a new layer to the 'long' sides of the brick until the model has become a cube – to not affect the validities in the model, this new layer can simply be a copy of the side to which it is added.

In the completeness proofs that follow, we use the notion of a *bisimulation*.

DEFINITION 19 (BISIMULATION). *A pointed Kripke model is a pair (K, s), where K is a Kripke model, and s is a state in K.*

A relation $\sim$ between pointed Kripke models is a bisimulation *iff whenever $(K, s) \sim (K', s')$, it holds that:*

I. $V(s) = V'(s')$

II. If $s \longrightarrow_i t$, then there is a t' with $s' \longrightarrow'_i t'$ and $(K, t) \sim (K', t')$

III. If $s' \longrightarrow'_i t'$, then there is a t with $s \longrightarrow_i t$ and $(K, t) \sim (K', t')$

We say that two pointed Kripke models are bisimilar iff there exists a bisimulation connecting the two models.

A basic result says that two bisimilar Kripke models satisfy the same sentences. If it does not lead to confusion, we will write $s \sim s'$ for $(K, s) \sim (K', s')$. Also, we will be slightly sloppy, and talk about bisimilarity of a Kripke model and a game, meaning that the first Kripke model is bisimilar to the model induced by the game.

PROPOSITION 9. *$EC \vdash \phi$ iff $\mathcal{M}, s \models \phi$ in all games $\mathcal{M}$ of exclusive control.*

proof: Soundness is just a matter of checking that the axioms are valid in a game of exclusive control.

A sketch of the completeness proof before starting: first we construct a canonical Kripke model in the standard way ([2]), then we show that each world in the canonical model is bisimilar to a state in (the Kripke model induced by) a game. This means that all sentences in the maximal consistent set in the canonical model are true in the related game, and thus that any consistent set can be satisfied in some game.

The canonical model $K^{can} = (W, \longrightarrow_i, V)$ has as its domain all maximal consistent sets (mcs) Σ. We set $V(\Sigma)(p) = 1$ iff $p \in \Sigma$, and the accessibility relations are given by $\Sigma \longrightarrow_i \Gamma$ iff for all ϕ: if $[i]\phi \in \Sigma$ then $\phi \in \Gamma$. It is straightforward to prove that the *truth lemma* holds: $K^{\mathsf{can}}, \Sigma \models \phi$ iff $\phi \in \Sigma$.

The interesting part of the proof is the construction of a game G_Σ corresponding to each mcs Σ. First, we define an allocation function alloc_Σ by setting $\mathsf{alloc}_\Sigma(i) = \{p \mid \mathsf{controls}(i, p) \in \Sigma\}$. Because Σ includes the axiom of exclusive control, we know that alloc_Σ is exclusive.

We introduce some notation: if v is a valuation over a finite set X, let ϕ_v be the conjunction of all literals of X true in v. The strategies $\mathsf{strategy}_\Sigma(i)$ of a player i in G_Σ are exactly those valuation functions v on the set $\mathsf{alloc}_\Sigma(i)$ such that $\langle i\rangle\phi_v \in \Sigma$. In the game G_Σ, let s_Σ be the state such that $\phi_{s_\Sigma(i)} \in \Sigma$ for each i. Such a state is guaranteed to exist by the reflexivity axiom.

We claim that there is a bisimulation $\sim$ between maximal consistent sets in the canonical model K^{can} and states in the set of all (Kripke models induced by) exclusive games, given by:

$$\Sigma \sim (G, s) \text{ iff } (G, s) = (G_\Sigma, s_\Sigma).$$

To make our proof run a bit more smoothly, we will write that $(G, s) \longrightarrow_i (G', s')$ iff $G = G'$ and $s \longrightarrow_i s'$ in G. We check the three bisimulation clauses:

I. From the definition of G_Σ and the assumption that the Pareto principle holds for G_Σ, it follows that $\Sigma \models p$ iff $(G_\Sigma, s_\Sigma) \models p$.

II. Suppose that $\Sigma \longrightarrow_i \Gamma$. We claim that $(G_\Sigma, s_\Sigma) \longrightarrow_i (G_\Gamma, s_\Gamma)$. It follows from lemma 21 that $G_\Sigma = G_\Gamma$, and that for each $p \notin \mathsf{alloc}(i)$, $p \in \Sigma$ iff $p \in \Gamma$. That means that $s_\Sigma(j) = s_\Gamma(j)$ for each $j \neq i$, and that therefore $s_\Sigma \longrightarrow_i s_\Gamma$.

III. For the other direction, suppose that $(G_\Sigma, s_\Sigma) \longrightarrow_i (G_\Sigma, s')$. Let $s = s_\Sigma$. By definition of G_Σ, we have that $\langle i\rangle\phi_{s'_i} \in \Sigma$. That means that there is a Γ in the canonical modal such that $\Sigma \longrightarrow_i \Gamma$ and $\phi_{s'_i} \in \Gamma$. By exclusivity and lemma 20, it follows that $[i]\phi_{s_j} \vee [i]\neg\phi_{s_j} \in \Gamma$ for each $j \neq i$. Since by definition, $\phi_{s_j} \in \Sigma$, it follows that $\phi_{s_j} \in \Gamma$, and therefore that $s' = s_\Gamma$.

The proofs of completeness for the logics with the axioms AC and FC is straightforward given these results. □

LEMMA 20 (DERIVED THEOREMS).

1. $\vdash \langle i\rangle[j]\phi \rightarrow [j]\langle i\rangle\phi$ *(confluence)*
2. $EC \vdash \mathsf{controls}(i, p) \rightarrow [j]\mathsf{controls}(i, p)$
   Moreover, if ϕ is propositional and X is the set of propositional variables occurring in ϕ, it holds that:
3. $\bigwedge_{p\in X} \neg\mathsf{controls}(i, p) \vdash [i]\phi \vee [i]\neg\phi$
4. $EC, \bigwedge_{p\in X} \mathsf{controls}(j, p), \langle j\rangle\phi \vdash [i]\langle j\rangle\phi$

proof:

2. In $S5$, it follows from $\mathsf{controls}(i, p)$ that $[i]\mathsf{controls}(i, p)$. Using the axiom EC, we have then that $[i]([j]p \vee [j]\neg p)$, which is equivalent (using reflexivity and propositional logic) to $[i]((p \rightarrow [j]p) \wedge (\neg p \rightarrow [j]\neg p))$. Since we have that $\langle i\rangle p \wedge \langle i\rangle\neg p$, it follows in $S5$ that $\langle i\rangle[j]p \wedge \langle i\rangle[j]\neg p$. Using confluence, we conclude that $[j]\langle i\rangle p \wedge [j]\langle i\rangle\neg p$, and hence that $[j]\mathsf{controls}(i, p)$.

3. Prove this by induction on the structure of ϕ.

4. Suppose that $\bigwedge_{p\in X} \mathsf{controls}(j, p)$. With (3) and EC, we have that $\vdash \bigwedge_{p\in X} \mathsf{controls}(j, p) \rightarrow ([i]\phi \vee [i]\neg\phi)$, and with necessitation and some further reasoning, we conclude that $[j]([i]\phi \vee [i]\neg\phi)$. Since $[i]\phi \vee [i]\neg\phi$ is equivalent in $S5$ to $(\phi \rightarrow [i]\phi) \wedge (\neg\phi \rightarrow [i]\neg\phi)$, we have in particular that $[j](\phi \rightarrow [i]\phi)$. Together with our assumption that $\langle j\rangle\phi$, it follows that $\langle j\rangle[i]\phi$. Using confluence, then, $[i]\langle j\rangle\phi$. □

LEMMA 21. *Let $\longrightarrow_i$ be the accessibility relation in the canonical model, and let alloc_Σ be defined as above. It holds that:*

1. *If $\Sigma \longrightarrow_i \Gamma$ then $\mathsf{alloc}_\Sigma = \mathsf{alloc}_\Gamma$*

2. *If* $\Sigma \longrightarrow_i \Gamma$ *and* $p \notin \mathsf{alloc}_\Sigma(i)$*, then* $p \in \Sigma$ *iff* $p \in \Gamma$

3. *If* $\Sigma \longrightarrow_i \Gamma$ *and* s_j *is a valuation on* $\mathsf{alloc}_\Sigma(j)$*, then* $\langle j\rangle\phi_{s_j} \in \Sigma$ *implies that* $\langle j\rangle\phi_{s_j} \in \Gamma$ *for each* j*.*

proof:

1. Suppose $\Sigma \longrightarrow_i \Gamma$, and that $\mathsf{controls}(j,p) \in \Sigma$. Then, by lemma 20, $[i]\mathsf{controls}(j,p) \in \Sigma$, so $\mathsf{controls}(j,p) \in \Gamma$. Using symmetry, we can conclude that also if $\mathsf{controls}(j,p) \in \Gamma$, then $\mathsf{controls}(j,p)) \in \Sigma$. So $\mathsf{alloc}_\Sigma = \mathsf{alloc}_\Gamma$.

2. From exclusivity EC it follows that if $p \notin \mathsf{alloc}_\Sigma(i)$, then $[i]p \vee [i]\neg p \in \Sigma$. From there it is an easy step to conclude that $p \in \Sigma$ implies $p \in \Gamma$.

3. Clearly, it holds $\mathsf{controls}(j,p)$ for all propositional variables occurring in ϕ_{s_j}. Suppose $\langle j\rangle\phi_{s_j} \in \Sigma$. It follows from lemma 20 that $[i]\langle j\rangle\phi_{s_j} \in \Sigma$, and therefore that $\langle j\rangle\phi_{s_j} \in \Gamma$. □

PROPOSITION 11. *$POS \vdash \phi$ iff $\mathcal{M}, s \models \phi$ in all games of positive control $\mathcal{M}$*

proof: The structure of the completeness proof is similar to that of proposition 9. We construct the canonical model K^{can}, and construct games G_Σ for each maximal consistent set Σ, and show that the model (G_Σ, s_Σ) is bisimilar to (K^{can}, Σ).

First, we need some notation. If v is a valuation over a finite set X, ϕ_v^+ is the conjunction of all propositional variables of X that are true in v, i.e. $\phi_v^+ = \bigwedge\{p \mid v(p) = 1\}$. We will write $[-i]\phi$ for $[\mathsf{Ag} \setminus \{i\}]\phi$.

To construct our game G_Σ, we set $\mathsf{alloc}(i)$ to be the set of all variables for any i. So the strategies $\mathsf{strategy}_\Sigma(i)$ for agent i consist of valuations over *all* variables, and is given by $\mathsf{strategy}_\Sigma(i) = \{v \mid \langle i\rangle[-i]\bigwedge\{p \mid v(p) = 1\} \in \Sigma\}$. So, $\mathsf{strategy}_\Sigma(i)$ contains a strategy for each set of variables that i can *guarantee* to be true.

We now claim that the following relation $\sim$ is a bisimulation:

$$(K^{can}, \Sigma) \sim (G, s) \text{ iff } G = G_\Sigma \text{ and for each } i: \\ s_i(p) = 1 \Leftrightarrow [-i]p \in \Sigma.$$

We need to show that the three bisimulation clauses hold.

I. Suppose $G_\Sigma, s \models p$. Then there must be an agent i with $s_i(p) = 1$. By definition of $\sim$, it holds that $[-i]p \in \Sigma$. By necessitation, $p \in \Sigma$, and by construction of the canonical model that $K^{can}, \Sigma \models p$.

For the other direction, if $p \in \Sigma$, then, with the axiom of positive control, $\bigvee_{i\in N}[-i]p \in \Sigma$. Because Σ is maximal, we have that $[-i]p \in \Sigma$ for some i. But then, by definition of $\sim$, $s_i(p) = 1$, and so, $G_\Sigma, s \models p$.

II. Suppose that $\Sigma \longrightarrow_i \Gamma$ and that $\Sigma \sim G_\Sigma, s$. By standard $S5$-reasoning, if $\langle i\rangle\phi \in \Sigma$, then so is $[i]\langle i\rangle\phi$, which means that $\langle i\rangle\phi \in \Gamma$. By symmetry, the other direction works as well. Since the definition of the game G_Σ depends only on sentences from Σ that are of the form $\langle i\rangle\phi$, so we can conclude that $G_\Sigma = G_\Gamma$.

Consider now the state s' for which it holds for each j that $s'_j(p) = 1$ iff $[-j]p \in \Gamma$. We show that $s \longrightarrow_i s'$. To see that $s_j = s'_j$ for $j \neq i$, note that if $[-j]p \in \Sigma$ then, by $S5$ and asynchronicity, $[i][-j]p \in \Sigma$, and therefore that $[-j]p \in \Gamma$; with symmetry, it follows that $s_j = s'_j$. We also need to show that s'_i is a strategy in G_Σ. This follows from the observation that $[-i]\bigwedge\{p \mid s'_i(p) = 1\} \in \Gamma$, and therefore that $\langle i\rangle[-i]\bigwedge\{p \mid s'_i(p) = 1\} \in \Sigma$.

III. Suppose that $\Sigma \sim (G_\Sigma, s)$, and that $s \longrightarrow_i s'$. We need to find a Γ such that $\Sigma \longrightarrow_i \Gamma$ and $\Gamma \sim (G_\Sigma, s')$. It is enough to show that the set $\{[-j]p \mid s'_j(p) = 1$ and $j \in N\} \cup \{\phi \mid [i]\phi \in \Sigma\}$ is consistent; we can then extend it to a maximal consistent set to obtain our wanted Γ. Because $s_j = s'_j$ for $j \neq i$, it must hold that $\{[-j]p \mid s'_j(p) = 1$ and $j \neq i\} \subseteq \Sigma$, and by $S5$ and asynchronicity, we have that $[i][-j]p \mid s'_j(p) = 1$ and $j \neq i\} \subseteq \Sigma$. So, it is enough to show that $\{[-i]p \mid s'_i(p) = 1\} \cup \{\phi \mid [i]\phi \in \Sigma\}$ is consistent. Suppose it is not. By definition of G_Σ, we have that $\langle i\rangle[-i]\bigwedge\{p \mid s'_i(p) = 1\} \in \Sigma$. But then it would follow that $\langle i\rangle\bot \in \Sigma$, which is a contradiction in $S5$.

□

PROPOSITION 14. *$CON \vdash \phi$ iff $\mathcal{M}, s \models \phi$ in all consensus games $\mathcal{M}$*

proof: Analogous to the completeness proof of the logic of positive control. Define the set of strategies for i in the game G_Σ as $\mathsf{strategy}_\Sigma(i) = \{v \mid \langle i\rangle[-i]\bigwedge\{p \mid v(p) = 0\} \in \Sigma\}$, and then adapt the proof accordingly. □

On the Complexity of Practical ATL Model Checking

Wiebe van der Hoek
Dept of Computer Science
University of Liverpool
Liverpool L69 7ZF, UK
wiebe@csc.liv.ac.uk

Alessio Lomuscio
Dept of Computer Science
University College London
London WC1E 6BT, UK
A.Lomuscio@cs.ucl.ac.uk

Michael Wooldridge
Dept of Computer Science
University of Liverpool
Liverpool L69 7ZF, UK
mjw@csc.liv.ac.uk

ABSTRACT

We investigate the computational complexity of reasoning about multi-agent systems using the cooperation logic ATL of Alur, Henzinger, and Kupferman. It is known that satisfiability checking is EXPTIME-complete for "full" ATL, and PSPACE-complete (in the general case) for the fragment of ATL corresponding to Pauly's Coalition Logic. In contrast, the model checking problems for ATL and Coalition Logic can both be solved in time polynomial in the size of the formula and the size of model against which the formula is to be checked. However, these latter results assume an *extensive* representation of models, in which all states of a model are explicitly enumerated. Such representations are not feasible in practice. In this paper we investigate the complexity of the ATL and Coalition Logic model checking problems for a more "reasonable" model representation known as SRML ("Simple Reactive Modules Language"), a simplified version of the actual model representation languages used for model checkers such as SMV and MOCHA. While it is unsurprising that, when measured against such representations, the model checking problems for ATL and Coalition Logic have a higher complexity than when measured against explicit state representations, we show that in fact the ATL and Coalition Logic model checking problems for SRML models have *the same complexity as the corresponding satisfiability problems*. That is, model checking ATL formulae against SRML models is EXPTIME-complete, and model checking Coalition Logic formulae against SRML models is PSPACE-complete. We conclude by investigating some technical issues around these results, and discussing their implications.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*Logics of programs*; F.2 [**Analysis of algorithms and problem complexity**]

General Terms

Theory

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Keywords

logic, verification, cooperation, complexity

1. INTRODUCTION

Cooperation logics – such as Pauly's Coalition Logic [19, 20, 9] and its temporal extension, the Alternating-time Temporal Logic (ATL) of Alur, Henzinger, and Kupferman [2] – have attracted much interest from the multi-agent systems community [23, 14, 15, 11]. Using such logics, it is possible to reason about the strategic powers of agents and coalitions of agents in game-like multi-agent scenarios.

If one aims to use logics such as ATL and Coalition Logic to reason about systems, then it is obvious to ask how complex the associated reasoning problems are for such logics. Given a particular (multi-agent) system ξ, there are essentially two approaches that may be adopted to reasoning about ξ: theorem proving (via satisfiability checking) and model checking [4]. The satisfiability problem for ATL is EXPTIME-complete, and is hence prohibitively expensive [7, 25]; satisfiability for Coalition Logic is "only" PSPACE-complete [19, 20, 21], and thus apparently easier, but of course this "easiness" is relative: PSPACE-completeness is an extremely negative complexity result!

In contrast, model checking approaches to reasoning about systems appear to be computationally easier. Model checking was originally developed as a technique for verifying that finite state concurrent systems satisfy specifications expressed in the language of temporal logic [8]. The basic idea is that the state transition graph of a system ξ can be understood as a model M_ξ for a temporal logic such as Computation Tree Logic (CTL): if we express desirable or undesirable properties of ξ as formulae φ of CTL, we can then check whether or not ξ has these properties by evaluating whether or not $M_\xi \models \varphi$, and this can be done in deterministic polynomial time [8, p.1044].

Now, one of the attractive features of ATL (and Coalition Logic) is that its model checking problem is apparently no harder than that of its ancestor CTL. More formally, the model checking problem, as discussed in [2] is as follows. We are given an ATS (i.e., an ATL model) S, a state q from S, a formula φ over S and we are asked whether $S, q \models \varphi$. It is shown in [2] that this problem is tractable: it can be solved in time $O(|S| \times |\varphi|)$, where $|S|$ is the size of S and $|\varphi|$ is the size of φ. However, this apparently positive result has a well-known catch. It assumes that the ATS S is *explicitly enumerated* in the input to the problem, and in particular, that all the states of S are listed in the input. Of course, if S has n Boolean variables, then there may be 2^n possibles states. Moreover, in a very recent result, Jamroga and Dix point out that this tractability result only goes through if the number of agents is *fixed*, and *not* considered a parameter of the problem [13]. They show that if

the number of agents is considered an input to the model checking problem, then model checking is NP-complete with respect to a representation of ATL models as *alternating transition systems*, and even worse – Σ_2^p-complete – for the representation of ATL models as *concurrent game structures*.

So, the tractability result for ATL model checking is perhaps misleading, since we have tractability *only* if we explicitly enumerate all states of the system in the input, and *only* if we assume the number of agents is fixed, and not part of the input.

The complexity of ATL model checking is thus particularly sensitive to the *representation* of the model given as input. An obvious question to ask is what kinds of representations are used *in practice*, by actual model checking systems. In fact, practical model checkers such as SMV (for CTL) [17], SPIN (for LTL) [12], and MOCHA (for ATL) [3] use a succinct *model specification language* for defining models. While the model specification languages used by these systems differ on details, they all have the flavour of programming languages, providing the verifier with a relatively high-level way of defining models. But this then raises the question of the extent to which the apparently positive complexity results for CTL, ATL, and Coalition Logic model checking hold when using such model representation languages. It comes as no surprise that model checking against such model representations will be harder. For example, a PSPACE-completeness result for CTL model checking against a representation of concurrent programs was obtained by Kupferman et al [16], and extended to combinations of CTL with knowledge in [22].

However, to date, the issue of complexity of ATL model checking for realistic model specification languages has not been investigated, and this is the aim of the present paper. The particular model specification language we focus on is called SRML ("Simple Reactive Modules Language"), and is a slightly simplified version of the REACTIVE MODULES LANGUAGE (RML) [1]. We are particularly interested in RML for several reasons: first, RML is the language that is actually used by the ATL model checking system MOCHA [3]; it has been shown to be well suited to building models of a range of multi-agent protocols; and RML programs can easily be used to code basic reactive agents. As we pointed out above, it is unsurprising that, assuming representations such as SRML, the model checking problems for ATL and Coalition Logic have a higher complexity than when measured against explicit state representations, particularly given the results of Jamroga and Dix [13]. However, we show that in fact the ATL and Coalition Logic model checking problems for SRML programs have *the same complexity as the corresponding satisfiability problems*! That is, model checking ATL formulae against SRML programs is EXPTIME-complete, and model checking Coalition Logic formulae against SRML programs is PSPACE-complete. There is thus a *huge* gulf – much larger than perhaps might be expected – between the provably tractable model checking problem for explicit state models and the provably *in*tractable model checking problem for "realistic" model representations. Indeed, it again raises the question of the relative merits of theorem proving *versus* model checking [10]. We begin by describing ATL and then SRML, the simplified variant of RML that we use throughout the paper. We present our main results in section 4. We conclude by investigating some technical issues around these results, and discussing their implications.

2. ATL

Alternating-time Temporal Logic (ATL) [2] can be understood as a generalisation of the well-known branching time temporal logic CTL [8], in which path quantifiers are replaced by *cooperation modalities*. A cooperation modality $\langle\langle C\rangle\rangle\varphi$, where C is a group of agents, expresses the fact that there exists a strategy profile for C such that by following this strategy profile, C can ensure φ. Thus, for example, the system requirement "agents 1 and 2 can cooperate to ensure that the system never enters a fail state" may be captured by the ATL formula $\langle\langle 1,2\rangle\rangle\,\Box\neg fail$. The "$\Box$" temporal operator means "now and forever more": additional temporal connectives in ATL are "$\Diamond$" ("either now or at some point in the future"), "$\mathcal{U}$" ("until"), and "$\bigcirc$" ("in the next state").

To give a precise definition of ATL, we must first introduce the semantic structures over which formulae of ATL are interpreted. An *alternating transition system* (ATS) is a 6-tuple

$$S = \langle \Pi, \Sigma, Q, Q_0, \pi, \delta \rangle, \text{ where:}$$

- Π is a finite, non-empty set of *Boolean variables*;
- $\Sigma = \{a_1, \ldots, a_n\}$ is a finite, non-empty set of *agents*;
- Q is a finite, non-empty set of *states*, with $Q_0 \subseteq Q$ representing the *initial* states of Q;
- $\pi : Q \to 2^{\Pi}$ gives the set of Boolean variables satisfied in each state;
- $\delta : Q \times \Sigma \to 2^{2^Q}$ is the system transition function, which maps states and agents to the choices available to these agents. Thus $\delta(q, a)$ is the set of choices available to agent a when the system is in state q. We require that this function satisfy the requirement that for every state $q \in Q$ and every set $Q_1, \ldots, Q_n$ of choices $Q_i \in \delta(q, a_i)$, the intersection $Q_1 \cap \cdots \cap Q_n$ is a singleton.

(Initial states Q_0 were not considered part of ATSs in the original publications on ATL: we introduce them to be consistent with the behaviour of model checkers for ATL, which check formulae against the initial state of the system: see below.)

An ATL formula, formed with respect to an alternating transition system $S = \langle \Pi, \Sigma, Q, Q_0, \pi, \delta \rangle$, is then defined by the following grammar:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle C\rangle\rangle \bigcirc \varphi \mid \langle\langle C\rangle\rangle \Box \varphi \mid \langle\langle C\rangle\rangle \varphi \,\mathcal{U}\, \varphi$$

where $p \in \Pi$ is a Boolean variable, and $C \subseteq \Sigma$ is a set of agents. We assume the remaining connectives of classical logic ("$\bot$", "$\rightarrow$", "$\leftarrow$", "$\leftrightarrow$", "$\wedge$") are defined as abbreviations in the usual way, and define $\langle\langle C\rangle\rangle \Diamond \varphi$ as $\langle\langle C\rangle\rangle \top \,\mathcal{U}\, \varphi$.

To give the semantics of ATL, we need some further definitions. For two states $q, q' \in Q$ and an agent $a \in \Sigma$, we say that state q' is an *a-successor* of q if there exists a set $Q' \in \delta(q, a)$ such that $q' \in Q'$. Intuitively, if q' is an a-successor of q, then q' is a possible outcome of one of the choices available to a when the system is in state q. We denote by $succ(q, a)$ the set of a-successors to state q, and say that q' is simply a *successor* of q if for all agents $a \in \Sigma$, we have $q' \in succ(q, a)$; intuitively, if q' is a successor to q, then when the system is in state q, the agents Σ can cooperate to ensure that q' is the next state the system enters.

A *computation* of an ATS $\langle \Pi, \Sigma, Q, Q_0, \pi, \delta \rangle$ is an infinite sequence of states $\lambda = q_0, q_1, \ldots$ such that for all $u > 0$, the state q_u is a successor of q_{u-1}. A computation $\lambda \in Q^\omega$ starting in state q is referred to as a *q-computation*; if $u \in \mathbb{N}$, then we denote by $\lambda[u]$ the u'th state in λ; similarly, we denote by $\lambda[0, u]$ and $\lambda[u, \infty]$ the finite prefix $q_0, \ldots, q_u$ and the infinite suffix $q_u, q_{u+1}, \ldots$ of λ respectively.

Intuitively, a *strategy* is an abstract model of an agent's decision-making process; a strategy may be thought of as a kind of plan for an agent. Formally, a strategy f_a for an agent $a \in \Sigma$ is a total

function $f_a : Q^+ \rightarrow 2^Q$, which must satisfy the constraint that $f_a(\lambda \cdot q) \in \delta(q, a)$ for all $\lambda \in Q^*$ and $q \in Q$. Given a set $C \subseteq \Sigma$ of agents, and an indexed set of strategies $F_C = \{f_a \mid a \in C\}$, one for each agent $a \in C$, we define $out(q, F_C)$ to be the set of possible outcomes that may occur if every agent $a \in C$ follows the corresponding strategy f_a, starting when the system is in state $q \in Q$. That is, the set $out(q, F_C)$ will contain all possible q-computations that the agents C can "enforce" by cooperating and following the strategies in F_C. Note that the "grand coalition" of all agents in the system can cooperate to uniquely determine the future state of the system, and so $out(q, F_\Sigma)$ is a singleton. Similarly, the set $out(q, F_\emptyset)$ is the set of all possible q-computations of the system.

We can now give the rules defining the satisfaction relation "$\models$" for ATL, which holds between pairs of the form S, q (where S is an ATS and q is a state in S), and formulae of ATL:

$S, q \models \top$;

$S, q \models p$ iff $p \in \pi(q)$ (where $p \in \Pi$);

$S, q \models \neg\varphi$ iff $S, q \not\models \varphi$;

$S, q \models \varphi \vee \psi$ iff $S, q \models \varphi$ or $S, q \models \psi$;

$S, q \models \langle\langle C \rangle\rangle \bigcirc \varphi$ iff there exists a set of strategies F_C, such that for all $\lambda \in out(q, F_C)$, we have $S, \lambda[1] \models \varphi$;

$S, q \models \langle\langle C \rangle\rangle \Box \varphi$ iff there exists a set of strategies F_C, such that for all $\lambda \in out(q, F_C)$, we have $S, \lambda[u] \models \varphi$ for all $u \in \mathbb{N}$;

$S, q \models \langle\langle C \rangle\rangle \varphi \mathcal{U} \psi$ iff there exists a set of strategies F_C, such that for all $\lambda \in out(q, F_C)$, there exists some $u \in \mathbb{N}$ such that $S, \lambda[u] \models \psi$, and for all $0 \leq v < u$, we have $S, \lambda[v] \models \varphi$.

Satisfaction for the ATL formula $\langle\langle C \rangle\rangle \Diamond \varphi$ is defined via equivalence to $\langle\langle C \rangle\rangle \top \mathcal{U} \varphi$.

We identify two important fragments of ATL which we will subsequently refer to:

- *Propositional logic* is the fragment of ATL in which no cooperation modalities are permitted. The truth of a propositional logic formula is determined on an ATS by using the first four items of the definition for satisfaction above.
- *Coalition Logic* is the fragment of ATL in which the only cooperation modalities allowed are of the form $\langle\langle C \rangle\rangle \bigcirc$ [19, 20, 9]. The truth of a Colition Logic formula is determined on an ATS by using the first five items of the definition for satisfaction above.

The (explicit state) *model checking* problem[1] is the problem of determining, for a given formula φ and ATS $S = \langle \Pi, \Sigma, Q, Q_0, \pi, \delta \rangle$, whether $S, q \models \varphi$ for all $q \in Q_0$. The model checking problems for ATL (and hence Coalition Logic) can be solved by a deterministic algorithm in time polynomial in the size of the inputs ATS S and the formula φ [2, pp.690–691] – assuming that ATS S in question is explicitly enumerated in the input in all its components, and the number of agents is considered fixed [13].

For the *satisfiability* problem, we are simply given a formula φ and asked whether there is an ATS S and state q in S such that $S, q \models \varphi$. The satisfiability problem for ATL is EXPTIME-complete [7, 25], while for Coalition Logic it is PSPACE-complete in the general case [19, p.63].

3. SRML

We have already observed that the positive tractability results for ATL and Coalition Logic model checking rely on the assumption that the model against which a formula is to be checked is explicitly enumerated in the input to the model checking algorithm, and that this assumption is both infeasible (there will be exponentially many states in the number of Boolean variables), and never used in practice. Instead, practical model checkers such as SMV [17], SPIN [12], and MOCHA [3] use model specification languages which permit the *succinct, high-level* specification of models. In this section, we describe the language used in this paper for this purpose: the Simple Reactive Modules Language (SRML). As its name suggests, this language is a strict subset of the RML language used in the MOCHA model checker[2]. Note that our choice of language is not arbitrary: we comment on it below. We begin with an informal overview.

3.1 Overview

Agents in RML are known as *modules*, and for consistency with the pre-existing literature on RML, we will stay with this terminology in SRML. An SRML module consists of:

- an *interface*, which defines the name of the module and lists the Boolean variables under the *control* of the module; and
- a number of *guarded commands*, which define the choices available to the module at every state.

The states of an SRML-defined model correspond to the possible valuations that may be given to the Boolean variables under the control of modules in the system: for the purposes of SRML, two states that agree on the valuation of all variables are the same[3].

The guarded commands of a module are divided into two sets: those used for *initialising* the variables under the module's control (`init` guarded commands), and those used for *updating* these variables subsequently (`update` guarded commands). A guarded command has two parts: a "condition" part (the "guard"), and a corresponding "action" part, which defines how to update the value of (some of) the variables under the control of a module. The intuitive reading of a guarded command $\varphi \rightsquigarrow e$ is "if the condition φ is satisfied, then *one* of the choices available to the module is to execute the assignment expression e". The assignment expression e is essentially a sequence of assignment statements, just as in conventional imperative programming languages: these assignment statements define how (some subset of) the module's controlled variables should be updated if the guarded command is executed. Notice that the truth of the guard φ does not mean that e *will* be executed: only that it is *enabled* for execution – it *may* be executed. If no guarded commands of a given module are enabled in some state, then the values of the variables in that module are assumed to remain unchanged in the next state: the module has no choice.

[1] Model checking is sometimes formulated as the problem of finding the set of states in a structure that satisfy a given formula; our formulation is chosen to be consistent with the way in which actual model checkers work, and it should be stressed that from the asymptotic analysis point of view, there is no difference in complexity between the formulations.

[2] For those familiar with RML and MOCHA, SRML assumes finite, propositional modules, with each module containing a single atom, in which no variable is awaited by any module.

[3] This is of course not the case in general in modal/temporal logic.

The choices available to a module wrt initialisation of its variables are thus defined by the `init` guarded commands of the module, while the choices available to a module wrt updating its variables are defined by the `update` guarded commands. To make this more concrete, here is an example guarded command:

$$\underbrace{(x \wedge y)}_{\text{guard}} \rightsquigarrow \underbrace{x' := \bot; y' := \top}_{\text{action}}$$

The guard is the propositional logic formula $x \wedge y$, so this guarded command will be enabled in any system state where both x and y take the value "$\top$". If the guarded command is chosen, then the effect is that in the next state of the system, the variable x will take value $\bot$, while y will take value $\top$. (The "prime" notation v' means "the value of variable v *after* the statement is executed".) We will write `skip` as an abbreviation for the empty assignment expression, which leaves the value of every variable controlled by a module unchanged.

Here is an example of a module, illustrating the concrete syntax that we will use for modules (which is essentially that of RML [1]).

```
module toggle controls x
  init
  [] ⊤ ⇝ x' := ⊤
  [] ⊤ ⇝ x' := ⊥
  update
  [] x ⇝ x' := ⊥
  [] (¬x) ⇝ x' := ⊤
```

This module, named *toggle*, controls a single Boolean variable, x. There are two `init` guarded commands and two `update` guarded commands. (The symbol "`[]`" is a syntactic separator.) The `init` guarded commands of *toggle* define two choices for the initialisation of this variable: assign it the value $\top$ or the value $\bot$. With respect to `update` guarded commands, the first command says that if x has the value $\top$, then the corresponding choice is to assign it the value $\bot$, while the second command says that if x has the value $\bot$, then it can subsequently be assigned the value $\top$. In other words, the module non-deterministically chooses a value for x initially, and then on subsequent rounds toggles this value. Notice that in this example, the `init` commands of this module are non-deterministic, while the `update` commands are deterministic.

3.2 Formal Definition

In this section, we will use Π to denote the set of Boolean variables in a system, as in ATL. Formally, a guarded command γ over Π is an expression

$$\gamma : \qquad \varphi \rightsquigarrow v_1' := \psi_1; \ldots; v_k' := \psi_k$$

where φ (the guard) is a propositional logic formula over Π, each v_i is a member of Π and ψ_i is a propositional logic formula over Π. We require that no variable v_i appears on the l.h.s. of two assignment statements in the same guarded command (hence no issue on the ordering of the updates arises). The intended interpretation is that if the formula φ evaluates to true against the interpretation corresponding to the current state of the system, then the statement is *enabled* for execution; executing the statement means evaluating each ψ_i against the current state of the system, and setting the corresponding variable v_i to the truth value obtained from evaluating ψ_i. We say that $v_1, \ldots, v_k$ are the *controlled variables* of γ, and denote this set by $ctr(\gamma)$. A set of guarded commands is said to be *disjoint* if their controlled variables are mutually disjoint.

Given a propositional valuation $\theta \subseteq \Pi$ and a guarded command $\gamma : \varphi \rightsquigarrow v_1' := \psi_1; \ldots; v_k' := \psi_k$ such that θ enables γ (i.e., $\theta \models \varphi$) we denote the result of *executing* γ on θ by $\theta \oplus \gamma$. For example, if $\theta = \{p, r\}$, and $\gamma = p \rightsquigarrow q' := \top; r' := p \wedge \neg r$, then $\theta \oplus \gamma = \{p, q\}$.

Given a propositional valuation $\theta \subseteq \Pi$, and set Γ of disjoint guarded commands over Π such that every member of Γ is enabled by θ, then the interpretation θ' resulting from Γ on θ is denoted by $\theta' = \theta \oplus \Gamma$: since the members of Γ are disjoint, we can pick them in any order to execute on θ.

As described above, there are two classes of guarded commands that may be declared in an atom: `init` and `update`. An `init` guarded command is only used to initialise the values of variables, when the system begins execution. Full RML allows for quite sophisticated initialisation schemes, but in SRML, we will assume that the guards to `init` command are "$\top$", i.e., every `init` command is enabled for execution in the initialisation round of the system.

An SRML *module*, m, is a triple:

$$m = \langle ctr, init, update \rangle \text{ where:}$$

- $ctr \subseteq \Pi$ is the (finite) set of variables controlled by m;
- $init$ is a (finite) set of *initialisation* guarded commands, such that for all $\gamma \in init$, we have $ctr(\gamma) \subseteq ctr$; and
- $update$ is a (finite) set of *update* guarded commands, such that for all $\gamma \in init$, we have $ctr(\gamma) \subseteq ctr$.

Given a module m, we denote the controlled variables of m by $ctr(m)$, the initialisation guarded commands of m by $init(m)$, and the update guarded commands of m by $update(m)$. An SRML system is then an $(n+2)$-tuple

$$\langle \Sigma, \Pi, m_1, \ldots, m_n \rangle$$

where $\Sigma = \{1, \ldots, n\}$ is a set of agents and Π is a vocabulary of Boolean variables, (as in ATSs), and for each $i \in \Sigma$, m_i is the corresponding module defining i's choices; we require that $\{ctr(m_1), \ldots, ctr(m_n)\}$ forms a partition of Π (i.e., every variable in Π is controlled by some agent, and no variable is controlled by more than one agent).

A *joint* guarded command (j.g.c.) for a coalition $C \subseteq \Sigma$ is an indexed tuple $\langle \gamma_1, \ldots, \gamma_k \rangle$, with a guarded command $\gamma_i \in m_i$ for each $i \in C$. A j.g.c. $\langle \gamma_1, \ldots, \gamma_k \rangle$ is enabled by a propositional valuation $\theta \subseteq \Pi$ iff all its members are enabled by θ.

Given an SRML system R, the corresponding ATS, which we denote by S_R, is defined in the obvious way:

- the states Q of S_R correspond to the possible valuations to variables Π, with Q_0 being the subset of Q that may be generated by the $init$ guarded commands of the modules in R; and
- the δ function of S_R is defined in each state for each possible coalition by the sets of enabled guarded commands (choices) for that coalition.

We write $R \models \varphi$ to indicate that $S_R, q_0 \models \varphi$ for every initial state q_0 in S_R: the SRML model checking problem is then the problem of checking, for given R and φ, whether or not $R \models \varphi$.

4. MAIN RESULTS

We now present our main results. Given the concerns of this paper – understanding the complexity of ATL model checking for a particular model representation – the first of these results may seem somewhat strange, but its role will become clear shortly. We show that, given an SRML system R, we can construct a formula

$mimic(R)$ of ATL such that $mimic(R)$ is *only* satisfied in models that are "equivalent" to S_R. We can thus understand $mimic(R)$ as representing the ATL *theory* of R. An important property of the construction of $mimic(R)$ is that this formula is of size polynomial in the size of R. For such a $mimic(R)$, we must characterize the behaviour of R in the sense that:

(a) abilities in R are transferred to models satisfying $mimic(R)$;

(b) agents should not be able to achieve more in any $mimic(R)$-model than the modules can do in R.

The construction of $mimic(R)$ is as follows. Let the variables $ctr(i)$ under control of module i be $v^i_1, \ldots, v^i_j$, and let the module i be $\langle ctr(i), init, update\rangle$. We are now going to define requirements $t(\mathtt{init}, i)$, (2), (3) and (4); they are supposed to take care of our condition (a). Regarding $init$, we do the following. Suppose there are c_i *init* guarded commands for agent i, and let a typical $init$ guarded command ι^i_a ($a \leq c_i$) for agent i be composed of a positive part $\iota^{+_i}_{a_u}$ and a negative part $\iota^{-_i}_{a_v}$ ($u \leq nu, v \leq nv$) as follows:

$$\iota^{+_i}_{a_u} \quad \top \rightsquigarrow v^{+_i}_{a_1} := \top, \ldots, \top \rightsquigarrow v^{+^i}_{a_{nu}} := \top,$$

$$\iota^{-_i}_{a_v} \quad \top \rightsquigarrow v^{-_i}_{a_1} := \bot, \ldots, \top \rightsquigarrow v^{-_i}_{a_{nv}} := \bot$$

Where no $v^{+_i}_{a_j}$ can be the same as any $v^{+_i}_{a_k}$, for any $j \leq nu$ and $k \leq nv$ (this is guaranteed by the fact that $\iota^{+_i}_a$ and $\iota^{-_i}_a$ represent one guarded command). Then denote

$$t(\mathtt{init}, i) = \bigvee_{a \leq c_i} \Big(\bigwedge_{j \leq nu} v^{+_i}_{a_j} \wedge \bigwedge_{k \leq nv} \neg v^{-_i}_{a_k} \Big) \tag{1}$$

This formula $t(\mathtt{init}, i)$ expresses the effect of executing one of the c_i guarded commands ι_a for agent i. Note that we do not have an *exclusive* disjunction here, since it might well be that the *effect* of executing one initialisation command ι_a may coincide with that of another. Given an SRML-system R, it is obvious that in its associated ATS S_R, a state q verifies $t(\mathtt{init}, i)$ iff q is one of the initial states in Q_0.

Now we turn to the *update* part of the module. Suppose it looks like the following (note that all v'^i_j are occurrences of i's variables, not the variables themselves).

$$\begin{array}{lllll}
\gamma^i_1: & \varphi_1 \rightsquigarrow & {v^i_{1_1}}' := \psi_{1_1} & \ldots & {v^i_{1_{n1}}}' := \psi_{1_{n1}} \\
\gamma^i_2: & \varphi_2 \rightsquigarrow & {v^i_{2_1}}' := \psi_{2_1} & \ldots & {v^i_{2_{n2}}}' := \psi_{2_{n2}} \\
\ldots & \ldots & \ldots & \ldots & \ldots \\
\gamma^i_m: & \varphi_m \rightsquigarrow & {v^i_{m_1}}' := \psi_{m_1} & \ldots & {v^i_{m_{nm}}}' := \psi_{m_{nm}}
\end{array}$$

Given an SMRL R, introduce for every ψ_{k_t} ($k \leq m, t \leq nk$) a new propositional variable x_{k_t}. The idea is that every x will store the old value of its corresponding ψ, so we stipulate:

$$\bigwedge_{k \leq m, t \leq nk} \langle\langle\rangle\rangle \Box (\psi_{k_t} \leftrightarrow \langle\langle\rangle\rangle \bigcirc x_{k_t}) \tag{2}$$

Some of i's variables will not be changed by executing a command, so we also need to keep track of the variables old values. For that, we introduce, for every $v^i_e \in ctr(i)$, variables y^i_e ($e \leq j$), and stipulate

$$\bigwedge_{e \leq j} \langle\langle\rangle\rangle \Box (v^i_e \leftrightarrow \langle\langle\rangle\rangle \bigcirc y^i_e) \tag{3}$$

We now look at our constraint (b), which should take care of the fact that the if σ is ATL-satisfiable, it should be in a context that is behaving well enough in order to be transferred to an SMRL R. To guarantee that an agent i cannot spontaneously bring about anything that is not triggered by the execution of a command γ^i_z, we introduce propositional variables $done(i,1), \ldots, done(i,m)$ and a special atom $none(i)$. The latter atom denotes that i did not execute any of its guarded commands. Now consider a command γ^i_z. As before, let $ctr(\gamma^i_z)$ be i's variables occurring in the command γ^i_z. Now represent γ^i_z by

$$\varphi_z \rightarrow \langle\langle i\rangle\rangle \bigcirc (new(i) \wedge rec(i,z)) \tag{4}$$

where

$$new(i) = \bigwedge_{v \in ctr(\gamma^i_z)} (v \leftrightarrow x) \wedge \bigwedge_{u \in ctr(i) \setminus ctr} (u \leftrightarrow y)$$

and

$$rec(i,z) = done(i,z) \wedge \bigwedge_{s \leq m, s \neq z} \neg done(i,s) \wedge \neg none(i)$$

$new(i)$ says that the values of the relevant ψ_{k_t} are, via x_{k_t}, copied in the corresponding variable $v^i_{k_t}$, and the other variables u of i receive their 'old' values through a copy of the corresponding variables y. This takes care of our goal specified as (a). For the (b)-part, $rec(i,z)$ takes a record of which guarded command has been applied. This ensures that agent i indeed 'only' executes guarded commands, if we add the global properties (5)–(8):

$$\langle\langle\rangle\rangle \Box ((\bigwedge_{k \leq m} \neg \varphi_k) \leftrightarrow \langle\langle\rangle\rangle \bigcirc none(i)) \tag{5}$$

$$\langle\langle\rangle\rangle \Box \bigwedge_{k \leq m} (\langle\langle\rangle\rangle \bigcirc done(i,k) \rightarrow \varphi_k) \tag{6}$$

$$\langle\langle\rangle\rangle \Box \bigwedge_{k \leq m} (done(i,k) \rightarrow (new(i) \wedge \neg none(i))) \tag{7}$$

$$\begin{array}{ll} \langle\langle\rangle\rangle \Box (& (\neg none(i) \rightarrow \bigtriangledown_{k \leq m} done(i,k)) \\ & \wedge (none(i) \rightarrow \bigwedge_{v \in ctr(i)} v \leftrightarrow y) \ ) \end{array} \tag{8}$$

Equation (5) expresses that agent i will only do nothing if all his guards are false; (6) says that i can only execute the k-th command if the k-th guard is true; (7) expresses that if a flag is set indicating that i executes the k-th command, then, indeed, this command is executed, and the $none(i)$-flag is not set to true, and, finally, (8) expresses that either i does nothing, or he executes exactly one of his guarded commands and that doing nothing implies keeping all the "old" values for i's variables.

We are now in a position to define $mimic(R)$:

$$\bigwedge_{i \leq n} (t(\mathtt{init}, i) \wedge (2) \wedge (3) \wedge (4) \wedge (5) \wedge (6) \wedge (7) \wedge (8))$$

Note that the size of $mimic(R)$ is polynomial in R, since we only add a given number of new variables, where the number depends on the number of local variables and the number of formulas ψ appearing at the right of a ":=" in any command, and each of the conjuncts in $mimic(R)$ is polynomial in R.

Now, suppose we have some S, q such that $S, q \models mimic(R)$. The states of S will not look exactly like the states of S_R, since

they will include propositional variables that will not be in S_R (i.e., the new variables we introduced in the translation). But nevertheless, the properties of agents – their abilities, with respect to the propositional variables of R – must be the same in the ATS S that satisfies $mimic(R)$ as in R. In short: *the model S must preserve the truth of* ATL *formulae over R.*

Returning to the main concerns of the paper, we now see the relevance of this construction.

THEOREM 1. *The* ATL *model checking problem for* SRML *is polynomial-time reducible to* ATL *unsatisfiability.*

PROOF. Let R be the given SRML system, and φ be the formula we want to check. Then the following are equivalent:

1. $R \models \varphi$
2. $mimic(R) \rightarrow \varphi$ is valid in ATL.

□

The rationale for stating this result will become clear in the following:

THEOREM 2. *The* ATL *model checking problem for* SRML *models is* EXPTIME*-complete.*

PROOF. Membership of EXPTIME follows from the fact, proved in Theorem 1, that the ATL model checking problem for SRML is polynomial time reducible to ATL unsatisfiability. Since the ATL satisfiability problem is EXPTIME complete [7, 25], then ATL unsatisfiability is in co-EXPTIME; but since EXPTIME is a deterministic complexity class, EXPTIME = co-EXPTIME, and we conclude that the SRML model checking for SRML is in EXPTIME.

We prove EXPTIME-hardness by reduction from the problem of determining whether a given player has a winning strategy in the two-player game PEEK-G_4 [24, p.158]. An instance of PEEK-G_4 is a quad:

$$\langle X_1, X_2, X_3, \varphi \rangle \text{ where:}$$

- X_1 and X_2 are disjoint, finite sets of Boolean variables, with the intended interpretation that the variables in X_1 are under the control of agent 1, and X_2 are under the control of agent 2;
- $X_3 \subseteq (X_1 \cup X_2)$ are the variables deemed to be true in the initial state of the game; and
- φ is a propositional logic formula over the variables $X_1 \cup X_2$, representing the winning condition.

The game is played in a series of rounds, with the agents $i \in \{1, 2\}$ alternating (with agent 1 moving first) to select a value (true or false) for one of their variables in X_i, with the game starting from the initial assignment of truth values defined by X_3. Variables that were not changed retain the same truth value in the subsequent round. An agent wins in a given round if it makes a move such that the resulting truth assignment defined by that round makes the winning formula φ true. The decision problem associated with PEEK-G_4 involves determining whether agent 2 has a winning strategy in a given game instance $\langle X_1, X_2, X_3, \varphi \rangle$. Notice that PEEK-$G_4$ only requires "memoryless" (Markovian) strategies: whether or not an agent i can win depends only on the current truth assignment, the distribution of variables, the winning formula, and whose turn it is currently. As a corollary, if agent i can force a win, then it can force a win in $O(2^{|X_1 \cup X_2|})$ moves.

Given an instance $\langle X_1, X_2, X_3, \varphi \rangle$ of PEEK-G_4, we produce an instance of SRML model checking as follows. For each Boolean variable $x \in (X_1 \cup X_2)$, we create a variable with the same name in our SRML model, and we create an additional Boolean variable $turn$, with the intended interpretation that if $turn = \top$, then it is agent 1's turn to move, while if $turn = \bot$, then it is agent 2's turn to move. We have a module $move$, the purpose of which is to control $turn$, toggling its value in each successive round, starting from the initial case of it being agent 1's move.

```
module move controls turn
  init
  [] ⊤ ⇝ turn' := ⊤
  update
  [] turn ⇝ turn' := ⊥
  [] (¬turn) ⇝ turn' := ⊤
```

For each of the two PEEK-G_4 players $i \in \{1, 2\}$, we create an SRML module ag_i that `controls` the variables X_i. The module ag_i is as follows. It begins by deterministically initialising the values of all its variables to the values defined by X_3 (that is, if variable $x \in X_i$ appears in X_3 then this variable is initialised to $\top$, otherwise it is initialised to $\bot$). Subsequently, when it is this player's turn, it can non-deterministically choose at most one of the variables under its control and toggle the value of this variable; when it is not this player's turn, it has no choice but to do nothing, leaving the value of all its variables unchanged. The general structure of ag_1 is thus as follows, where $X_1 = \{x_1, \ldots, x_k\}$.

```
module ag1 controls x1, ..., xk
  init
  // initialise to values from X3
  [] ⊤ ⇝ x1' := ...; xk := ...
  update
  [] turn ⇝ x1' := ⊥
  [] turn ⇝ x1' := ⊤
  ...
  [] turn ⇝ xk' := ⊥
  [] turn ⇝ xk' := ⊤
  [] ⊤ ⇝ skip
```

Notice that an agent can always `skip`, electing to leave its variables unchanged; and, if it is not this agent's turn to move, this is the *only* choice it has.

The SRML system under consideration contains just these three modules. It can be shown easily that player 2 has a strategy for φ in the PEEK-G_4 game $\langle X_1, X_2, X_3, \varphi \rangle$ iff the SRML system satisfies the formula $\langle\langle 2 \rangle\rangle (\neg\varphi) \mathcal{U} (\varphi \wedge turn)$. □

There are several small points of interest about this proof. First, notice that the reduction used for EXPTIME-hardness requires *only a fixed number of agents* (three). Thus we have the following, stronger result: ATL model checking for SRML models is EXPTIME complete for any fixed number n of modules, where $n \geq 3$. Second, notice the form of the guarded commands used in the reduction: on the rhs of any guard, we only change the value of at most one variable. Thus, again, we have a stronger result: EXPTIME hardness even when guarded commands are assumed to be of this particularly impoverished form.

Next, we turn our attention to significant subsets of ATL: first, Coalition Logic, the fragment in which the only temporal modality allowed is "$\bigcirc$" [19, 21, 9].

THEOREM 3. *The Coalition Logic model checking problem for* SRML *models is* PSPACE*-complete.*

PROOF. We first prove PSPACE-hardness, by reducing the problem of determining the truth of Quantified Boolean Formulae (QBF) [18, pp.456–458] to that of Coalition Logic model checking against SRML models.

An instance of QBF is given by a formula

$$\exists x_1 \forall x_2 \cdots Q_k \varphi(x_1, x_2, \ldots, x_k)$$

where $x_1, \ldots, x_k$ are Boolean variables, the quantifier Q_k is $\exists$ if k is odd, and $\forall$ if k is even, and $\varphi(x_1, \ldots, x_k)$ is a propositional logic formula over the variables $x_1, x_2, \ldots, x_k$. Such a formula is true if there exists a valuation for x_1 such that for all valuations for x_2, ..., such that the formula $\varphi(x_1, x_2, \ldots, x_k)$ is true. We proceed to create an SRML system containing three modules, $move$, $ag_\exists$, $ag_\forall$, the variables $x_1, x_2, \ldots, x_k$, and, in addition, k variables $m_1, m_2, \ldots, m_k$, such that m_i will be true if we are about to assign a value for variable x_i.

The module $turn$ simply passes the value $\top$ along each of the variables $m_1, m_2, \ldots, m_k$ in turn.

```
module turn controls m_1, ..., m_k
  init
  [] ⊤ ⇝ m_1' := ⊤, m_2' := ⊥; ...; m_k := ⊥
  update
  [] m_1 ⇝ m_1' := ⊥; m_2' := ⊤
  ...
  [] m_i ⇝ m_i' := ⊥; m_{i+1}' := ⊤
  ...
  [] m_{k-1} ⇝ m_{k-1}' := ⊥; m_k' := ⊤
```

Notice that at most one of the variables $m_1, m_2, \ldots, m_k$ will be true at any given time.

We define the modules $ag_\exists$ as follows. This module `controls` all odd numbered variables. On even rounds, the module simply `skip`s, leaving the values of its variables unchanged. On odd numbered rounds i, $1 \leq i \leq k$, the module will have two choices: make variable x_i true or make x_i false.

```
module ag_∃ controls ...// odd numbered variables
  init
  [] ⊤ ⇝ skip
update
  [] m_1 ⇝ x_1' := ⊥
  [] m_1 ⇝ x_1' := ⊤
  [] m_2 ⇝ skip
  [] m_3 ⇝ x_3' := ⊥
  [] m_3 ⇝ x_3' := ⊤
  [] m_4 ⇝ skip
  ...
```

We define $ag_\forall$ similarly, swapping "even" for "odd". Noting that (i) the module $turn$ plays no part in determining the value of variables other than the move variables $m_1, \ldots, m_k$; (ii) when it is player i's turn in round j, then the only choices i has are to assign truth or falsity to variable x_j; (iii) when it is not player is turn in round j, then j must leave all its variables unchanged; (iv) that the construction is clearly polynomial in the size of the input formula.

The QBF instance $\exists x_1 \forall x_2 \cdots Q_k \varphi(x_1, x_2, \ldots, x_k)$ is true iff the the SRML system above satisfies the formula

$$\langle\langle ag_\exists \rangle\rangle \bigcirc \neg \langle\langle ag_\forall \rangle\rangle \bigcirc \neg \cdots \varphi(x_1, x_2, \ldots, x_k).$$

To see that the model checking problem is decidable in PSPACE, we present an algorithm for deciding the problem that works in polynomial space: see Figure 1. The algorithm takes as input an

```
1.  function eval(φ, R) returns ⊤ or ⊥
2.    θ := ∅
3.    for each θ-enabled initial j.g.c. ⟨γ_1, ..., γ_n⟩ for Σ do
4.      if not aux(φ, θ ⊕ {γ_1, ..., γ_n}, R) then
5.        return ⊥
6.      end-if
7.    end-for
8.    return ⊤
9.  end-function eval

10. function aux(φ, θ, R) returns ⊤ or ⊥
11.   if φ ∈ Π then
12.     if φ ∈ θ then return ⊤ else return ⊥ end-if
13.   elsif φ = ¬ψ then
14.     return not aux(ψ, θ, R)
15.   elsif φ = ψ_1 ∨ ψ_2 then
16.     return aux(ψ_1, θ, R)
17.       or aux(ψ_2, θ, R)
18.   elsif φ = ⟨⟨C⟩⟩◯ψ then
19.     for each θ-enabled update j.g.c. ⟨γ_1, ..., γ_l⟩ for C do
20.       flag := ⊤
21.       for each θ-enabled update j.g.c. ⟨γ_{l+1}, ..., γ_n⟩ for Σ \ C do
22.         if not aux(ψ, θ ⊕ {γ_1, ..., γ_l, γ_{l+1}, ..., γ_n}, R) then
23.           flag := ⊥
24.         end-if
25.       end-for
26.       if flag then return ⊤ end-if
27.     end-for
28.     return ⊥
29.   end-if
30. end-function aux
```

Figure 1: A polynomial space algorithm for checking Coalition Logic formulae against SRML models.

SRML system R, and a formula φ of coalition logic to check against this system. The algorithm is in two parts: the first part (the function $eval(\cdots)$) generates each initial state of the system in turn from the initial guarded commands of the modules comprising R, and then invokes the auxiliary function $aux(\cdots)$ to check whether the input φ is true in these initial states. The $aux(\cdots)$ function is recursive; we note that the number of recursive calls will be bounded by the size of the input formula φ, with each call requiring only polynomial space. Termination and correctness are immediate from construction; it only remains to note that we can loop through the enabled initial guarded commands (lines 3–7) and the enabled update guarded commands (lines 19–23) in polynomial space. □

So, under standard complexity theoretic assumptions, model checking Coalition Logic against SRML models is "easier" than full ATL. Suppose we consider other restrictions on the logic that we check.

THEOREM 4. *The propositional logic model checking problem for* SRML *is co-*NP*-complete.*

PROOF. For membership, simply note that we can universally select all initial j.g.c.s $\langle \gamma_1, \ldots, \gamma_m \rangle$, and check that if $\langle \gamma_1, \ldots, \gamma_m \rangle$ is satisfied by the empty truth assignment $\theta_\emptyset$, then the interpretation $\theta_\emptyset \oplus \{\gamma_1, \ldots, \gamma_m\}$ satisfies φ. For completeness, we can reduce the problem TAUT, of checking that a propositional logic formula is a tautology, i.e., satisfied under all truth assignments. For each Boolean variable x appearing in the input instance φ, we create an agent controlling x, with two initial guarded commands, both enabled by the empty assignment $\theta_\emptyset$, which set x to $\top$ and $\bot$ respectively. The formula to be checked against this system is simply the TAUT instance φ. The initial states of the system thus constructed

correspond to all valuations of the variables of φ, and hence φ is a tautology iff φ holds in all initial states of the system. □

5. DISCUSSION

In this paper, we have shown that, for three important subsets of ATL (including ATL itself), the model checking problem has exactly the same complexity as the corresponding theorem proving problem, assuming that models are represented using SRML, a simplified version of RML [1]. This seems to us to be a striking result: *practical* model checking for ATL and Coalition Logic has the same complexity as theorem proving for these logics. We note that others have pointed out that ATL model checking is more complex than it might appear [13], and other results hint that practical ATL model checking might be more complex than it appears at first sight [5, 6]: we have obtained tight bounds on the complexity of these problems for a specific, practical representation for models.

It is important to note that the SRML language is *not* contrived: it is a strict subset of the RML language that is used by several practical model checking systems, including MOCHA, the ATL model checker [3]. Indeed, SRML is arguably the smallest "useful" subset of RML that one can imagine: it is hard to imagine how one could simplify it without making it unusable in practice. It is also worth noting that the guarded command structures used to define SRML models are also used (modulo syntactic differences) for the same purpose in other model checking systems such as SPIN [12] and SMV [17].

We note that one of the motivations for developing the Coalition Logic of Propositional Control (CL-PC), a simpler modal variation of ATL [11], was that the structure of controlled variables in languages like RML permitted a more direct (and simpler) semantics to cooperation modalities. This link – between the interpretation of cooperation modalities and the languages used to define models – is worth exploring further.

6. REFERENCES

[1] R. Alur and T. A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(11):7–48, July 1999.

[2] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, September 2002.

[3] R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Taşiran. Mocha: Modularity in model checking. In *CAV 1998: Tenth International Conference on Computer-aided Verification, (LNCS Volume 1427)*, pages 521–525. Springer-Verlag: Berlin, Germany, 1998.

[4] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press: Cambridge, MA, 2000.

[5] L. de Alfaro, T. A. Henzinger, , and F. Y. C. Mang. The control of synchronous systems. In *CONCUR 2000: Concurrency Theory, 11th International Conference (LNCS Volume 1877)*, pages 458–473. Springer-Verlag: Berlin, Germany, 2000.

[6] L. de Alfaro, T. A. Henzinger, , and F. Y. C. Mang. The control of synchronous systems part ii. In *CONCUR 2001: Concurrency Theory, 12th International Conference (LNCS Volume 2154)*, pages 566–580. Springer-Verlag: Berlin, Germany, 2001.

[7] G. van Drimmelen. Satisfiability in alternating-time temporal logic. In *Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 208–217, Ottawa, Canada, 2003.

[8] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science Volume B: Formal Models and Semantics*, pages 996–1072. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[9] V. Goranko. Coalition games and alternating temporal logics. In J. van Benthem, editor, *Proceeding of the Eighth Conference on Theoretical Aspects of Rationality and Knowledge (TARK VIII)*, pages 259–272, Siena, Italy, 2001.

[10] J. Y. Halpern and M. Y. Vardi. Model checking versus theorem proving: A manifesto. In V. Lifschitz, editor, *AI and Mathematical Theory of Computation — Papers in Honor of John McCarthy*, pages 151–176. The Academic Press: London, England, 1991.

[11] W. van der Hoek and M. Wooldridge. On the logic of cooperation and propositional control. *Artificial Intelligence*, 164(1-2):81–119, May 2005.

[12] G. Holzmann. The Spin model checker. *IEEE Transactions on Software Engineering*, 23(5):279–295, May 1997.

[13] W. Jamroga and J. Dix. Do agents make model checking explode (computationally)? In M. Pechoucek, P. Petta, and L. Z. Varga, editors, *Multi-Agent Systems and Applications IV (LNAI Volume 3690)*, 2005.

[14] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63(2-3):185–219, 2004.

[15] M. Kacprzak and W. Penczek. A SAT-based approach to unbounded model checking for alternating-time temporal epistemic logic. *Synthese*, 142(2):203–227, November 2004.

[16] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching time model checking. *Journal of the ACM*, 47(2):312–360, March 2000.

[17] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers: Dordrecht, The Netherlands, 1993.

[18] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley: Reading, MA, 1994.

[19] M. Pauly. *Logic for Social Software*. PhD thesis, University of Amsterdam, 2001. ILLC Dissertation Series 2001-10.

[20] M. Pauly. A logical framework for coalitional effectivity in dynamic procedures. *Bulletin of Economic Research*, 53(4):305–324, 2002.

[21] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002.

[22] F. Raimondi and A. Lomuscio. The complexity of symbolic model checking temporal-epistemic logics. In *Proceedings of Concurrency, Specification & Programming (CS&P)*, Ruciane-Nida, Poland, September 2005.

[23] M. Ryan and P.-Y. Schobbens. Agents and roles: Refinement in alternating-time temporal logic. In J.-J. Ch. Meyer and M. Tambe, editors, *Intelligent Agents VIII: Proceedings of the Eigth International Workshop on Agent Theories, Architectures, and Languages, ATAL-2001 (LNAI Volume 2333)*, pages 100–114, 2002.

[24] L. J. Stockmeyer and A. K. Chandra. Provably difficult combinatorial games. *SIAM Journal of Computing*, 8(2):151–174, 1979.

[25] D. Walther, C. Lutz, F. Wolter, and M. Wooldridge. ATL satisfiability is indeed EXPTIME-complete, 2005. accepted for publication.

Knowing How to Play: Uniform Choices in Logics of Agency

Andreas Herzig
CNRS, Institut de Recherche en Informatique de Toulouse
Université Paul Sabatier
Toulouse, France
herzig@irit.fr

Nicolas Troquard
Institut de Recherche en Informatique de Toulouse
Université Paul Sabatier, Toulouse, France
and
Laboratorio di Ontologia Applicata
Università degli Studi di Trento, Trento, Italy
troquard@irit.fr

ABSTRACT

Reasoning about capabilities, strategies and knowledge is important in the analysis of multiagent systems. Alternating-time Temporal Epistemic Logic (ATEL) was designed with this aim. Nevertheless, the original interpretation of the language suffered from some counterintuitive properties. These are due to the fact that the strategies the agent applies in worlds that he cannot distinguish may not be uniform, in the sense that the same action is applied in all indistinguishable worlds. Several refinements of the original ATEL semantics were proposed since then. In this paper we argue that the STIT framework can easily account for uniform strategies. STIT is a logic of agency that has been proposed in the 90ies in the domain of philosophy of action. It is the logic of constructions of the form "agent a sees to it that φ". To support our claim, we first present a straightforward solution in STIT logic augmented by a modal operator of knowledge. Then we offer a simplification, by introducing a modal logic of knowledge-based uniform agency, for *one-step* strategies, alias choices.

Categories and Subject Descriptors

I.2.11 [**Distributed artificial intelligence**]: multiagent systems

General Terms

Theory

Keywords

formal models of agency, logics for agent systems, modal logic

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for pro t or commercial advantage and that copies bear this notice and the full citation on the rst page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior speci c permission and/or a fee.
AAMAS'06 May 8 12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1›59593›303›4/06/0005 ...$5.00.

1. INTRODUCTION

In the last years there has been increasing interest in logics enabling reasoning about strategies of agents and coalitions of agents, and the agents' knowledge about such strategies. Such logics combine two kinds of modal logics:

- logics of knowledge such as S5, and multiagent versions thereof; such logics have modal operators K_a, where $K_a\varphi$ is read "agent a knows that φ";
- logics of agency, including in particular Coalition Logic (CL) and Alternating-time Temporal Logic (ATL); such logics have constructions such as CL's $[A]\varphi$ or ATL's $\langle\langle A\rangle\rangle\mathbf{X}\varphi$, both (roughly) reading "group of agents A has an action to ensure that φ holds (whatever the other agents choose to do)".

While each of these logics is by now well-established, the interaction between knowledge and agency is less consensual. A straightforward combination of for example ATL and epistemic logic (called ATEL) was proposed in [14]. In ATEL one can express things such as "agent a has an action to ensure that φ, but ignores that". It turned out that ATEL has some counterintuitive properties. The problem can be highlighted by the following example.

EXAMPLE 1. *There is a switch, a lamp, and a blind agent a_1, which ignores whether the light is on or off. a_1 can toggle the switch (and he knows that), and a_1 can remain passive.*

Clearly, $\langle\langle\{a_1\}\rangle\rangle\mathbf{X}\texttt{light}$ holds here, i.e., a_1 can ensure that the light is on (viz. by toggling the switch if the light is off, and by doing nothing if the light is already on). We should also be able to conclude that a_1 does not *know* which action to perform in order to do this.

ATEL makes us conclude here that $K_{a_1}\langle\langle\{a_1\}\rangle\rangle\mathbf{X}\texttt{light}$, i.e. the blind agent a_1 knows that he has an action to ensure the light is on. The problem is that this action corresponds to w a strategy that is what has been called *non-uniform*: it makes a_1 choose different actions in possible worlds that are indistinguishable for him.

Multiagent variants of our example can be devised.

EXAMPLE 2. *There are two blind agents and two toggle switches such that the light is on exactly when both switches are in the same position. a_1 can toggle the first switch and a_2 the second one.*

Contrary to intuitions, ATEL makes us conclude that there is both distributed and common knowledge that there is a joint strategy to switch the light on. Another version that we will study involves one blind agent and one lame agent.

EXAMPLE 3. *There is a switch, a lamp, a blind agent a_1 and a lame agent a_2 (who knows whether the light is on, but cannot toggle). Initially the light is off. This is known by a_2, and ignored by a_1.*

This example is less problematic since there is distributed knowledge about a joint strategy, however ATEL also claims that there is common knowledge.

Several authors have proposed modified versions of ATEL, trying to accommodate in one way or another the notion of uniform strategy [8, 9, 15]. It seems to be fair to say that all these tentatives resulted in rather complex formalisms with heavy notations, and that there is no consensus up to now what the appropriate logic of knowledge and strategies is.

In this paper we take as our starting point a slightly different logic of agency that has been developed in philosophical logic. Just as ATL, the logic of "Seeing To It That" (STIT) is a modal logic enabling us to speak about time and agents' choices of actions. In STIT, CL's and ATL's $\exists$-$\forall$-quantification ("*there is* a strategy of group A such that *for all* actions of the other agents") is split up into two different modal operators:

- an operator of historical possibility $\Diamond$;
- an operator of "seeing to it that" *Stit*.

In previous work [3] we have shown that STIT is at least as powerful as Coalition Logic (CL). We have proved this by giving a translation t from CL into STIT. The main clause of the translation maps CL's $[A]\varphi$ ("group A has an action to ensure that φ") into STIT's $\Diamond Stit_A \mathbf{X} t(\varphi)$ ("it is possible that group A sees to it that next $t(\varphi)$"). This translation can be extended to ATL [4].

If by coalition we mean a set of agents that are able to deliberate about their knowledge and the collective strategy to adopt then reasoning about *common knowledge* does not seem to be relevant when dealing with coalitions. Suppose a_1 cannot distinguish two worlds w_1 and w_2, and a_2 cannot distinguish w_2 and w_3, by common knowledge they together cannot distinguish w_1, w_2 and w_3. However, when standing at w_2, agents a_1 and a_2 by *communicating*, should be able to discriminate incompatible worlds: here w_1 and w_3. By definition, the intersection of a_1's and a_2's knowledge state corresponds to distributed knowledge of the coalition $\{a_1, a_2\}$.

We assume that a coalition can ensure φ if by *sharing their knowledge* and acting together they can ensure that φ. For that purpose, we introduce an operator $Kstit_A\varphi$ of *knowledge-based agency*. "Agent a knows an action to ensure that φ" can thus be expressed in our epistemic extension of STIT by: "there is an action such that the agent knows that φ holds next". In formulas: $\Diamond Kstit_a \mathbf{X}\varphi$. Thus in Example 1 we expect $\neg\Diamond Kstit_{a_1}\mathbf{X}\mathtt{light} \wedge K_{a_1}\Diamond Stit_{a_1}\mathbf{X}\mathtt{light}$ to hold. This corresponds to the classical distinction between de re sentences such as $\exists x K_a p(x)$ and de dicto sentences such as $K_a \exists x p(x)$.

We present a discrete version of STIT logic in Section 2 and augment it with an epistemic operator in Section 3. In Section 4, we show how the problem of enforcing uniform strategies can be treated straightforwardly in the resulting framework. We simplify it in Section 5 by presenting a logic of knowledge-based agency for groups of agents. Section 6 offers a formulation of "knowing how to play" and examples to illustrate the logic. We show in Section 7 how to handle static knowledge (or knowledge in the sense of Hintikka), and future work is discussed in Section 8.

2. DISCRETE STIT FRAMEWORK

The semantics of STIT is embedded in the branching time framework. It is based on structures of the form $\langle W, <\rangle$, in which W is a nonempty set of moments, and $<$ is a tree-like ordering of these moments, such that for any w_1, w_2 and w_3 in W, if $w_1 < w_3$ and $w_2 < w_3$, then either $w_1 = w_2$ or $w_1 < w_2$ or $w_2 < w_1$.

A maximal set of linearly ordered moments from W is a *history*. $w \in h$ denotes that the moment w is *on* the history h. We define $Hist$ as the set of all histories of a STIT structure. $H_w = \{h | h \in Hist, w \in h\}$ denotes the set of histories passing through w. An *index* is a pair w/h, consisting of a moment w and a history h from H_w (i.e. a history and a moment in that history). Moreover, we make the following hypothesis (that is not made in the original STIT for reasons of generality):

HYPOTHESIS 1 (DISCRETENESS). $<$ *is* discrete.

That is to say, given a moment w_1 and a history h with $w_1 \in h$, there exists a successor moment $w_2 \in h$ such that $w_1 < w_2$, and there is *no* moment w_3 such that $w_1 < w_3 < w_2$.

2.1 Models of individual agency

A *STIT-model* is a tuple $\mathcal{M}_{\mathsf{STIT}} = \langle W, Choice, <, v\rangle$, where:

- $\langle W, <\rangle$ is a branching time structure;
- $Choice : \mathcal{A}gt \times W \to 2^{2^{Hist}}$ is a function mapping each agent and each moment w into a partition of H_w;
- v is a valuation function $v : \mathcal{A}tm \to 2^{W \times Hist}$.

The equivalence classes belonging to $Choice_a^w$ can be thought of as possible choices or actions available to agent a at w.

Given a history $h \in H_w$, $Choice_a^w(h)$ represents the particular choice from $Choice_a^w$ containing h, or in other words, the particular action performed by a at the index w/h. We must have $Choice_a^w \neq \emptyset$ and $Q \neq \emptyset$ for every $Q \in Choice_a^w$.

DEFINITION 1 (CURRENT MOMENT / CURRENT CHOICE). *At index w/h we shall call w the* current moment *and $Choice_a^w(h)$ the* current choice/action.

In STIT-models, moments may have different valuations, depending on the history they are living in (cf. [7, footnote 2 p. 586]). Thus, at any specific moment, we have different valuations. This was first suggested by Prior and Thomason, and permits a natural evaluation of statements about the future.

2.2 Models of group agency

In order to deal with group agency, Horty defines in [6, section 2.4], the notion of collective choice. Horty first introduces *action selection functions* s_w from $\mathcal{A}gt$ into 2^{H_w} associating to each $w \in W$ and $a \in \mathcal{A}gt$, some $s_w(a) \in Choice_a^w$. So, a given selection function s_w selects a particular action for each agent at w.

Then, for a given w, $Select_w$ is the set of all selection functions s_w. For every $s_w \in Select_w$, it is assumed that $\bigcap_{a \in \mathcal{A}gt} s_w(a) \neq \emptyset$. This constraint corresponds to the hypothesis that the agents' choices are independent, in the sense that agents can never be deprived of some choice due to the choices made by other agents.[1] Moreover, we suppose that intersection of *all* agents' choices must exactly be the set of histories passing through an immediate next moment:

HYPOTHESIS 2 (DETERMINISM). $\forall w \in W, \exists w' \in W\,(w < w'$ *and* $\nexists w'' \in W, w < w'' < w', \bigcap_{a \in \mathcal{A}gt} s_w(a) = H_{w'})$

As explained for ATL in [5], this determinism is not a limitation of the modeling capabilities of the language, since we could introduce a neutral agent *nature*, in order to accommodate non-deterministic transitions.

Using selection functions s_w, the *Choice* function can be generalized to apply to groups of agents ($Choice : 2^{\mathcal{A}gt} \times W \rightarrow 2^{2^{Hist}}$). A collective choice for a group of agents $A \subseteq \mathcal{A}gt$ is defined as:

$$Choice_A^w = \{\bigcap_{a \in A} s_w(a) | s_w \in Select_w\}.$$

Again, $Choice_A^w(h) = \{h' |$ there is $Q \in Choice_A^w$ such that $h, h' \in Q\}$.

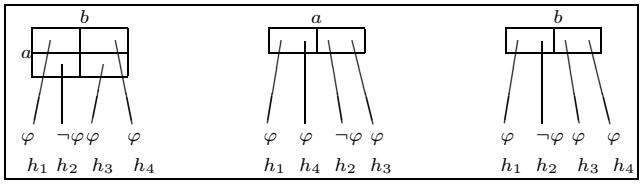


Figure 1: Group agency. $Choice_a^w = \{\{h_1, h_4\}, \{h_2, h_3\}\}$, $Choice_b^w = \{\{h_1, h_2\}, \{h_3, h_4\}\}$, $Choice_{\{a,b\}}^w = \{\{h_1\}, \{h_4\}, \{h_2\}, \{h_3\}\}$.

In Figure 1, a can choose between the upper and the lower row, and b can choose between the left and the right column. In the left model we can see that when a and b opt for a choice, the outcome is deterministic. However, actions of individual agents are not deterministic, as we can see on the two others schemes. And while both a and b can *see to it* that φ independently, they need to form a coalition in order to ensure $\neg\varphi$.

2.3 Truth conditions

[1]Note that from this constraint it follows that two agents cannot possibly have an identical set of choices at the same moment. It also follows that there are not less than $\prod_{a \in \mathcal{A}gt} |Choice_a^w|$ histories passing through a moment w. Moreover, at moments where the minimal number of histories satisfies this constraint, choices at future moments will be vacuous.

We write $\models_{\mathsf{STIT}} \varphi$ if $\mathcal{M}, w/h \models \varphi$ for every STIT-model $\mathcal{M}$, history h in $\mathcal{M}$ and moment w in h. A formula is evaluated with respect to a model and an index.

$$\begin{array}{lll} \mathcal{M}, w/h \models p & \iff & w/h \in v(p), p \in \mathcal{A}tm. \\ \mathcal{M}, w/h \models \neg\varphi & \iff & \mathcal{M}, w/h \not\models \varphi \\ \mathcal{M}, w/h \models \varphi \vee & \iff & \mathcal{M}, w/h \models \varphi \text{ or } \mathcal{M}, w/h \models \end{array}$$

Historical necessity (or inevitability) at a moment w in a history is defined as truth in all histories passing through w:

$$\mathcal{M}, w/h \models \Box\varphi \iff \mathcal{M}, w/h' \models \varphi, \forall h' \in H_w.$$

When $\Box\varphi$ holds at w then φ is said to be *settled true at* w. $\Diamond\varphi$ is defined in the usual way as $\neg\Box\neg\varphi$, and stands for historical possibility.

There are several STIT operators; we here introduce the so-called Chellas' STIT [7] which is defined as follows:

$$\mathcal{M}, w/h \models Stit_A\varphi \iff \mathcal{M}, w/h' \models \varphi, \forall h' \in Choice_A^w(h).$$

Intuitively it means that group A's current choices ensure φ, whatever other agents outside A do.[2] $Stit_A\varphi$ corresponds to $[A\,cstit\colon \varphi]$ in original notation.

As it is shown in [7], both Chellas' STIT and historical necessity are S5 modal operators, and $\models_{\mathsf{STIT}} \Box\varphi \rightarrow Stit_A\varphi$.

Since we here have discrete time, we can define the temporal operator $\mathbf{X}$ (*next*). We also introduce operator $\mathcal{U}$ (*until*):

$$\begin{array}{lll} \mathcal{M}, w/h \models \mathbf{X}\varphi & \iff & \exists w' \in h\,(w < w', \mathcal{M}, w'/h \models \varphi, \\ & & \nexists w'' \in h\,(w < w'' < w')). \\ \mathcal{M}, w/h \models \varphi\mathcal{U} & \iff & \exists w' \in h\,(w < w', \mathcal{M}, w''/h \models \quad , \\ & & \forall w''\,(w \quad w'' < w', \mathcal{M}, w''/h \models \varphi) \end{array}$$

$\mathbf{F}\varphi$ ("at some point in the future φ") is an abbreviation of $\top\mathcal{U}\varphi$, and $\mathbf{G}\varphi$ ("always in the future φ") is $\neg F\neg\varphi$.

3. ADDING EPISTEMIC FEATURES

In this section, we add imperfect knowledge to the STIT framework, assuming a particular structure of epistemic relations which will be constructed above agent choices' relations.

As a related work, we refer the reader to Pacuit et al. [11], which is as far as we know, the only to deal with epistemic notions in the STIT framework: an agent a is said to know that φ at a history h, if φ is true in every history in a's *local view*. We can enlighten differences with our notion of knowledge by the motivations, which are to investigate the relation between knowledge and obligation in Pacuit et al.'s work, while we are interested in epistemic based uniform agency.

3.1 Individual knowledge

If two moments are not distinguishable by an agent according to his knowledge, these moments are said to be *epistemically equivalent*.

Since the purpose is here to reason about actions and choices under imperfect knowledge, we should be able to identify identical choices (or choices involving actions of the same type)[3] at epistemically equivalent worlds.

[2]The more complex *deliberative* STIT can be defined as $Dstit_A\varphi =_{def} Stit_A\varphi \wedge \neg\Box\varphi$ [7].

[3]Intuitively, two actions are of the same *type* if *the way to*

Thus, in order to identify choices of the same type executable in moments epistemically equivalent, let $R_{UC_a} : W \times Hist \to 2^{W \times Hist}$ be a relation between indexes lying in equivalent choice types of epistemically equivalent moments for agent a. UC stands for "*U*niform *C*hoice". If two moment/history pairs are linked by a R_{UC_a} relation, the same type of choice is done by agent a at both indexes. We require

$$Choice_a^w(h) \quad \{h' \mid w'/h' \in R_{UC_a}(w/h)\}.$$

In Example 1, $R_{UC_{a_1}}$ respectively links indexes of the two moments that are possible for a_1 where the action `toggle` is executed, and indexes where a_1 does nothing (action `skip`).

For the sake of brevity we introduce $R_{UC} : \mathcal{A}gt \to (W \times Hist \to 2^{W \times Hist})$, which is a function associating a R_{UC_a} relation with each agent a. Hence, we obtain an *Epistemic STIT-model* (*ESTIT-model*) by adding R_{UC} to a *STIT-model*:

$$\mathcal{M}_{\mathsf{ESTIT}} = \langle W, <, Choice, R_{UC}, v \rangle$$

We can then offer a new kind of STIT operator $Kstit_a\varphi$, which reads "agent a knows that he ensures that φ". $Kstit_a$ can be seen as a Chellas' STIT adapted to imperfect knowledge:

$$\mathcal{M}, w/h \models Kstit_a\varphi \iff \forall w'/h' \in R_{UC_a}(w/h),\ \mathcal{M}, w'/h' \models \varphi.$$

It says that an agent ensures φ uniformly with its knowledge if among indistinguishable moments, and whatever the *current* one is, $Stit_a\varphi$ is true by doing the *current* choice type. Just as (static) knowledge, $Kstit_A$ is an $S5$ modality.

Hypothesis 3. *Two moments w_1 and w_2 are indistinguishable for an agent a only if for every choice open to a at w_1, there is a choice of the same type open to him at w_2.*

A corollary is that each agent knows which choices are open to him, or still that two moments with different sets of choices cannot be epistemically indistinguishable. Some authors have already used such an assumption ([13, 11]).

3.2 Group knowledge

We assume that agents of a group share their knowledge. Distributed knowledge of a group A is usually defined as the intersection of knowledge of individual agents of A. Hence,

$$R_{UC_A}(w/h) = \bigcap_{a \in A} R_{UC_a}(w/h).$$

In a straightforward manner, we will thus say that a group of agents A knows it ensures φ if among their indistinguishable moments, and whatever the current one is, $Stit_A\varphi$ is true by doing the *current* action type.

$$\mathcal{M}, w/h \models Kstit_A\varphi \iff \forall w'/h' \in R_{UC_A}(w/h),\ \mathcal{M}, w'/h' \models \varphi.$$

In Example 3 we intuitively have that $\Diamond Kstit_{\{a_1,a_2\}}\mathbf{X}\texttt{light}$, while in Example 2 we have $\neg\Diamond Kstit_{\{a_1,a_2\}}\mathbf{X}\texttt{light}$.

produce them or *the bodily movement part of the action* is the same. For example *turning off the light*, *turning on the light*, *toggle the switch* or *push the switch* are of the same type. Our aim is not to debate about types, so we stay at this intuitive definition.

4. UNIFORM STRATEGIES

In this section, we expose the problem of *uniform strategy* with a short summary of the state of the art of existing solutions. In order to support our claim that the STIT framework is appropriate for that purpose, we then show that we can adapt the semantics of the strategic STIT to obtain a rather simple solution to that problem.

4.1 The uniform strategies problem

Van der Hoek and Wooldridge's Alternating-time Temporal Epistemic Logic (ATEL) [14] was designed with the aim of reasoning about strategies and knowledge. Nevertheless, the original interpretation of the language suffered from counterintuitive properties. Situations such as Example 1 cannot be captured in such a way.

Some refinements were proposed [8, 9, 15]. They all aim at capturing the notion of a *uniform strategy*. Informally, a strategy σ is uniform if, when it is defined on a moment w epistemically equivalent to a moment w', then σ is also defined on w' and the choice type planned by the strategy is the same in both moments. In Example 1, a_1 has *no* uniform strategy to ensure `light`.

As far as we know, the most complete answer to the problem is offered in [9]. The authors propose two extensions to ATEL: ATOL, a logic of observation for agents with bounded recall of the past, and ATEL-R* for agents with both perfect and imperfect recall, and possibly reasoning about the past. Both formalisms are rather complex. In particular, the agency operator $\langle\langle A \rangle\rangle$ is indexed by epistemic formulas. This is corrected in a very recent work by Jamroga and Ågotnes [10]. They propose an interesting solution, but at the price of a non-standard negation operator (called *constructive* negation) which seems necessary to handle reasoning about non-strategic properties in their semantics.

4.2 Strategies and uniform strategies in STIT

Horty [6] and Belnap et al. [2] introduce strategies into STIT theory.

Definition 2. *A* strategy *for an agent a is a partial function σ such that $\sigma(w) \in Choice_a^w$ for each moment w from $Dom(\sigma)$, the domain of σ.*[4]

As we can see in the definition of the *Stit* operator, an agent's choice restricts the set of possible futures, in particular it restricts the histories to those corresponding with the choice being made. We expect a strategy to be a generalization of this, restricting histories to a set compatible to every choices at every moment of the domain.

Definition 3. *We say that a strategy σ* admits *a history h if and only if*

- *$Dom(\sigma) \cap h \neq \emptyset$ and*
- *for each $w \in Dom(\sigma) \cap h$ we have $h \in \sigma(w)$.*

The set of all histories admitted by a strategy σ is noted $Adh(\sigma)$.

We shall often use the notation σ_a to name a particular strategy of an agent a.

[4] σ is a partial function because there is no need to represent in σ an agent's choices in moments he will never reach if he follows σ.

DEFINITION 4 (COLLECTIVE STRATEGY). *A collective strategy for A $\mathcal{A}gt$ is a tuple $\sigma_A = \langle \sigma_a \rangle_{a \in A}$ (one strategy σ_a for every agent), and $Adh(\sigma_A) = \bigcap_{a \in A} Adh(\sigma_a)$.*

[6] proposes a characterization of a sort of properly-formed strategies with limited scope. It is necessary to introduce the notion of *field* at a moment w, which is a subset containing w of the subtree $Tree_w = \{w' \mid w < w' \text{ or } w = w'\}$ with a downward closure. With $Adm(\sigma) = \{m \mid m \in h, h \in Adh(\sigma)\}$, a strategy is properly-formed in M if it is *complete in the field* ($Adm(\sigma) \cap M$ $Dom(\sigma)$) and *irredundant* ($Dom(\sigma)$ $Adm(\sigma)$). Thus, an *ability* operator should be evaluated with respect to a field. But for simplicity, we do not use it here.[5]

However, just as it is done in [6] for "properly-formed" strategies, we here constrain strategies open to a group A at moment w to a set $Strategy_A^w$, such that every strategy in it is uniform. For notational convenience we say that two moments w_1 and w_2 are *indistinguishable*, if and only if every choice in w_1 has an indistinguishable choice in w_2, formally stated by

$w_1 \in R_{K_A}(w_2)$ iff $\forall h \in H_{w_1}, \exists h' \in H_{w_2}, w_1/h \in R_{UC_A}(w_2/h')$.

R_{K_A} is a relation of equivalence among moments.

$$Strategy_A^w = \{\sigma \mid \quad Dom(\sigma) \quad \bigcup_{w' \in R_{K_A}(w)} Tree_{w'}, \text{ and } \forall w' \in Dom(\sigma), \forall w'' \in R_{K_A}(w'), (w'' \in Dom(\sigma) \text{ and } \exists h' \in \sigma(w')), \exists h'' \in \sigma(w''), w''/h'' \in R_{UC_A}(w'/h'))\}$$

The set $Strategy_A^w$ contains strategies defined on moments lying in the future of the indistinguishable moments of w (included). Those strategies, if defined at a moment w' must also be defined at each moment w'' indistinguishable from w', where the strategy must apply the same choice type. The semantics of the operator for *S*trategic *A*bility with imperfect knowledge is then:

$$\mathcal{M}, w/h \models SAstit_A\varphi \iff \exists \sigma_A \in Strategy_A^w, \forall h' \in Adh(\sigma_A), \forall w' \in R_{K_A}(w) \cap h', \mathcal{M}, w'/h' \models \varphi$$

A possible reading is : "A has the ability to guarantee the truth of φ by carrying out an available strategy, and knows it". $SAstit_A\varphi$ corresponds to $\Diamond[A\,scstit : \varphi]$ in the original notation. We believe that this notation can be misleading, since the operator is atomic: Horty refers to it as a *fused* operator.

Figure 2 presents a scenario where a *blind* agent a enters a room. He does not know whether the light is on. At moments w_2 and w_3 he has the choice between switching the light (action `toggle`), remaining passive (action `skip`) or touching the light bulb (action `touch`). This last action is *informative*. Let strategy $\sigma = \{\langle w_1, \{h_1 \dots h_{12}\}\rangle, \langle w_2, \{h_3, h_4\}\rangle, \langle w_3, \{h_9, h_{10}\}\rangle, \langle w_5, \{h_4\}\rangle, \langle w_8, \{h_9\}\rangle\}$.[6] We can ckeck that σ is indeed in $Strategy_{\{a\}}^{w_1}$, i.e., that it is a uniform strategy for a at the moment w_1, and $Adh(\sigma) = \{h_4, h_9\}$. l holds

[5]It is easy to see that even histories are superfluous. We nevertheless keep them for uniformity.

[6]The notation is abusive: σ should be defined on the complete set $\{w_1 \dots w_{12}\}$, but it does not influence the set of admitted histories.

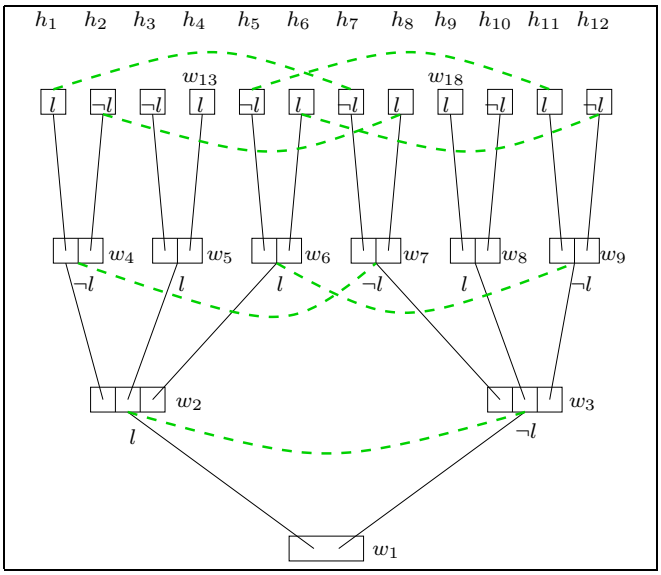


Figure 2: Suppose a blind agent a. For clarity, dotted lines are intended for R_{K_a} relations over moments, and R_{UC_a} relations can be deduced since choices of indistinguishable moments are the same: at w_1 $\{h_1 \dots h_{12}\}$ is for `enter`, at w_2 $\{h_1, h_2\}$ is for `toggle`, $\{h_3, h_4\}$ for `touch` and $\{h_5, h_6\}$ for `skip` (and identically for choices at w_3), at w_4 $\{h_1\}$ is for `toggle` and $\{h_2\}$ for `skip` (the same for moments $w_2 \dots w_9$). l stands for "the light is on", and time goes upward.

at both w_{13}/h_4 and w_{18}/h_9. Thus $\mathbf{F}l$ is true at w_1/h_4 and w_1/h_9 and it follows that considering σ, agent a knows how to play to ensure that the light is on in the future: $SAstit_a\mathbf{F}l$.

We can thus easily grasp the notion of uniform strategies in the STIT framework. This is due to the fact that the underlying semantics is more versatile than that of ATEL, because it allows to relate a moment/history pair w_1/h_1 to another w_2/h_2 via R_{UC_a} without relating all w_1/h_1' to w_2/h_2'. We here have changed the domain of quantification over strategies such that we just consider strategies satisfying the definition of uniformity. In the following, and for this preliminary work, we decide to simplify the framework into a STIT-like one, for *one-step* strategies.

5. TOWARDS A SIMPLIFICATION

The STIT branching time is based on the so called T × W logic for *historical necessity* (see [1]). Such a logic is a combination of tense and modal logic for worlds with the same time order. T×W semantics can be seen as a grid with tense on one dimension and set of histories on the other. In Figure 3 we have a grid with temporal aspects represented longitudinally, time going upward.

The aim of this section is to simplify the STIT-framework with imperfect knowledge into a semantics that does not involve histories. We focus on *one-step* strategies where the problem of uniform strategies already appears.

5.1 A logic of knowledge based uniform agency

We here deal with a framework permitting us to talk about agency, and more precisely, about agency with respect to

knowledge about actions. Thus, we do not deal with static knowledge in Hintikka's tradition: we are not interested here in sentences like "Mary knows that the light is on", but in sentences such as "Mary knows her current action makes sure that the light will be on". Nevertheless, we will briefly see in Section 7 how we can model the first case in that framework, too.

We propose a logic for *knowledge-based uniform agency* in which the knowledge scope stops at effects of actions. We keep on using the terminology of STIT. Thus, historically possible *indexes* lie in a same *moment*.

5.2 Syntax and Semantics

Given a finite set of agents $\mathcal{A}gt$ and a set of atomic propositions $\mathcal{A}tm$, a formula φ can have the following syntactic form:

$$\varphi ::= \bot \mid \mathrm{p} \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \Box\varphi \mid Kstit_A\varphi \mid Stit_A\varphi$$

with $p \in \mathcal{A}tm$ and $A \quad \mathcal{A}gt$.

A model is a tuple $\mathcal{M} = \langle W, R_X, R_\Box, R_{UC}, v\rangle$, where:

- W is a nonempty set of indexes;
- $R_X : W \to W$ is total function associating an index to its historical successor;
- $R_\Box$ is a relation between indexes;
- $R_{UC} : \mathcal{A}gt \to (W \to 2^W)$ is a function associating each agent $a \in \mathcal{A}gt$ to a relation. For every agent $a \in \mathcal{A}gt$ the relation $R_{UC}(a)$ will be noted R_{UC_a};
- $v : \mathcal{A}tm \to 2^W$ is an evaluation function.

$R_\Box(w)$ denotes the set of indexes lying in the same moment as w. R_{UC_a} is the relation between indexes of identical choices lying in moments indistinguishable by agent a.

For a group of agents A, we suppose that agents of A make choices collectively with distributed knowledge: they share their knowledge. We thus define the collective uniform choice relation as follows:

$$R_{UC_A}(w) = \bigcap_{a\in A} R_{UC_a}(w).$$

We make the same hypothesis as in STIT concerning the independence/consistency of choices of agents, and assume that for each index w, $R_{UC_{\mathcal{A}gt}}(w) \neq \emptyset$.

We assume models satisfy additional properties:

1. $(R_\Box \circ R_X)(w) \supseteq (R_X \circ R_\Box)(w)$
   If it is historically necessary that some state of affairs will hold next, then next, this state of affairs will be historically necessary.
2. $(R_\Box \circ R_{UC_A})(w) = (R_{UC_A} \circ R_\Box)(w)$
   The same types of choices are available at every indistinguishable moment.
3. $(R_{UC_A} \circ R_X)(w) \supseteq (R_X \circ R_\Box \circ R_{UC_A})(w)$
   If a coalition A makes a uniform choice to ensure a state of affairs at the next step, it will be historically inevitable at the next step that A knows that it ensures φ.

The first assumption forces a tree structure of moments. The second one is the translation of Hypothesis 3. The last assumption is motivated by the fact that it is not sufficient to say that a group of agents A knows how to play to ensure φ at moment w if A does not know when it has achieved φ. Nevertheless, agents know possible effects of their actions. We then adopt the principle that if agents know they ensure φ at the next step by doing the current choice, then at the very next step they will know that φ holds. It forces a *no forget* principle which seems realistic since we deal with *one-step* strategies.

A formula is evaluated with respect to a model and a world, which corresponds to STIT indexes and thus contains information about history.

$$\begin{array}{lcl}
\mathcal{M}, w \models p & \iff & w \in v(p),\ p \in \mathcal{A}tm \\
\mathcal{M}, w \models \neg\varphi & \iff & \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \mathbf{X}\varphi & \iff & \mathcal{M}, R_X(w) \models \varphi \\
\mathcal{M}, w \models \Box\varphi & \iff & \forall w' \in R_\Box(w), \mathcal{M}, w' \models \varphi \\
\mathcal{M}, w \models Kstit_A\varphi & \iff & \forall w' \in R_{UC_A}(w), \mathcal{M}, w' \models \varphi \\
\mathcal{M}, w \models Stit_A\varphi & \iff & \forall w' \in (R_{UC_A} \cap R_\Box)(w), \\
 & & \mathcal{M}, w' \models \varphi
\end{array}$$

$w \models \mathbf{X}\varphi$ means that φ is true at the immediate successor of w on the history. $w \models \Box\varphi$ means that φ is true at each world lying in the same moment as w. $Kstit_a\varphi$ is an agency operator with an implicit epistemic feature. It means that a knows that if he performs his chosen action, he ensures that φ. $Stit_A\varphi$ still stands for "A's current choice makes sure that φ".

5.3 Some validities

$\mathbf{X}$ is a KD-modality, and $\Box$ and every $Stit_A$ and $Kstit_A$ are $S5$-modalities.

$(MonKstit)$ $Kstit_{A_1}\varphi \to Kstit_{A_2}\varphi$ if $A_1 \quad A_2$

$(MonStit)$ $Stit_{A_1}\varphi \to Stit_{A_2}\varphi$ if $A_1 \quad A_2$

$(KnowAct)$ $\Box Kstit_A\varphi \leftrightarrow Kstit_A\Box\varphi$

$(Linearity)$ $\neg\mathbf{X}\neg\varphi \to \mathbf{X}\varphi$

$(XBox)$ $\Box\mathbf{X}\varphi \to \mathbf{X}\Box\varphi$

$(NoForget)$ $Kstit_A\mathbf{X}\varphi \to \mathbf{X}\Box Kstit_A\varphi$

$(StitBox)$ $\Box\varphi \to Stit_A\varphi$

$(StitKstit)$ $Kstit_A\varphi \to Stit_A\varphi$

$(MonKstit)$ and $(MonStit)$ are valid by the construction of the collective uniform choice relation R_{UC_A}. $(KnowAct)$ is the immediate translation of the second constraint. $(Linearity)$ follows from the definition of the definition of the R_X function. $(XBox)$ is the translation of the first constraint. $(NoForget)$ is the direct translation of the third constraint.[7] $(StitBox)$ and $(StitKstit)$ are valid by the semantics of the $Stit$ operator.

[7]The link with the *no forget* principle is maybe not clear, but we say more about this in section 7.

6. KNOWING HOW TO PLAY

The fact that a group of agents A knows how to play to ensure φ at the next step can be captured by the formula

$$\Diamond Kstit_A \mathbf{X}\varphi.$$

A group of agents knows how to play to ensure φ at the next step if "A can make a choice such that it knows it ensures φ at the next step".

In order to illustrate the logic of Section 5, and particularly the capabilities of groups with incomplete knowledge, we present two scenarii constructed above examples 1 and 3. First we deal with one agent with lack of knowledge (the blind agent a_1). Then, we add another agent that cannot act (the lame agent a_2), and show how they can put their strengths together.

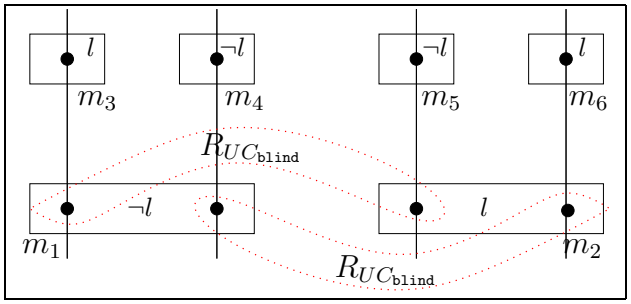


Figure 3: (Schematic of Example 1) Vertical lines represent histories, rectangles are historical necessity relations and delimit moments.

In Figure 3 the moments m_1 and m_2 are epistemically equivalent w.r.t. static knowledge. Suppose a blind person enters a room at moment m_0. The problem is to know whether he can ensure that the light is on. For that he has two possible actions, `toggle` and `skip`. If at m_1 or m_2 he can act such that the light is on, he cannot be aware of that: he has no uniform choice such that the result is always l. Let us call $w_{1,r}$ the right index of moment m_1, and $w_{1,l}$ the left one. The following holds:

- $w_{1,r} \models Stit_{\text{blind}}\mathbf{X}l \wedge \neg Kstit_{\text{blind}}\mathbf{X}l$
- $w_{1,l} \models \neg\Box\neg Stit_{\text{blind}}\mathbf{X}l \wedge \Box\neg Kstit_{\text{blind}}\mathbf{X}l$.

Figure 4 presents an example where the blind person from the previous example is helped by a lame person. While the blind does not know the current state of the light, the lame does. While the lame cannot act such that the light is on at the next step, the blind can. If we call $w_{1,l}$ the left index of the moment m_1 and $w_{1,r}$ the right one, the following holds:

- $w_{1,l} \models Stit_{\text{blind}}\mathbf{X}l \wedge \neg Kstit_{\text{blind}}\mathbf{X}l$
- $w_{1,r} \models \neg\Box\neg Stit_{\text{blind}}\mathbf{X}l \wedge \Box\neg Kstit_{\text{blind}}\mathbf{X}l$
- $w_{1,l} \models \neg Stit_{\text{lame}}\mathbf{X}l$
- $w_{1,r} \models \Box\neg Stit_{\text{lame}}\mathbf{X}l$

but,

- $w_{1,l} \models Kstit_{\{\text{blind,lame}\}}\mathbf{X}l$

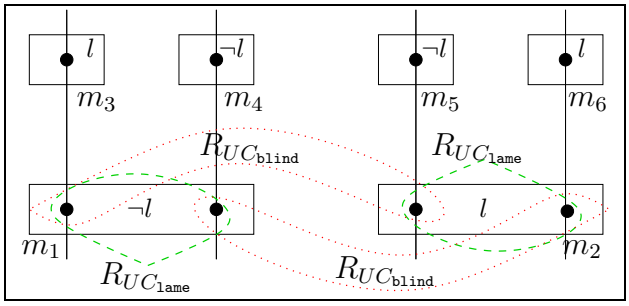


Figure 4: (Schematic of Example 3) The actual moment is m_1, where the light is off ($\neg l$). The lame agent cannot act but has complete knowledge. The blind agent does not know if the light is on or off, but can choose to toggle or not (`skip`). Individually, neither the lame agent nor the blind agent can ensure that the light is on, but together, they can form an effective coalition for that purpose.

- $w_{1,r} \models \neg\Box\neg Kstit_{\{\text{blind,lame}\}}\mathbf{X}l$

Now, we see that this last formula corresponds to the concept of *knowing how to play* that we wanted to grasp.

7. STATIC KNOWLEDGE

We have not included static knowledge in the language of our logic of knowledge-based uniform agency. However, it can be defined: we say that an agent a knows that φ if and only if whatever a chooses, it knows that it's choice makes sure that φ. We say that a group of agents A knows that φ if and only if whatever agents of A choose, they distributively know that their choices make sure that φ. We thus can define in our logic, a modal operator reading "group A knows that φ", as:

$$K_A\varphi =_{def} \Box Kstit_A\varphi.$$

"A knows that φ" is equivalent to "it is historically inevitable that A knows that it ensures φ". We also can reformulate the $(NoForget)$ validity by $Kstit_A\mathbf{X}\varphi \rightarrow \mathbf{X}K_A\varphi$, which makes sense now, and can be read as "if A knows it ensures that φ is true at the next step, then at the next step the group will know that φ holds". The following formulas are valid:

- $K_A\varphi \rightarrow \Box\varphi$
- $K_A Kstit_A\varphi \leftrightarrow K_A\varphi$
- $Kstit_A\neg\Box\neg\varphi \rightarrow K_A\neg\Box\neg\varphi$

Finally, the formulation of "a group of agents A know how to play to ensure that φ at the next step" is equivalent to

$$K_A\Diamond Kstit_A\mathbf{X}\varphi.$$

K_A focuses on what agents know at a moment, before doing a choice, or whatever the choice they do. $Kstit_A$ is about knowledge after having made a choice. The threshold from static knowledge to dynamic knowledge is monotonic: what is known before doing a choice is also known after ($K_A\varphi \rightarrow Kstit_A\varphi$ is valid). Thus $Kstit_A$ is a refinement of Hintikka's static knowledge.

This distinction is of interest even with perfect knowledge about the current world. Indeed, if Mary knows that the light is on, she is the only one to have the control on it, and that the switch is functioning well, she nevertheless ignores the state of the light at the next moment, until she makes her choice between toggling and remaining passive. But when she is committed to a choice, she knows the next state of the light, and this already before performing the action.

8. FUTURE WORK

A further extension is a strategic operator for global rather than local effectivity. Two paths open before us: (1) We can define an iterated operator $(\Diamond Kstit_A\mathbf{X})\ \varphi$, which we believe can be done in a rather simple manner, following [12, Chapter 4] and capitalizing on the *no forget* principle. However, this remains an idealization about knowledge of agents. (2) We can adapt strategies in our logic as we have done in Section 3.

Also, our aim was here to deal with cooperative coalitions of agents. We hence used distributed knowledge. It could be interesting to investigate refinements of group knowledge.

9. ACKNOWLEDGMENTS

This reasearch is part of the MOSTRO project, funded by the "Fondo provinciale per i progetti di ricerca" of the Provincia Autonoma of Trento.

We would like to thank Wojtek Jamroga and two anonymous reviewers for fruitful comments.

10. REFERENCES

[1] L. Åqvist. The logic of historical necessity as founded on two-dimensional modal tense logic. *Journal of Philosophical Logic*, 28(4):329–369, 1999.

[2] N. Belnap, M. Perloff, and M. Xu. *Facing the future: agents and choices in our indeterminist world*. Oxford, 2001.

[3] J. Broersen, A. Herzig, and N. Troquard. From Coalition Logic to STIT. In W. van der Hoek, A. Lomuscio, E. de Vink, and M. Wooldridge, editors, *Proceedings of the 3rd Int. Workshop on Logic and Communication in Multiagent Systems (LCMAS 2005)*, 2005. To appear in ENTCS.

[4] J. Broersen, A. Herzig, and N. Troquard. Embedding Alternating-time Temporal Logic in Strategic STIT Logic of Agency, 2006. Submitted.

[5] V. Goranko and W. Jamroga. Comparing semantics of logics for multi-agent systems. *Synthese*, 139(2):241–280, 2004.

[6] J. F. Horty. *Agency and Deontic Logic*. Oxford University Press, Oxford, 2001.

[7] J. F. Horty and N. D. Belnap, Jr. The deliberative STIT: A study of action, omission, and obligation. *Journal of Philosophical Logic*, 24(6):583–644, 1995.

[8] W. Jamroga. Some remarks on alternating temporal epistemic logic. In *Proc. of the Int. Workshop on Formal Approaches to Multi-Agent Systems (FAMAS'03)*, 2003.

[9] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 2004.

[10] W. Jamroga and T. Ågotnes. What agents can achieve under incomplete information. In *Proceedings of AAMAS'06*, 2006.

[11] E. Pacuit, R. Parikh, and E. Cogan. The logic of knowledge based obligation, 2005. To appear in Knowledge, Rationality and Action. Early versions were presented at DALT 2004 and SEP 2004.

[12] M. Pauly. *Logic for Social Software*. PhD thesis, University of Amsterdam, 2001. ILLC Dissertation Series 2001-10.

[13] P. Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2), 2004.

[14] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 1167–1174, New York, NY, USA, 2002. ACM Press.

[15] S. van Otterloo and G. Jonker. On epistemic temporal strategic logic. *Electronic Notes in Theoretical Computer Science*, 85(2), 2004. In Proceedings of the 2nd Int. Workshop on Logic and Communication in Multiagent Systems (LCMAS 2004).

Verifying space and time requirements for resource-bounded agents

Natasha Alechina
University of Nottingham
Nottingham, UK
nza@cs.nott.ac.uk

Piergiorgio Bertoli
ITC-IRST
Trento, Italy
bertoli@itc.it

Chiara Ghidini
ITC-IRST
Trento, Italy
ghidini@itc.it

Mark Jago
University of Nottingham
Nottingham, UK
mtw@cs.nott.ac.uk

Brian Logan
University of Nottingham
Nottingham, UK
bsl@cs.nott.ac.uk

Luciano Serafini
ITC-IRST
Trento, Italy
luciano.serafini@itc.it

ABSTRACT

We present a novel procedure for automatically verifying the space and time requirements for resource-bounded reasoning agents. We represent agents as a finite state machines in which the states correspond the formulas currently held in the agent's memory and the transitions between states correspond to applying the agent's inference rules. To check whether an agent has enough memory to derive a formula ϕ, we specify the FSM as input to the model-based planner MBP, and check whether the agent has a plan (a choice of memory allocations and inference rule applications), all executions of which lead to states containing ϕ. Our approach is general enough to admit verification of reasoners with any set of inference rules which can be encoded as transitions between FSM states.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents, Multiagent Systems*; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Deduction (e.g., natural, rule-based)*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs

General Terms

Design, Verification

Keywords

Resource-bounded agents, MBP

1. INTRODUCTION

The question of how much memory a reasoning agent needs to derive a formula is of considerable theoretical and practical interest. From a theoretical point of view, it is interesting to investigate how the deductive strength of a particular logic changes when only a fixed number of formulas are allowed to be 'active' in a derivation. This is different from the question of bounds on the size proofs [3], since we assume that the whole proof does not need to be held in memory (some intermediate steps may be overwritten).

From a practical point of view, the question of whether an agent will run out of memory or time before achieving its goal(s) is clearly a major concern for the agent developer. As agent tasks become more open ended, the amount of memory required to achieve them becomes harder to predict a priori. For example, the reasoning capabilities of agents assumed by many web service applications is non trivial (e.g., reasoning over complex ontologies or about business processes described by a set of business rules) and the memory requirements correspondingly difficult for the agent developer to determine a priori. At the same time trends towards mobile agents and agents which run on mobile devices such as PDAs and smart phones imply more processor and memory efficient agent designs.

In this paper, we present a novel procedure for automatically verifying the space and time requirements for resource-bounded reasoning agents. Specifically, we address the question: given an agent and a formula ϕ, does the agent have sufficient memory to derive ϕ, and, if it does, what is the length of the shortest derivation within the specified memory bound? In outline, our approach is as follows. We represent a reasoning agent as a finite state machine in which the states correspond the formulas currently held in the reasoner's memory and the transitions between states correspond to applying the reasoning rules. To check whether a reasoner has enough memory to derive a formula ϕ, we specify the FSM as input to the model-based planner MBP [2], and check whether the reasoner has a plan (a choice of memory allocations and inference rule applications), all executions of which lead to states containing ϕ.

Our approach is general enough to admit verification of reasoners with any set of inference rules, provided that those rules can be encoded as transitions between FSM states. To illustrate the generality of our approach, we show how to encode two example reasoners: a forward-chaining rule-based agent of the kind found in many applications employing ontological reasoning and business rules, and a classical propositional reasoner which can derive all classical consequences of its knowledge base given unlimited memory.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. FORMAL MODEL

We model resource-bounded agents as finite state machines (FSM) or transition systems. Let the internal language of the agent be some language L (e.g. propositional language). The definition of a transition system is given relative to the following components: the bound n on the agent's memory size; the agent's reasoning rules; the agent's knowledge base $K \subseteq L$; the agent's goal formula $A_G \in L$. The set of all subformulas of K and A_G will be denoted by Ω. We abstract away from the size of the formulas. However, given K and A_G, the maximum size of any formula in the agent's state is fixed.

In the remainder of this section, we first define the language and transition systems for 'definite reasoning' agents and give an example of such an agent. We then introduce a more complex logic for agents that need to maintain a set of epistemic alternatives, such as classical reasoners.

2.1 Definite reasoners

The language of the logic BML^d (for bounded memory logic, definite case) is defined relative to the agent's internal language L. If A is a formula of L, then $\mathrm{B}\,A$ (the agent believes A) is a formula; the language is closed under the boolean connectives and unary modalities EX ('there exists a next state where...') and EF ('there exists a future state where...'). Other boolean connectives are defined in the usual way. We also define $\mathrm{AX}\,\phi$ as $\neg\mathrm{EX}\,\neg\phi$ and $\mathrm{AG}\,\phi$ as $\neg\mathrm{EF}\,\neg\phi$.

A transition system $M = (S, R, V)$ consists of a set of states S, a serial binary relation R on S (transitions between states) and an assignment $V : S \longrightarrow \mathcal{P}(\Omega)$ (assigning to the state the set of formulas the agent believes in that state). Notice that $V(s)$ is not a classical truth assignment, as it might contain complex formulas, e.g., $A \wedge B$, as well as contradictory formulas, e.g., $A \wedge B, \neg B \in V(s)$. To reflect the fact that the agents have bounded memory, we postulate that V can assign at most n formulas to any given state. The transitions which the agent can make depend on the agent's inference rules. In our model, we assume that one of the agent's possible transitions is 'reading' a K formula into its memory or 'active state'. Reading a formula may correspond to reading from flash memory, asking for user input, or reading data from the server over the network.

The definition of a formula being satisfied in $M, s \in S$ is as follows:

$M, s \models \mathrm{B}\,A$ iff $A \in V(s)$

$M, s \models \neg\phi$ iff $M, s \not\models \phi$

$M, s \models \phi \wedge \psi$ iff $M, s \models \phi$ and $M, s \models \psi$

$M, s \models \mathrm{EX}\,\phi$ iff there exists a state t such that $R(s,t)$ and $M, t \models \phi$.

$M, s \models \mathrm{EF}\,\phi$ iff there exists a sequence of states $t_1, \ldots, t_k$ such that for all $i \in \{1, \ldots, k-1\}$, $R(t_i, t_{i+1})$, $t_1 = s$ and $M, t_k \models \phi$

The bound n on the size of the agent's memory is expressed by the following axiom:

B(n) $\mathrm{B}\,A_1 \wedge \ldots \wedge \mathrm{B}\,A_n \rightarrow \neg\mathrm{B}\,A_{n+1}$ where $A_i \neq A_j$ for all $i, j \in \{1, \ldots, n+1\}$ such that $i \neq j$.

We can express that the agent can derive its goal A_G from its knowledge base K as $\mathrm{EF}\,\mathrm{B}\,A_G$ (there is some future state where the agent believes A_G). The fact that a formula is derivable in k steps can be expressed as $\mathrm{EX}^k\mathrm{B}\,A_G$ (where EX^k denotes k applications of the operator EX). Similarly, the fact that an agent needs at least $k+1$ steps to derive a formula A_G can be expressed as $\mathrm{AX}^k\neg\mathrm{B}\,A_G$.

A simple example of a definite reasoner would be an agent which reasons using rules, e.g., ontology rules, or business rules. We assume that agent's knowledge base consists of ground atomic formulas and rules of the form $A_1 \wedge \ldots \wedge A_n \rightarrow B$, where $A_1, \ldots, A_n, B$ are atomic formulas (see, for example, [4]). By generating all possible substitutions of constants occurring in the knowledge base into the rule, we can reduce the knowledge base to a purely propositional set of formulas, consisting of propositional variables and implications of the form $p_1 \wedge \ldots \wedge p_n \rightarrow q$. Then the only rules the agent needs to derive all 'rule-based' consequences are conjunction introduction and modus ponens.

Let $V(s)'$ stand for any subset of $V(s)$ which has cardinality at most $n-1$ and differs from $V(s)$ in at most one formula. The rule-based reasoner has the following transitions:

Read $R(s,t)$ if $V(t) = V(s)' \cup \{A\}$ for some $A \in K$.

AND $R(s,t)$ if $A_1, A_2 \in V(s)$ and $V(t) = V(s)' \cup \{A_1 \wedge A_2\}$.

MP $R(s,t)$ if $A_1 \in V(s)$, $A_1 \rightarrow A_2 \in V(s)$, and $V(t) = V(s)' \cup \{A_2\}$

Reflexivity $R(s,s)$

A complete axiomatisation of this logic is given in [1].

2.2 More general reasoners

It is also interesting to model reasoners which can reason by cases, or in general consider hypothetical states; this means that their transitions do not necessarily follow the logical consequence relation. We also extend the language to express disbelief as well as belief.

Consider a reasoner who believes $\{A \vee B, A \rightarrow C, B \rightarrow C\}$. To derive C, it has to reason by cases: assume A; derive C. Then, assume B; derive C. Hence, it is safe to believe C. However, if the process of assuming A corresponds to a transition to a state where A is believed, the modelling is not 'safe' — the agent's beliefs are not justified by valid inference steps. In the state where it assumes A, the agent should remember that this is just one of the epistemic alternatives, and that in others A is false and B is true.

To deal with such reasoners, we add an extra set of 'epistemic alternatives' or possible worlds to each state. Epistemic alternatives are introduced when the classical reasoner is applying non-deterministic rules, such as elimination of disjunction. Intuitively, a formula is now believed in a state if it is true in all of the epistemic alternatives associated with this state. We express this as $\Box\mathrm{B}\,A$.

The language of the logic BML (for bounded memory logic) extends the language of BML^d by adding extra clauses: if A is a formula of L, then $\overline{\mathrm{B}}\,A$ (the agent disbelieves A) is a formula; if ϕ is a formula, then $\Diamond\phi$ is a formula. We also define $\Box\phi$ as $\neg\Diamond\neg\phi$.

For such general reasoners, we can express that the agent can derive A_G from its knowledge base K as $\mathrm{EF}\,\Box\mathrm{B}\,A_G$ (there is some future state where in all epistemic alternatives the agent believes A_G).

A BML transition system $M = (S, W, R, U, T, F)$ consists of a set of states S, a set of possible worlds or epistemic alternatives W, a binary relation R on S, a function assigning to each state a set of epistemic alternatives $U : S \longrightarrow \mathcal{P}(W)$, and two assignments $T : W \longrightarrow \mathcal{P}(\Omega)$ and $F : W \longrightarrow \mathcal{P}(\Omega)$ which say whether the value of an (internal language) formula in a world is true or false (where, as before Ω is the set of subformulas of K and A_G). As before, to reflect the bound on the agent's memory, we require

$|T(w)| + |F(w)| \leq n$, for any given state w. Moreover, the truth assignments should be consistent, i.e., $T(w) \cap F(w) = \emptyset$. The following truth definitions have been added or modified compared to BML^d. Note that we talk about truth in a world and truth in a state:

$M, w \models \mathrm{B}\, A$ iff $A \in T(w)$

$M, w \models \bar{\mathrm{B}}\, A$ iff $A \in F(w)$

$M, s \models \Diamond\phi$ iff there exists w in $U(s)$, such that $M, w \models \phi$.

The bound n on the size of the agent's memory is expressed by the following axiom:

B'(n) $\Box(\widetilde{\mathrm{B}}\ A_1 \wedge \ldots \wedge \widetilde{\mathrm{B}}\ A_n \rightarrow \neg\ \widetilde{\mathrm{B}}\ A_{n+1})$, where $\widetilde{\mathrm{B}}\ A_i$ stands for either $\mathrm{B}\, A_i$ or $\bar{\mathrm{B}}\, A_i$ and $A_i \neq A_j$ for all $i, j \in \{1, \ldots, n+1\}$ such that $i \neq j$,

The transition relation R between states is defined in terms of expansion relation between epistemic alternatives $\preceq$. Expansion corresponds to applying an inference rule to formulas in the epistemic alternative. Formally, $R(s,t)$ holds if $U(s) = \{w_1, \ldots, w_m\}$, and for some $w_i \in U(s)$, $U(t) = (U(s) \backslash \{w_i\}) \cup \{v : w_i \preceq v\}$.

Note that the classical reasoner agent can construct new formulas in addition to decomposing formulas. We only allow the construction of formulas which are in Ω (the set of subformulas of K and A_G). This does not affect the completeness of agent's rules (since these are the only formulas it may possibly need in the derivation of A_G from K), but allows us to represent it as a finite state machine.

Since the agent can both believe and disbelieve formulas (and its language contains negation), an issue of inconsistent possible worlds arises. An agent *cannot* make a transition to a possible world where the same formula is assigned true and false. All rules therefore have to have a proviso that if $w \preceq v$ then it impossible, for any formula A, to have $A \in T(w)$ and $A \in F(v)$ or vice versa. If the agent starts with inconsistent state, it is allowed to assert any formula from Ω. A full list of transitions the classical reasoner is given in [1].

In [1], we prove that a classical reasoner with unbounded memory can derive A_G from K whenever A_G is a classical consequence of K.

3. VERIFYING REASONING CAPABILITIES

The problem of identifying the existence (and the minimal length) of a deduction for A_G from a knowledge base K for an agent with bounded memory modelled as a transition system M, can be recast as a *planning problem*: find a control strategy for M (a plan) which, starting from any state in K, leads to some state in A_G. In general, M is a *nondeterministic* transition system, since applying a rule may lead to several epistemic alternatives. Thus, we are interested in *strong* plans [2]: tree-structured plans such that their execution leads to the goal, for *every* possible outcome of the actions in the plan.

From the few planners capable of dealing with strong planning for nondeterministic domains we selected MBP, a system which combines effective algorithms with an input language which allows a concise description of transition systems in logical terms. In this section, we provide a high-level description of how the proof existence problem is recast as a planning domain in MBP.

States are identified with assignments $v : \Omega \longrightarrow \{\top, \bot, U\}$ (where U stands for 'undefined'). The transition relation is defined as having as argument the formula to be derived or decomposed, and the formula(s) to be overwritten. For example, the precondition of $\mathbf{Read}(A, B)$ is that A is in KB, and the postcondition is that in the next state, A is true and B is undefined (the agent believes A and does not believe or disbelieve B). If the state is not full, then it is possible that B is a special 'empty' formula (nothing gets overwritten). In some cases a transition may result in more than one formula being overwritten. For example, the precondition for disjunction elimination $\mathbf{elimOr}(A, B_1, B_2)$ is that A is of the form $A_1 \vee A_2$, and that in the current state, $v(A) \neq U$. The postcondition is that in the next state, the value of $v(A_1) \vee v(A_2)$ is equal to the assignment to A in the current state, and B_1, B_2 are assigned U. Note that if in the current state A is assigned $\top$, then the effect of this transition is non-deterministic, since several different values of $v(A_1), v(A_2)$ would satisfy the constraint.

Given this encoding, the planning problem is described by an initial state where $\forall A \in \Omega : v(A) = U$, and by a goal state $v(A_G) = \top$. The constraints on transitions are directly represented in MBP as TRANS constraints over R, allowing for a compact representation of the problem.

MBP implements a number of different search styles. We chose breadth-first backward search which guarantees that the shortest plan is selected. The computational burden imposed by this search style is effectively constrained by the use of symbolic representation techniques that allow a very compact encoding, and an efficient handling of extremely large state sets; details can be found in [2].

Experimental results can be found in [1].

4. CONCLUSIONS AND FUTURE WORK

In this paper, we have attempted to take seriously the idea that reasoning is a process which requires memory and time. While the temporal aspect of reasoning has been studied before, we believe that our treatment of the memory aspect is novel. We have proposed a new kind of epistemic logic where both resources are explicitly modelled. The logic is interpreted on state transition systems, where the agent's state can contain only a fixed finite number of formulas (beliefs), and transitions correspond to application of inference rules by the agent. By specifying the state transition system as an input to the MBP planner, we can automatically verify the lower bounds on space and time required by the agent to derive a certain formula.

In future work, we plan to model multi-agent systems, including agents which are capable of reasoning about other agents' beliefs and their reasoning in epistemic logic. We also would like to model reasoners in description logics, since verifying agents which do ontological reasoning is one of our prime motivations.

Acknowledgements This work was supported by the Royal Society UK-Italy Joint Project grant 'Model-checking resource-bounded agents'.

5. REFERENCES

[1] N. Alechina, P. Bertoli, C. Ghidini, M. Jago, B. Logan, and L. Serafini. Verifying space and time requirements for resource-bounded agents. Technical Report T05-10-03, ITC-irst, Trento, Italy, 2005.

[2] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. Weak, Strong, and Strong Cyclic Planning via Symbolic Model Checking. *Artificial Intelligence Journal*, 147(1,2):35–84, July 2003.

[3] A. Haken. The intractability of resolution. *Journal of Theoretical Computer Science*, 39(2-3):297–308, Aug. 1985.

[4] I. Horrocks and P. F. Patel-Schneider. A proposal for an OWL rules language. In *Proceedings of the 13th international conference on World Wide Web, WWW 2004*, pages 723–731. ACM, 2004.

The Interpreted System Model of Knowledge, Belief, Desire and Intention

Kaile Su[1], Xiangyu Luo[1], Abdul Sattar[1], Mehmet A Orgun[2]
[1]Institute for Integrated and Intelligent Systems, Grif th University, Brisbane, Australia
[2]Department of Computing, Macquarie University, Sydney, Australia
{k.su, x.luo, a.sattar}@grif th.edu.au; mehmet@comp.mq.edu.au

ABSTRACT

We present a new model of knowledge, belief, desire and intention, called the interpreted KBDI-system model (or KBDI-model for short). The key point of the interpreted KBDI-system model is that we express an agent's knowledge, belief, desire and intention as a set of runs (computing paths), which is exactly a *system* in the interpreted system model, a well-known agent model due to Halpern and his colleagues. Our KBDI-model is *computationally grounded* in that we are able to associate a KBDI-model with a computer program, and formulas, involving agents' knowledge, belief, desire (goal) and intention, can be understood as properties of program computations. We present a sound and complete proof system with respect to our KBDI-model and explore how symbolic model checking techniques can be applied to model checking multi-agent systems with KBDI-models.

1. INTRODUCTION

The *possible worlds semantics* [6] is a fruitful approach to formalizing agent systems via modal logics. The well-known theory of intention [3] and the formalism of the belief-desire-intention paradigm [10], for example, are along this line. However, it is still not very clear how to obtain concrete agent models with the belief, desire and intention accessibility relations from speci c agent programs.

The *interpreted system* model [4] offers a natural interpretation, in terms of the states of computer processes, to S5 epistemic logic. The salient point of the interpreted system model is that we are able to associate an interpreted system with a computer program, and formulas in epistemic logic that are valid with respect to the interpreted system can be understood as valid properties of program computations. In this sense, the interpreted system model is *computationally grounded* [14].

The aim of this paper is to present a computationally grounded model of knowledge, belief, desire and intention, called the interpreted KBDI-system model (or KBDI-*model* for short), by extending the interpreted system model. The key point of the KBDI-model is that an agent's beliefs, desires and intentions as well as its knowledge is characterized as a set of runs (computing paths), which is exactly a computationally grounded *system* in the interpreted system model.

Intuitively, a KBDI-model includes a system $\mathcal{K}$ as in the interpreted system model, as well as, for each agent i, its beliefs $\mathcal{B}_i$, desires $\mathcal{D}_i$ and intentions $\mathcal{I}_i$, which are subsystems or subsets of $\mathcal{K}$. Agent i's knowledge is determined by the set of those runs that are consistent with its local *observations* (or local state). Similarly, agent i's beliefs are de ned by the set of runs in $\mathcal{B}_i$ that are consistent with agent i's observations. Agent i's desires and intentions can also be de ned in the same way. Intuitively, runs in $\mathcal{B}_i$ are possible computing paths from the viewpoint of the agent and those in $\mathcal{D}_i$ are the computing paths that the agent desires. Thus, it is reasonable to assume that $\mathcal{D}_i \quad \mathcal{B}_i$ because every desired computing path should be possible. Nevertheless, we need not assume that $\mathcal{I}_i \quad \mathcal{D}_i$ or even $\mathcal{I}_i \quad \mathcal{B}_i$ because an agent's intention may fail to achieve its goal and the actual computing path may be beyond the agent's belief even though the agent has chosen and completed an intentional series of actions.

2. KBDI LOGIC

This section introduces a multimodal logic of knowledge, belief, desire and intention, referred to as Observation-based KBDI logic (KBDI). As shown below, the semantics of KBDI logic is given in terms of the interpreted KBDI-system model. According to this semantics, the computation of agents' knowledge, belief, desire, and intention is based on agents' *observations*, that is, local states.

2.1 Interpreted systems

Consider a system composed of multiple (say n) agents in an environment. We represent the system's state or the *global state* as a tuple $(s_e, s_1, \quad , s_n)$, where s_e is the *environment's local state* and, for $1 \quad i \quad n$, s_i is *agent i's local state*.

Let L_e be a set of possible local states of the environment and L_i a set of possible local states for agent i, for $i = 1, \quad , n$. We take $G \quad L_e \times L_1 \times \quad \times L_n$ to be the set of *reachable global* states of the system. A *run r* over G is a function from the time domain the natural numbers in our case to G. Thus, a run over G can be identi ed with a sequence of global states in G. A *point* is a pair (r, m) consisting of a run r and time m. Given a point (r, m), we denote the rst component of the tuple $r(m) = (s_e, s_1, \quad , s_n)$ by $r_e(m)$ $(= s_e)$ and, for each i $(1 \quad i \quad n)$, the $i+1$'th component of the tuple $r(m)$ by $r_i(m)$ $(= s_i)$. Thus, $r_i(m)$ is the local state of agent i in run r at time m.

The idea of the interpreted system semantics is that a run represents one possible computation of a system and a system may have

This work was supported by the Australian Research Council grant DP0452628, and partially by the NSFC grants 60496327, 10410638 and 60473004.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

a number of possible runs, so we say a *system* is a set of runs. Assume that we have a set Φ of primitive propositions, which we can think of as describing basic facts about the system. An *interpreted system* I consists of a pair $(\mathcal{R}, \ )$, where $\mathcal{R}$ is a set of runs over a set of global states and is a valuation function, which gives the set of primitive propositions true at each point in $\mathcal{R}$ [4].

For every agent i, let the notation $(r,u) \ _i \ (r',v)$ denote that $r_i(u) = r'_i(v)$. Intuitively, $(r,u) \ _i \ (r',v)$ means that (r,u) and (r',v) are indistinguishable to agent i. We also use the notation $(r,u) \ _i^{spr} \ (r',v)$ to denote that $u = v$ and, for every $j \ u$, $r_i(j) = r'_i(j)$ (here *spr* stands for *synchronous* systems with *perfect recall*).

Let KL_n denote the language of propositional logic augmented by the future-time connectives $\bigcirc$ (next) and $\mathbf{U}$ (until) and a modal (knowledge) operator K_i for each agent i. The language KL_n can be interpreted by using an interpreted system. The related satisfaction relationship $\models_{KL_n}$ is as follows: Given $I = (\mathcal{R}, \ )$ and a point (r,u) in $\mathcal{R}$, we de ne $(I,r,u) \models_{KL_n}$ by induction on the structure of . When is of the form $K_i\varphi$, $(I,r,u) \models_{KL_n}$ iff $(I,r',v) \models_{KL_n} \varphi$ for all (r',v) such that $(r,u) \ _i \ (r',v)$.

2.2 Interpreted KBDI Systems

Given a set G of global states and system $\mathcal{K}$ over G, an agent's *mental state* over system $\mathcal{K}$ is a tuple $\langle \mathcal{B}, \mathcal{D}, \mathcal{I} \rangle$, where $\mathcal{B}$, $\mathcal{D}$ and $\mathcal{I}$ are systems (sets of runs over G) such that $\mathcal{I} \ \mathcal{K}$ and $\mathcal{D} \ \mathcal{B} \ \mathcal{K}$. A *KBDI-system* is a structure $\langle \mathcal{K}, \mathcal{M}_1, \ , \mathcal{M}_n \rangle$, where $\mathcal{K}$ is a system and for every i, $\mathcal{M}_i$ is agent i's mental state over $\mathcal{K}$.

Assume that we have a set Φ of primitive propositions which describe basic facts about agents and their environment. An *interpreted KBDI-system* I consists of a pair $(\mathcal{S}, \ )$, where $\mathcal{S}$ is a KBDI-system and is a valuation function, which gives the set of primitive propositions true at each point in G.

2.3 Syntax

Given a set Φ of propositional atoms, the language of KBDI logic is de ned by the following BNF notations:

$$\begin{aligned} \langle wff \rangle ::= \ & \text{any element of } \Phi \mid \neg\langle wff \rangle \mid \langle wff \rangle \wedge \langle wff \rangle \mid \\ & \bigcirc\langle wff \rangle \mid \langle wff \rangle \mathbf{U} \langle wff \rangle \mid \\ & \mid K_i\langle wff \rangle \mid B_i\langle wff \rangle \mid D_i\langle wff \rangle \mid I_i\langle wff \rangle \end{aligned}$$

Informally, $K_i\varphi$, $B_i\varphi$ and $D_i\varphi$ means that agent i knows, believes and desires φ, respectively, while $I_i\varphi$ denotes that φ holds under the assumption that agent i acts based on his intention. The formulas not containing modalities K_i, B_i, D_i and I_i ($i = 1, \ , n$) are called linear-temporal logic (LTL) formulas.

2.4 Semantics

We now proceed to interpret KBDI logic formulas in terms of interpreted KBDI-systems. In the following, we inductively de ne the satisfaction relation $\models_O$ between a formula φ and a pair of interpreted KBDI-system and a point. Given an interpreted KBDI-system $I = (\mathcal{S}, \ )$, suppose that $\mathcal{S} = \langle \mathcal{K}, \mathcal{M}_1, \ , \mathcal{M}_n \rangle$ and for every i, $\mathcal{M}_i = \langle \mathcal{B}_i, \mathcal{D}_i, \mathcal{I}_i \rangle$. Let r be a run in $\mathcal{K}$ and u a natural number, then we have that:

- $(I,r,u) \models_O K_i\varphi$ iff $(I,r',v) \models_O \varphi$ for those (r',v) such that $r' \in \mathcal{K}$ and $(r,u) \sim_i (r',v)$;
- $(I,r,u) \models_O B_i\varphi$ iff $(I,r',v) \models_O \varphi$ for those (r',v) such that $r' \in \mathcal{B}_i$ and $(r,u) \sim_i (r',v)$;
- $(I,r,u) \models_O D_i\varphi$ iff $(I,r',v) \models_O \varphi$ for those (r',v) such that $r' \in \mathcal{D}_i$ and $(r,u) \sim_i (r',v)$;
- $(I,r,u) \models_O I_i\varphi$ iff $(I,r',v) \models_O \varphi$ for those (r',v) such that $r' \in \mathcal{I}_i$ and $(r,u) \sim_i (r',v)$;

The semantics of atomic formulas or formulas of the form $\neg\varphi$, $\varphi \wedge \varphi'$, $\bigcirc\varphi$ or $\varphi\mathbf{U}\varphi'$ can be dealt with in the usual manner.

We use $\models_O \varphi$ to denote that φ is valid in every interpreted KBDI-system.

According to our de nition, $D_i\varphi$ is true iff φ is true along those runs that are desirable to agent i and consistent with agent i's observations. Thus, $D_i\varphi$ intuitively means that agent i's goal implies that formula φ holds.

For those agents with perfect recall and a global clock, we may use $\ _i^{spr}$ instead of $\ _i$ to interpret those formals with modalities B_i, D_i and I_i and get an alternative satisfaction relationship $\models_O^{spr}$.

PROPOSITION 1. *The following axioms are valid with respect to both $\models_O$ and $\models_O^{spr}$:*

- $K_i\varphi \Rightarrow \varphi$
- $\Delta_i(\varphi \Rightarrow \psi) \Rightarrow (\Delta_i\varphi \Rightarrow \Delta_i\psi)$
  $\Delta_i\varphi \Rightarrow K_i\Delta_i\varphi \qquad \neg\Delta_i\varphi \Rightarrow K_i\neg\Delta_i\varphi$
  where Δ stands for K, B, D or I.
- *Relationship between knowledge, belief, desire and intention*
  $B_i\varphi \Rightarrow D_i\varphi \qquad K_i\varphi \Rightarrow B_i\varphi \qquad K_i\varphi \Rightarrow I_i\varphi$
- *Temporal operators*
  $\bigcirc(\varphi \Rightarrow \psi) \Rightarrow (\bigcirc\varphi \Rightarrow \bigcirc\psi)$
  $\bigcirc(\neg\varphi) \Rightarrow \neg \bigcirc \varphi$
  $\varphi\mathbf{U}\psi \Leftrightarrow \psi \vee (\varphi \wedge \bigcirc(\varphi\mathbf{U}\psi))$

PROPOSITION 2. *The following axioms are valid with respect to $\models_O^{spr}$: $\Delta_i \bigcirc \varphi \Rightarrow \bigcirc\Delta_i\varphi$, where Δ stands for K, B, D and I.*

The formula $D_i \bigcirc \varphi \Rightarrow \bigcirc D_i\varphi$ says that if agent i's current goal implies φ holds at the next point in time, then at the next point in time her goal will imply φ, that is, agent i persists on her goal.

2.5 Proof system

We now discuss a proof system, called the KBDI proof system, for those agents with perfect recall and a global clock. The proof system contains the axioms of propositional calculus plus those in Propositions 1 and 2. It is closed under the propositional inference rules plus for every agent i: $\frac{\vdash \varphi}{\vdash K_i\varphi}$.

THEOREM 3. *The KBDI proof system for agents with perfect recall and a global clock is sound and complete with respect to interpreted KBDI-systems with satisfaction relation $\models_O^{spr}$.*

3. MODEL CHECKING KBDI LOGIC

The model checking problem is to give an algorithm to test whether a model satis es a concerned speci cation expressed by a formula. However, in order to make our model checking algorithm practically useful, we must consider from where our model, an interpreted KBDI-system, comes. It would be satisfactory if we can derive our model from a program implemented in, say C or Java. However, to simplify the matter, we may consider some abstract programs such as *nite-state programs*, which are expressive enough from the standpoint of theoretical computer science. Moreover, to make our model checking system practically ef cient, we use symbolic model checking techniques. Thus, a nite-state program in our approach is represented in a symbolic way.

3.1 Symbolic Representation of Interpreted KBDI System Model

A system as a set of in nite runs is not well suited to model checking directly as it is generally applied to nite state systems. However, we can represent an interpreted system as a *nite-state program* (G, G_0, R, V), where G is a set of states, G_0 a set of initial states, and R a total next time relation, and V associates each state with a truth assignment function. A set of in nite runs is then obtained by unwinding the relation R starting from initial states in G_0.

We can present a symbolic representation of a nite-state program (G, G_0, R, V) as $(\mathbf{x}, \theta(\mathbf{x}), (\mathbf{x}, \mathbf{x}'))$. Also, we represent a system as a (symbolic) *nite-state program with n agents*, which is de ned as a tuple $\mathcal{P} = (\mathbf{x}, \theta(\mathbf{x}), (\mathbf{x}, \mathbf{x}'), O_1, \ , O_n)$, where

1. $\mathbf{x}$ is a set of system variables;
2. θ is a boolean formula over $\mathbf{x}$, called the *initial condition*;
3. is a boolean formula over $\mathbf{x} \cup \mathbf{x}'$, called the *transition relation*; and
4. for each i, O_i $\mathbf{x}$, containing agent i's *local variables*, or *observable variables*.

For convenience, we may use $\mathcal{P}(\theta, \ )$ to denote a nite-state program with n agents $(\mathbf{x}, \theta(\mathbf{x}), (\mathbf{x}, \mathbf{x}'), O_1, \ , O_n)$, if $\mathbf{x}$ and $O_1, \ , O_n$ are clear from the context.

Given a nite-state program $\mathcal{P}(\theta, \ )$ with n agents , we de ne an *agent's internal program* over $\mathcal{P}(\theta, \ )$ as a tuple

$$\langle \mathcal{P}(\theta_1, \ _1), \mathcal{P}(\theta_2, \ _2), \mathcal{P}(\theta_3, \ _3) \rangle$$

where, for $j = 1, 2, 3$, $(\theta_j \Rightarrow \theta) \wedge \ _j \Rightarrow$ is valid, and $(\theta_2 \Rightarrow \theta_1) \wedge \ _2 \Rightarrow \ _1$ is valid.

Clearly, an agent's internal program is exactly related to an agent's mental state. Thus, we de ne a (symbolic) *BDI-program with n agents* as a tuple $P_A = (\mathcal{P}_K, P_1, \ , P_n)$, where $\mathcal{P}_K$ is a nite-state program with n agents and for each agent i, P_i is agent i's internal program over $\mathcal{P}_K$. We use I_{P_A} to denote the corresponding interpreted KBDI-system .

3.2 Simpli ed BDI Programs

To obtain a BDI-program with n-agents, one should de ne $1 + 3n$ nite-state programs. Our experience with a previous version of **MCKBDI** indicated that it is inconvenient for a programmer to code a BDI-program with n-agents because there may be many nite-state programs involved. Therefore, we should simplify the de nition of a BDI-program with n-agents.

DEFINITION 4. *Let $\mathcal{P} = (\mathbf{x}, \theta(\mathbf{x}), (\mathbf{x}, \mathbf{x}'), O_1, \ , O_n)$ be a nite-state program with n-agents, where, for each agent i, O_i contains at least 3 particular boolean variables $\mathtt{WB}_i$, $\mathtt{WD}_i$ and $\mathtt{WI}_i$. A simpli ed BDI-program with n-agents derived from $\mathcal{P}$ is a tuple $(\mathcal{P}, P_1, \ , P_n)$, where for each agent i, $P_i = \langle \mathcal{P}(\theta_1^i, \ _1^i), \mathcal{P}(\theta_2^i, \ _2^i), \mathcal{P}(\theta_3^i, \ _3^i) \rangle$ such that*

1. $\theta_1^i = \theta \wedge \mathtt{WB}_i$ *and* $\ _1^i = \ \wedge \mathtt{WB}_i$;
2. $\theta_2^i = \theta \wedge \mathtt{WB}_i \wedge \mathtt{WD}_i$ *and* $\ _2^i = \ \wedge \mathtt{WB}_i \wedge \mathtt{WD}_i$;
3. $\theta_3^i = \theta \wedge \mathtt{WI}_i$ *and* $\ _3^i = \ \wedge \mathtt{WI}_i$.

It is more convenient for a programmer to code a simpli ed BDI-program with n-agents, and it seems that the notion of a simpli ed BDI-Program does not lose much generality of that of a BDI-Program.

4. CONCLUDING REMARKS

We have proposed a new KBDI-model by using interpreted systems, and developed a computationally grounded BDI logic, called KBDI logic. We interpret KBDI logic via two different semantics. One of them is based on the assumption that agents have perfect recall and there is a global clock, with respect to which a sound and complete proof system has been presented. We have explored how symbolic model checking techniques can be applied to model checking BDI-agents and obtained encouraging preliminary experimental results.

This paper is inspired by Su [11] and Su *et al* [12], which also provide agent models by extending the interpreted system model. However, Su [11] does not consider agents' belief, desire and intention. The main advantages of the present paper over Su *et al* [12] are: (1) We introduce, for each agent i, not only belief, desire and intention modalities B_i, D_i and I_i, but also a knowledge modality K_i. Moreover, the resulting axiomatic system is more succinct and proved to be sound and complete. (2) A model checker with KBDI-models, called **MCKBDI**, is developed.[1] Empirical results are more comprehensive, supported by convincing and motivating examples, where agents' observations and actions are explicitly de ned.

Model checking multi-agent systems has become an active research topic in the community of multi-agent systems. Many efforts have been devoted to model checking knowledge in multi-agent systems [5, 13, 11, 7]. However, comparatively little work has been carried out on model checking BDI-agents. Some general approaches to model checking BDI-agents were proposed in Rao and Georgeff [9] and Benerecetti and Cimatti [1], but no method was given for generating models from actual systems, and so the techniques given there could not easily be applied to verifying real multi-agent systems. In Bordini *et al* [2], model checking techniques for AgentSpeak(L) [8] have been reported; however, AgentSpeak(L) has very simple semantics. The salient point of our work is that we present a general form of a BDI agent program, from which BDI agent models are generated and speci cations in full BDI logics can be veri ed by symbolic model checking techniques.

5. REFERENCES

[1] M. Benerecetti and A. Cimatti. Symbolic model checking for multi-agent systems. In *Proc. MoChart-02*, pages 1–8, Lyon, France, 2002.

[2] R. Bordini, M. Fisher, C. Pardavila, and M. Wooldridge. Model checking agentspeak. In *Proc. AAMAS-03*, pages 14–18, Melbourne, Australia, 2003.

[3] P. Cohen and H. Levesque. Intension is choice with commitment. *Artificial Intelligence*, 42:23–261, 1990.

[4] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about knowledge*. MIT Press, Cambridge, MA, 1995.

[5] W. v. d. Hoek and M. Wooldridge. Model checking knowledge and time. In *9th Workshop on SPIN (Model Checking Software)*, 2002.

[6] S. Kripke. A semantical analysis of modal logic. i: Normal modal propositional calculi. *Z. Math. Logik Grundl. Math.*, 9:67–96, 1963.

[7] F. Raimondi and A. Lomuscio. Verification of multiagent system via ordered binary decision diagrams: an algorithm and its implementation. In *Proc. of AAMAS-04*, 2004.

[8] A. Rao. Bdi agent speak out in a logical computable language. In *LNAI*, volume 1038, pages 42–55, 1996.

[9] A. Rao and M. Georgeff. A model theoretic approach to the verification of situated reasoning systems. In *Proc. 13th International Joint Conference on Artificial Intelligence*, pages 318–324, 1993.

[10] A. Rao and M. Georgeff. Decision procedures for BDI logics. *Journal of Logic and Computation*, 8(3):293–344, 1998.

[11] K. Su. Model checking temporal logics of knowledge in distributed systems. In *AAAI-04*, pages 98–103. AAAI, 2004.

[12] K. Su, A. Sattar, K. Wang, and et al. The observation-based model for bdi-agents. In *AAAI-05*, pages 190–195. AAAI, 2005.

[13] R. van der Meyden and K. Su. Symbolic model checking the knowledge of the dining cryptographers. In *Proc. of IEEE CSFW-04*, pages 280–291, 2004.

[14] M. Wooldridge. Computationally grounded theories of agency. In E. Durfee, editor, *ICMAS-00*, pages 13–22. IEEE Press, 2000.

[1]The source codes and the user manual of **MCKBDI** are available at http://www.cit.gu.edu.au/ s360126/arc-mml/

Operationalisation of Norms for Usage in Electronic Institutions

Huib Aldewereld*
Frank Dignum
Institute of Information and Computing Sciences
Utrecht University
The Netherlands
{huib, dignum}@cs.uu.nl

Andrés García-Camino†
Pablo Noriega
Juan Antonio Rodríguez-Aguilar
Carles Sierra
Artificial Intelligence Research Institute, IIIA
Spanish Council for Scientific Research, CSIC
Campus de la UAB, Barcelona, Spain
{andres, pablo, jar, sierra}@iiia.csic.es

ABSTRACT

Agent-mediated electronic institutions belong to a new and promising field where interactions between a group of agents are regulated by means of a set of explicit norms. Current implementations of such open-agent systems are, however, mostly using constraints on the behaviour of the agents, thereby severely limiting the autonomy of the agents. To increase the autonomy of agents and possibly boost the efficiency of the overall system, a more flexible norm enforcement is required. However, as norms make extensive use of vague and ambiguous concepts and lack operational meaning (not expressing how the norm should be enforced), translating norms for usage with such a flexible enforcement mechanism might be difficult. In this paper we propose an extension to electronic institutions to allow for a flexible enforcement of norms, and manners to help overcome the difficulties of translating abstract norms.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems*

General Terms

Theory, Legal Aspects, Design

Keywords

Norms, Electronic Institutions, Normative Systems

*The first author of this paper was supported by the Netherlands Organisation for Scientific Research (NWO) under project number 634.000.017

†A. García-Camino was supported by the Spanish Council for Scientific Research (CSIC) by means of an I3P grant.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

Agent-mediated institutions, introduced in [12, 13], are open agent systems that allow agents to enter and perform tasks. Because of the heterogeneous nature of the agents joining an electronic institution (e-institution), measures have to be taken to control and regulate the behaviour of these agents. These measures are needed to improve and guarantee the safety and stability of the system, as agents joining the institution might, (un)intentionally, break the system by behaving in non-expected or non-accepted manners. It has been widely accepted that norms can be used for ensuring this safety, since norms, which are vague and abstract in order to express various, different circumstances without the need for change, can be used for defining the legality and illegality of actions (and states) in e-institutions [3].

For these norms to be used in e-institutions, thereby regulating the agents joining the institution, enforcement mechanisms must be devised to implement them.Although the norms should be available to the agents joining the institution to allow them to work more efficiently in the regulated domain, it is not the agents reasoning and trying to adhere to the norms that provides the safety that the institution needs; it is the institution itself that has to ensure that this safety exists. This enforcement of norms, as discussed in [14], comes down to either: 1) defining constraints on unwanted behaviour, or 2) detecting violations of norms and reacting to these violations. The first approach is simpler and used by current implementations of e-institutions such as [5]. However, it seriously reduces the autonomy of the agents. To allow the agents in e-institutions more freedom and flexibility, while still complying to the norms, we would like to extend the implementation of e-institutions with the second approach to enforcement.

Previous work on normative systems (mainly focussed on deontic frameworks [10]) is mostly *declarative in nature*, while the implementation of norms and norm enforcement in e-institutions, as mentioned above, requires norms to have an *operational semantics* as well. Where the declarative nature of norms is necessary for reasoning about norms (reasoning about what is and what is not accepted), the operational semantics define how norms are to be implemented (e.g. what to do when norms are violated). Recent approaches on normative systems have begun to research

and express this operational meaning of norms, as seen in [14, 1, 11, 7]. These approaches represent norms and their operational meaning, but are not conclusive on how the implementation in an agent system, such as an e-institution, should be obtained. In this paper we are trying to bridge this gap, by proposing a translation from the (operational) approach proposed in [14] to elements usable for norm enforcement in AMELI ([5]).

In this paper we assume institutions to be defined as a set of norms, which are to be enforced by a distributed set of (internal) agents. Secondly we assume that the norms can sometimes be violated by agents in order to keep their autonomy, which can also be functional for the system as a whole as argued in [2]. The violation of norms is handled from the organisational point of view by violation and sanction mechanisms. And finally, we assume that the internal state of agents is neither visible, nor controllable from an institution's point of view, which, basically, means that enforcement of norms can only use the observable behaviour of agents to detect the violations.

2. IMPLEMENTING NORMS

E-institutions, as viewed here [4, 12, 13], are agent environments that restrict the behaviour of agents to ensure that agents interact in safe conditions. E-institutions constrain agent behaviour by defining the valid sequences of (dialogical) interactions that agents can have to attain their goals.

In order to be able to handle violations of norms we need a manner to express the violations that can occur in an institution, for which we introduce *integrity constraints* (derived from [6]):

DEFINITION 1. *Integrity constraints are first-order formulas of the form*

$$\left(\bigwedge_{i=1}^{n} uttered(s_i, w_{k_i}, \boldsymbol{i}_{l_i}) \wedge \bigwedge_{j=0}^{m} e_j\right) \rightarrow \bot$$

where s_i are scene identifiers or variables, w_{k_i} is a state k_i of scene s_i or a variable, $\boldsymbol{i}_{l_i}$ is an illocution scheme l_i matching the schema labelling an outgoing arc from w_{k_i} and e_j are boolean expressions over variables from uttered predicates.

Integrity constrains define sets of states that *should not* occur within an e-institution, i.e. if grounded illocutions matching the illocution schemes $\boldsymbol{i}_{l_1}, \ldots, \boldsymbol{i}_{l_n}$ are uttered in the corresponding scene states, and expressions $e_1, \ldots, e_m$ are satisfied, then a violation occurs ($\bot$).

This, however, is not enough to implement norms, as we need to define the reactions to the violations of the norms as well. These reactions (normally seen as sanctions and repairs, [14]) are expressed by using *dialogical constraints*:

DEFINITION 2. *Dialogical constraints are first-order formulas of the form:*

$$\left(\bigwedge_{i=1}^{n} uttered(s_i, w_{k_i}, \boldsymbol{i}^*_{l_i}) \wedge \bigwedge_{j=0}^{m} e_j\right) \Rightarrow \left(\bigwedge_{i=1}^{n'} uttered(s'_i, w'_{k_i}, \boldsymbol{i}'^*_{l_i}) \wedge \bigwedge_{j=0}^{m'} e_j\right)$$

where s_i, s'_i are scene identifiers or variables, w_{k_i}, w'_{k_i} are variables or states of scenes s_i and s'_i respectively, $\boldsymbol{i}^*_{l_i}$, $\boldsymbol{i}'^*_{l_i}$ are illocution schemes l_i matching the schema labelling an outgoing arc from w_{k_i} of scenes s_i and s'_i respectively, and e_j, e'_j are boolean expressions over variables from uttered predicates. These boolean expressions can include functions to check the state of the institution.

The intuitive meaning of a dialogical constraint is that, if the violation of the norm has been detected (expressed in the left-hand side of the constraint), the enforcers in the e-institution become obliged to perform a specific pattern of illocutions to punish the violator (this sanction is expressed in the right-hand side of the constraint).

The integrity constraints are then implemented in the infrastructure of the e-institutions, thereby providing the means to detect violations of norms, where the dialogical constraints are implemented in the enforcing agents which use them to determine the illocutions that should be uttered when a norm has been violated (thereby punishing the agent that violated the norm).

Semantics of the mentioned constraints, and an implementation for the AMELI platform can be given, but is omitted here due to space limitations.

3. CONTEXTUALISING NORMS

In the previous section we described how norm enforcement could be implemented in AMELI. However, the description of the norms should be given completely in terms of the utterances used in the e-Institution. However, usually norms are described in more abstract terms and need first to be "contextualised". The following norm, expressed in the formalism developed in [14]:

OBLIGED($(buyer$ DO $pay(Price,seller))$ IF $done(buyer,won(Item,Price)))$

can be translated into the utterances that would be used in an e-institution:

$uttered(payment,W,inform(A,buyer,B,payee,pay(Item,Price),T))$
$uttered(auction,w2,inform(C,auctioneer,A,buyer,won(Item,Price),T'))$

However, if we regard institutional norms that are derived (or translated) from human laws and regulations, the contextualisation becomes much harder, as laws contain vague and ambiguous concepts and cannot always be related to a single integrity constraint. In order to implement such norms with a high level of abstraction two steps must be taken: 1) interpreting the abstract concepts and link them to concrete concepts used in the institution (by using a *counts as* operator as described in [8, 9]), and 2) adding procedural information and artifacts to the institution to simplify (or allow) the enforcement of the norm.

The operational approach to norms developed in [14] tries to implement norms from an institutional perspective. It views norms as a way to describe how someone should behave. The norm frame used in this approach contains the *norm condition*, which defines obligations, permissions and prohibitions also known as the *declarative meaning of norms* (cf. [10]), of which an example was given above. However, since a system needs responses to the violations that occur, the norm frame also includes a definition of the responses to violations of the norms, which are known as sanctions and repairs (also known as the *operational meaning of the norm*). Finally, the norm frame includes the *violation condition* (obtained from the norm condition to express when a violation occurs) and the *detection mechanisms* (to describe the available resources on the platform that can be used for detecting violations).

Norms expressed in this formalism can be translated to

the constraints mentioned earlier. The integrity constraint is directly derivable from the violation condition of the formalism of [14]. Dialogical constraints are then formed by combining the violation condition and responses of the system to violations. For instance, if a norm with sanction S and repairs R (which can be obtained from the norm framework), can be translated to the integrity constraint $IC \rightarrow \bot$, the following rule can be created (automatically) to oblige the enforcers to utter the illocution (DC) that is derived from contextualising S and R:

$$IC \Rightarrow DC$$

Since these dialogical constraints are considered obligations to the enforcers, we can specify another integrity constraint to detect that this obligation has been violated, which is when the original norm was violated, but the enforcer did not punish the violating agent.

$$(IC \wedge \sim DC) \rightarrow \bot$$

4. CONCLUSIONS

With the development of electronic institutions, with the aim of implementing normative open-agent systems, comes the problem of ensuring the safety and stability of the system. Previous implementations of electronic institutions enforced norms by ensuring that the agents joining the system followed a pre-defined protocol, thereby guaranteeing norm compliance of the agents. As this approach severely limits the autonomy of the agents, a more flexible enforcement was desired. This paper proposes the use of integrity constraints and dialogical constraints to implement such a flexible enforcement of norms. This norm enforcement is based on the detection of and reacting to violations of norms.

In order for any kind of norm enforcement to be implemented, norms need to be expanded with an operational meaning, as the declarative nature of norms only defines what is legal/illegal, but never expresses how this legality/illegality is obtained/averted. In [14] we introduced several mechanisms for operationalising norms, where we annotated norms (expressed in deontic logic) with operational aspects, like sanctions and repairs. In this paper we have used this normative frame to design an implementation scheme usable for implementing norm enforcement in electronic institutions. However, before norms can be implemented using this scheme, the norms need to be contextualised. This contextualisation amounts to 1) connecting the abstract concepts of the norm to the concrete concepts used in the institution, and 2) extending the norm with additional procedural information before attempting to implement it. The contextualisation of the norms is, in fact, a further operationalisation of the norms, where, in contrast to declarative norms (which never change the world), the second step of this operationalisation changes the world in order to enforce the norm.

Acknowledgements

This paper was partially supported by the Spanish Science and Technology Ministry as part of the Web-i-2 project (TIC-2003-08763-C02-00).

5. REFERENCES

[1] A. Artikis, L. Kamara, J. Pitt, and M. Sergot. A protocol for resource sharing in norm-governed ad hoc networks. In *Proc. of the Declarative Agent Languages and Technologies (DALT) workshop*. Springer, 2004.

[2] C. Castelfranchi. Formalizing the informal?: Dynamic social order, bottom-up social control, and spontaneous normative relations. *Journal of Applied Logic*, 1(1-2):47–92, February 2003.

[3] F. Dignum. Abstract norms and electronic institutions. In *Proc. of the Int. Workshop on Regulated Agent-Based Social Systems: Theories and Applications (RASTA '02), Bologna*, pages 93–104, 2002.

[4] M. Esteva. *Electronic Institutions: from specification to development*. Number 19 in IIIA Monograph Series. PhD Thesis, 2003.

[5] M. Esteva, J. Rodríguez-Aguilar, B. Rosell, and J. Arcos. AMELI: An Agent-based Middleware for Electronic Institutions. In *3rd Int. Joint Conf. on Autonomous Agents and Multi-agent Systems*, New York, US, July 2004.

[6] M. Esteva, W. Vasconcelos, C. Sierra, and J. Rodríguez-Aguilar. Verifying norm consistency in electronic institutions. In *Proceedings of The AAAI-04 Workshop on Agent Organizations: Theory and Practice (ATOP)*, San Jose, California, July 2004.

[7] A. García-Camino and J. Rodríguez-Aguillar. Implementing norms in electronic institutions. In *Proc. of the 4th Int. Joint Conf. on Autonomous Agents & Multi Agent Systems (AAMAS-05)*, Utrecht, The Netherlands, July 2005.

[8] D. Grossi, H. Aldewereld, J. Vázquez-Salceda, and F. Dignum. Ontological aspects of the implementation of norms in agent-based electronic institutions. *Computational and Mathematical Organization Theory*, to appear in 2006.

[9] D. Grossi, F. Dignum, and J.-J. Ch. Meyer. Contextual taxonomies. In J. Leite and P. Toroni, editors, *Proceedings of CLIMA V Workshop, Lisbon, September*, pages 2–17, 2004.

[10] A. Lomuscio and D. Nute, editors. *Proc. of the 7th Int. Workshop on Deontic Logic in Computer Science (DEON04)*, volume 3065 of *LNCS*. Springer Verlag, 2004.

[11] F. López y López and M. Luck. Towards a Model of the Dynamics of Normative Multi-Agent Systems. In G. Lindemann, D. Moldt, M. Paolucci, and B. Yu, editors, *Proc. of the Int. Workshop on Regulated Agent-Based Social Systems: Theories and Applications (RASTA '02)*, volume 318 of *Mitteilung*, pages 175–194. Universität Hamburg, 2002.

[12] P. Noriega. *Agent-Mediated Auctions: The Fishmarket Metaphor*. Number 8 in IIIA Monograph Series. PhD Thesis, 1997.

[13] J. A. Rodriguez-Aguilar. *On the Design and Construction of Agent-mediated Electronic Institutions*. Number 14 in IIIA Monograph Series. PhD Thesis, 2001.

[14] J. Vázquez-Salceda, H. Aldewereld, and F. Dignum. Norms in multiagent systems: From theory to practice. *International Journal of Computer Systems Science & Engineering*, 20(4):pages 225–236, 2005.

Mutual Enrichment Through Nested Belief Change

Laurent Perrussel
IRIT–Université Toulouse 1
Manufacture des Tabacs, 21
allée de Brienne, F-31042
Toulouse Cedex - France
laurent.perrussel@irit.fr

Jean-Marc Thévenin
IRIT–Université Toulouse 1
Manufacture des Tabacs, 21
allée de Brienne, F-31042
Toulouse Cedex - France
thevenin@univ-tlse1.fr

Thomas Meyer*
NICTA and UNSW
223 Anzac Parade
Kensington, NSW, 2052,
Australia
thomas.meyer@nicta.com.au

ABSTRACT

We investigate the dynamics of nested beliefs in the context of agent interactions. Nested beliefs represent what agents believe about the beliefs of other agents. We consider the *tell* KQML performative which allows agents to send their own beliefs to others. Whenever agents accept a new belief, or refuse to change their own beliefs after receiving a message, both receiver and sender enrich their nested beliefs by refining their beliefs about the other agent's beliefs, as well as the preferences of the other agent. The main objective of nested beliefs is to improve cooperation between agents. We propose a logical framework for the acquisition process of nested beliefs and preferences. This acquisition process is the first step toward the elaboration of sophisticated interaction protocols. In this short paper we provide an informal outline of the framework, guided by an intuitive running example.

Keywords

multi-agent systems, reasoning about action, belief revision

1. INTRODUCTION

In a multi-agent context *nested beliefs* represent what agents believe about the beliefs of other agents. Nested beliefs improve interaction and cooperation among agents [5, 2, 10, 14]. Few papers deal with the problem of how agents acquire nested beliefs [9, 4]. None of them considers this in the context of cooperative dialogues. We propose an acquisition process for nested beliefs which describes how agents construct nested beliefs as they receive information.This provides the foundation on which we can build sophisticated dialogues or cooperation protocols.

*National ICT Australia is funded by the Australia Government's Department of Communications, Information and Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Centre of Excellence program. It is supported by its members the Australian National University, University of NSW, ACT Government, NSW Government and affiliate partner University of Sydney.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

We consider multi-agent dialogues based on the *tell* KQML performative [6] which enables an agent to send a message to inform a receiver agent that a statement holds in its own belief base. We suppose that the sender provides the *source* of this statement [12]. The receiver may accept or ignore the incoming statement leading to non-prioritised revision of basic beliefs [8]. To perform revision agents need to be able to define preferences over their basic beliefs and thus the incoming statement may be considered as less preferred than conflicting basic beliefs. Preferences over basic beliefs is obtained by maintaining preferences over agents at the origin of basic beliefs, representing the reliability of the statement origin.

Based on this we define the acquisition of nested beliefs for the receiver of a *tell* performative. Regardless of the acceptance of the *tell* performative, the receiver is informed that the sender believes the incoming statement. After a *tell* performative the receiver performs non-prioritised revision on its nested beliefs about the sender. Since the sender provides the source of its beliefs, the receiver can deduce the nested preferences of the sender. The sender can also enrich its nested beliefs in several ways. For this, we introduce two performatives *accept* and *deny* which enable the receiver agent to inform the sender whether or not it actually changed its own beliefs after a *tell*. Using these performatives the sender can enrich its nested beliefs in several ways. After an *accept* the sender believes that the receiver believes in the content of the *tell*. If, during the revision process, an inconsistency occurs in the nested beliefs of the receiver, the sender can refine its knowledge about the preferences of the receiver. This is also the case after a *deny* statement.

To illustrate the issues involved, consider the following example. Agent Peter receiving messages from agent Paul and the police department. Suppose Peter considers Paul to be less reliable than the police department. At time 0, Paul tells Peter that *John is a murderer* (origin Paul). At time 1 Peter accepts the message. At time 2, Paul tells Peter that *if John is a murderer, John will go to jail* (origin Paul). Peter accepts the message and consequently Peter believes that *John will go to jail* at time 3. In addition, Peter believes that Paul believes both statements. After the second accept, Paul also believes that Peter believes both statements. At time 4, the police department tells Peter that *John is not a murderer* (origin Police). Because of its preferences, Peter accepts message at time 5 and now believes *John is not a murderer*. In addition, Peter believes that the police department believes *John is not a murderer*. After the accept message, the police believes that Peter believes *John is not a murderer*. Because it believes that Paul still believes that *John is a murderer*, at time 6, Peter tells Paul that *John is not a murderer* and the origin of this information is the police. Whether or not Paul accepts Peter's message, Paul has to change its nested beliefs about Peter. Paul now believes that Peter believes *John is not a murderer* and that Peter considers the police to be more reliable than himself.

The paper is structured as follows. Section 2 sketches a logical framework for reasoning about nested beliefs, preferences and the performatives *tell*, *accept*, and *deny*. Sections 3 and 4 give an overview of the mutual enrichment for the receiver and the sender. We conclude with a discussion on related, and future work.

2. THE LOGICAL FRAMEWORK

The *belief state* of an agent contains basic beliefs as well as nested beliefs. The basic beliefs of an agent a, denoted by B_a, are represented as signed statements, each consisting of a propositional sentence and the agent id of the source of the statement. Every agent a also maintains a set of signed statements for each of the other agents, representing what each of them currently believes, according to a (the nested beliefs of a). $B_{a,b}$ denotes what agent b believes according to agent a. We also assume that agents have preferences, arranged as a total preorder, over the set of agents A which describe the reliability of the sources of information [3, 12]. So $b \preceq_a c$ means: agent c is at least as preferred as b for agent a ($b \prec_a c$ and $b =_a c$ have the obvious meanings). Agents can also handle nested preferences. $c \preceq_{a,b} d$ means: agent a believes that for agent b agent d is at least as preferred as c. Together, these constitute the preference state of the agent.

We first present the language $\mathcal{L}_1$ for reasoning about basic beliefs and preferences. $\mathcal{L}_1$ is a restricted first order language which enables quantification over agent ids. It has a modal-like operator Bel for basic beliefs. $\mathsf{Bel}(a, \phi_0, t)$ means agent a believes ϕ_0 signed by t. We limit agent beliefs to propositional sentences. To represent preferences we introduce the notation $b \leqslant_a c$ which stands for: agent a considers agent c to be at least as preferred as b. We extend $\mathcal{L}_1$ to the language $\mathcal{L}_1^K$ to deal with nested beliefs. Nested beliefs are restricted to $\mathcal{L}_1$-formulae. We introduce the operator B to describe nested beliefs. Intuitively, $\mathsf{B}(a, \mathsf{Bel}(b, \phi_0, s))$ means agent a believes that agent b believes ϕ_0 signed by s, while $\mathsf{B}(a, c \leqslant_b d)$ means that agent a believes that agent b considers agent d to be at least as preferred as c. So, in our example, at time 3 agent $peter$ has the following belief state:

$$B_{peter} = \{\langle murd, paul\rangle, \langle murd \rightarrow jail, paul\rangle\}$$
$$B_{peter,paul} = \{\langle murd, paul\rangle, \langle murd \rightarrow jail, paul\rangle\}$$
$$B_{peter,police} = \emptyset$$

Considering that $peter$ does not know $paul$ and $police$'s preferences at the beginning, the preference state of $peter$ is:

$$paul \prec_{peter} peter \prec_{peter} police$$
$$paul =_{peter,paul} peter =_{peter,paul} police$$
$$paul =_{peter,police} peter =_{peter,police} police$$

We now formalise the communication between agents. We assume that agents exchange information about their basic beliefs. We represent interactions between agents using three performatives. $\texttt{Tell}(s, r, \phi_0, t)$ stands for: agent s informs r that it believes ϕ_0 signed by t according to the standard KQML semantics [6]. When receiving a `Tell` performative, agent r performs non-prioritised revision according to its preferences [8]. $\texttt{Accept}(r, s, p)$ stands for: agent r informs s that it accepts the `Tell` performative p. Let $p = \texttt{Tell}(s, r, \phi_0, t)$. Then $\texttt{Accept}(r, s, p)$ means that agent r has revised its basic beliefs by $\langle \phi_0, t\rangle$ and thus believes ϕ_0. $\texttt{Deny}(r, s, p)$ stands for: agent r informs s that it refuses to process the `Tell` performative p. If performative $p = \texttt{Tell}(s, r, \phi_0, t)$ it means that agent r has not revised its basic beliefs by $\langle \phi_0, t\rangle$.

The dynamics of the system is given by performatives. At each step one agent receives a performative and may change its state accordingly, i.e. performatives are viewed as transition functions. We extend $\mathcal{L}_1^K$ to a language $\mathcal{L}_2$ to handle performative actions by considering a modal-like action operator $[p]\phi$, which states that after performative action p, ϕ is true. With $\mathcal{L}_2$ we can specify the behaviour of agents when they interact. For instance, we can express the honesty postulate as follows.

$$[\texttt{Tell}(s, r, \phi_0, t)]\top \rightarrow \mathsf{Bel}(s, \phi_0, t) \qquad \textbf{(Hon)}$$

This postulate states that if a `Tell` performative occurs, then at the same time the sender believes the corresponding signed statement. Postulates such as (Hon) place constraints on the logical framework, ensuring that it behaves as required. Due to lack of space we shall, in the rest of the paper, only provide informal descriptions of the postulates required for our framework. The first such postulate, (KeNB1), requires that the nested beliefs and preferences of agents not involve in a dialogue do not change, and that the dialogue between the sender and the receiver is a two step process: first a `Tell` and then an `Accept` or a `Deny`.

3. DYNAMICS OF THE RECEIVER

In this section, we discuss the dynamics of the nested beliefs and nested preferences of agent r for the $\texttt{Tell}(s, r, \phi_0, t)$ performative, and then consider the acknowledgement sent by agent r to agent s. The nested beliefs and preferences of all agents except s have to remain unchanged. (KeNB2) propagates them to the next state and ensures that no new nested beliefs or preferences appear. Also, agent r changes its beliefs about s, whether or not it ignores the input statement. Because agents are supposed to be honest, agent r performs a standard prioritised revision of its nested beliefs about s by signed statement $\langle \phi_0, t\rangle$. (AdNB1) states that after the `Tell` performative r believes that s believes ϕ_0 and thus does not believe $\neg\phi_0$ anymore (because belief states are consistent). (ProNB1) states that all nested beliefs which hold after the `Tell` performative also hold at the previous state or are consequences of the new nested belief $\mathsf{B}(r, \mathsf{Bel}(s, \phi_0, t))$. (AdNB1) and (ProNB1) describe the expected behaviour of a revision operator for the nested beliefs but do not specify which nested beliefs have to be removed to facilitate the removal of $\neg\phi_0$ from the nested beliefs of r, since we do not insist on a specific method for revising nested beliefs.

(KeNP1) states that nested preferences about all agents except s are unchanged: it propagates them to the next state and ensures that no new preference appears. If agent r currently believes that s believes $\neg\phi_0$ signed by t' then it refines the s preferences as follows. According to the honesty principle and (AT) (see below), t is better than t'. (AdNBP1) formalises this intuition. (KeNP2) requires that the nested preferences of agent s about all agents except t are unchanged. But if r does not believe that s believes $\neg\phi_0$ then it cannot draw conclusion about agent s preferences. Agent r can refine preferences only if it faces inconsistencies during the revision process. (KeNP3) states that if r does not believe that s believes $\neg\phi_0$ then all nested preferences about s are unchanged.

This bring us to the receiver's acknowledgement. Whenever agent r receives $\texttt{Tell}(s, r, \phi_0, t)$, r determines if its beliefs are consistent with ϕ_0, and if not, which beliefs have to be dropped by taking into account its preference relation. Due to the non-prioritised aspect the receiver may deny or accept the `Tell` performative. If agent r believes $\neg\phi_0$ and the signatures of $\neg\phi_0$ are at least as preferred as t (the signature of ϕ_0), postulate (DT) requires that agent r deny the `Tell` performative. Otherwise there is no conflict or t is strictly more preferred than all the signatures of $\neg\phi_0$, and postulate (AT) requires that r accept the `Tell` performative.

4. DYNAMICS OF THE SENDER

In this section we describe the dynamics of the sender's nested beliefs and preferences with respect to the performatives `Accept` and `Deny`. (KeNB3) states that the nested beliefs of agents not involved in the `Accept` or `Deny` performative do not change. Next, we focus on the nested beliefs about r. According to (AT), if r accepts the message of s, agent s believes that r believes ϕ_0 and thus does not believe $\neg\phi_0$, as specified by (AdNB3). According to (DT), if r refuses the `Tell` performative then s concludes that r believes $\neg\phi_0$, as specified by (AdNB2). Like (ProNB1), postulates (ProNB2) and (ProNB3) state that all nested beliefs holding at the next state also hold at the current state, or are consequences of the new nested belief $\mathsf{B}(r, \mathsf{Bel}(s, \phi_0, t))$. So in both cases, s nested beliefs about r undergoes a prioritised revision: by $\langle \phi_0, t\rangle$ in the case of `Accept`, and by $\langle \neg\phi_0, t\rangle$ in the case of `Deny`.

This bring us to the sender's nested preferences. (KeNP4) states that the sender's nested preferences about agents other than r do not change. Next, agent s draws conclusions about t depending on whether r accepts or refuses a `Tell` performative. Whenever agent r accepts, agent s refines its nested preferences only if s currently believes that r believes $\neg\phi_0$. In that context s refines agent r's preferences. (AdNP2) requires that t be strictly more preferred than the signatures of $\neg\phi_0$ if r accepts the performative $\texttt{Tell}(s, r, \phi_0, t)$. (AdNP3) requires that if r refuses the message, the nested preferences of s about the signatures of $\neg\phi_0$ change. According to condtion (DT) they have to be as preferred as t. Finally, (KeNP5) and (KeNP6) deal with those nested preferences, about r, of s that do not change. They are similar to (KeNP2) and (KeNP3).

We now conclude our example. At time 4, the $police$ tells $peter$ that *John is not a murderer*, and $peter$ accepts it.

$$p_4 = \texttt{Tell}(police, peter, \neg murd, police)$$
$$p_5 = \texttt{Accept}(peter, police, p_4)$$

By (AdNB1) and (AdNB3) we get the new nested beliefs:

$$[p_4]\mathsf{B}(peter, \mathsf{Bel}(police, \neg murd, police))$$
$$[p_5]\mathsf{B}(police, \mathsf{Bel}(peter, \neg murd, police)))$$

At time 1 $peter$ accepted $paul$'s message that *John is a murderer* ($murd$) which gives

$$\mathsf{B}(paul, \mathsf{Bel}(peter, murd, paul) \wedge paul =_{peter} police)$$

according to (KeNB1). At time 6, this nested belief is still true according to the accepted behaviour of nested beliefs given by (KeNB1), (AdNB3), (ProNB2) and (KeNP6). At time 6 $peter$ tells $paul$ that *John is not a murderer*: $p_6 = \texttt{Tell}(peter, paul, \neg murd, police)$.

According to (AdNB1) and (AdNP2) we obtain the following new nested beliefs and preferences for $paul$:

$$[p_6]\mathsf{B}(paul, \mathsf{Bel}(peter, \neg murd, police))$$
$$[p_6]\mathsf{B}(paul, paul <_{peter} police)$$

So, at stage 6 $paul$ has refined its nested preferences about $peter$. At the beginning of the dialogue $paul$ does not know whether $peter$ prefers $paul$ or the $police$. At stage 6, $paul$ believes that $peter$ prefers the police to himself.

5. CONCLUSION

We have outlined a framework for handling the dynamics of nested beliefs and preferences in the context of agent interactions. We proposed a logical framework which allows for the expression of nested beliefs, preferences, and performatives, and showed how agents acquire nested beliefs and preferences. For this we have specified the expected behaviour of agents handling the `Tell`, `Accept` and `Deny` performatives.

Related work includes that of [4], which proposes a logic-based definition for nested belief change and mental state construction related to agent-communication languages. Our proposal differs from theirs in that it deals with preferences of agents. Our work is also closely related to [13, 12] where it is shown how to embed basic belief change in a logical system. We extend this by considering nested beliefs and preferences. Finally, our work is loosely related to the dynamics of preferences [1]. Our contribution is to formulate this dynamics in terms of agent interactions. We plan to extend our framework by considering nested *disbeliefs* [7]. Explicit disbeliefs allow for a more nuanced type of interaction between agents and provide new opportunities for the discovery of nested beliefs and preferences. We also plan to revisit persuasion protocols [11] so that agents can generate smarter arguments.

6. REFERENCES

[1] G. Brewka. Well-Founded Semantics for Extended Logics Programs with Dynamic Preferences. *Journal of Artificial Intelligence Research*, (4):19–36, 1996.

[2] P. Cohen and H. Levesque. Rational Interaction as the Basis for Communication. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 221–256. MIT Press, 1990.

[3] A. Dragoni and P. Giorgini. Revising beliefs received from multiple sources. In M. Williams and H. Rott, editors, *Frontiers of Belief Revision, Applied Logic*. Kluwer, 1999.

[4] A. Dragoni, P. Giorgini, and L. Serafini. Mental States Recognition from Communications. *J. Logic Computat.*, 12(1):119–136, 2002.

[5] A. C. F, Grasso and R. Jones. Dialectical argumentation to solve conflicts in advice giving: a case study in the promotion of healthy nutrition. *International Journal of Human-Computer Studies*, 53(1077–1115), 2000.

[6] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. Bradshaw, editor, *Software Agents*. MIT Press, 1997.

[7] A. Ghose and R. Goebel. Belief states as default theories: studies in non-prioritised belief change. In H. Prade, editor, *ECAI'98*, pages 8–12, 1998.

[8] S. O. Hansson. A survey of non-prioritized belief revision. *Erkenntnis*, 50:413–427, 1999.

[9] A. Herzig and D. Longin. Belief dynamics in cooperative dialogues. *J. of Semantics*, 17(2), 2000. vol. published in 2001.

[10] D. Kinny, M. Ljungberg, A. S. Rao, E. Sonenberg, G. Tidhar, and E. Werner. Planned team activity. In C. Castelfranchi and E. Werner, editors, *Artificial Social Systems – MAAMAW'92*, volume 830 of *LNAI*, pages 226–256. Springer-Verlag, 1992.

[11] S. Parsons, M. Wooldridge, and L. Amgoud. An analysis of formal inter-agent dialogues. In M. Gini, T. Ishida, C. Castelfranchi, and W. L. Johnson, editors, *Proceedings of AAMAS'02*, pages 394–401. ACM Press, July 2002.

[12] L. Perrussel and J. Thévenin. A logical approach for describing (dis)belief change and message processing. In *AAMAS'04*, pages 614–621. IEEE C.S., 2004.

[13] J. Roorda, W. van der Hoek, and J. Meyer. Iterated Belief Change in MAS. In C. Castelfranchi and W. Johnson, editors, *AAMAS'02*, pages 889–896. ACM Press, July 2002.

[14] D. Sperber and D. Wilson. *Relevance*. Harvard University Press, 1986.

An Architecture of a Normative System

Counts-as Conditionals, Obligations and Permissions

Guido Boella
Università di Torino
Italy
guido@di.unito.it

Leendert van der Torre
University of Luxembourg
Luxembourg
leendert@vandertorre.com

ABSTRACT

Normative systems are traditionally described and analyzed using deontic logic, describing the logical relations among obligations and permissions. However, there is still a gap between deontic logic and normative multi-agent systems such as electronic institutions, which may be seen as an instance of the gap between on the one hand logical agent specification languages and on the other hand agent architectures and programming languages. To bridge the gap, in this paper we propose an architecture containing separate subsystems or components for counts-as conditionals, conditional obligations and conditional permissions. We add a norm database component in which the three kinds of rules are stored, and we use a channel based coordination model to describe the relations among the four normative components.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multi-agent systems

General Terms

MAS theory

Keywords

Normative systems, normative multi-agent systems

1. NORMATIVE ONTOLOGY

Searle [13, 14] distinguishes two types of norms: "Some rules regulate antecedently existing forms of behavior. For example, the rules of polite table behavior regulate eating, but eating exists independently of these rules. Some rules, on the other hand, do not merely regulate an antecedently existing activity called playing chess; they, as it were, create the possibility of or define that activity. The activity of playing chess is constituted by action in accordance with these rules. Chess has no existence apart from these rules. The institutions of marriage, money, and promising are like the institutions of baseball and chess in that they are systems of such constitutive rules or conventions" [13, p. 131].

Thus far the two kinds of norms have mainly been studied in isolation, which raises the question how regulative and constitutive norms are related. In the deontic logic literature, the interaction among obligations and permissions has been studied in some depth, see, e.g., [16, 7, 12]. In most formalizations, obligations, prohibitions and permissions have a conditional nature. Their conditions could directly refer to entities and facts of the commonsense world, but they often refer to a legal and more abstract classification of the world, making them more independent from the commonsense view. E.g., they refer to money instead of paper sheets, to properties instead of houses and fields. This more natural and economical way to model the relation between commonsense reality and legal reality uses constitutive norms, and allows regulative norms to refer to the legal classification of reality. In this way, it is not necessary that each regulative norm refers to all the conditions involved in the classification of paper as money or of houses and fields as properties. A constitutive norm has been defined by Searle as "X counts as Y in context C", and has been formalized as a counts-as conditional [10, 3, 9, 4].

Since all kinds of norms are represented by conditionals, we can represent them as subsystems within the normative system, or as input/output components in an architecture. This raises the question what kind of input and output these components have. According to Searle, institutional facts like marriage, money and private property emerge from an independent ontology of "brute" natural facts through constitutive norms of the form "such and such an X counts as Y in context C" where X is any object satisfying certain conditions and Y is a label that qualifies X as being something of an entirely new sort. Thus, the propositions describing the world are distinguished in two categories. First, "brute facts" are natural facts and events produced by actions of agents. Second, "institutional facts" are a legal classification of brute facts; they belong only to the beliefs of the normative system and have no direct counterpart in the world. Moreover, counts-as conditionals can be iterated, in the sense that X counts-as Y and Y counts-as Z (to construct the complexity of social reality), and therefore we also allow institutional literals in X. We call a set of brute literals a brute description, a set of institutional literals an institutional description, and a set of both brute and institutional literals a mixed description.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

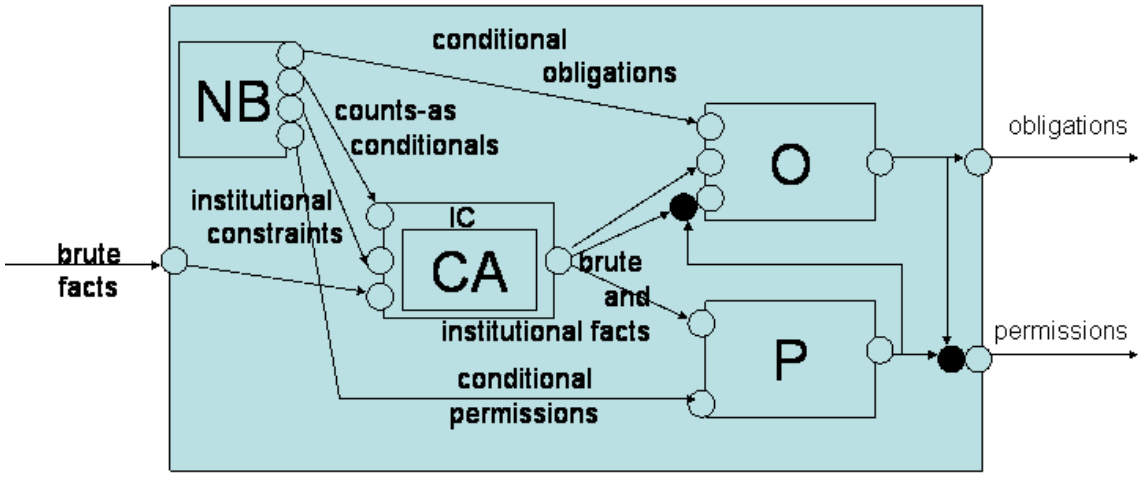


Figure 1: The Architecture of the Normative System.

2. NORMATIVE COMPONENTS

We follow the definition of normative systems proposed by Alchourrón and Bulygin [1], inspired by Tarski's definition of deductive systems. They consider as input factual descriptions and as output a classification of obligatory and permitted situations. Permissions are not closed under logical consequence, in the sense that a permission for p and a permission for $\neg p$ do not imply a permission for a contradiction. For technical reasons, we describe the permissions by sets of mixed descriptions. In our architecture, visualized in Figure 1, the input of the normative system is a context, a brute description, and the output is a set of obligations, a mixed description, and a set of permissions, a set of mixed descriptions.

There are four components in the architecture. Besides the ones for counts-as conditionals (CA), obligations (O) and permissions (P), there is a component that contains the norms (NB). The counts-as component has a wrapper (IC) to enable the connection with the other components. The components are connected by channels. The data flowing over the channels is not indexed by modal operators. So if a proposition p is flowing over a permission channel it is interpreted as a permission, usually represented in deontic logic by $P(p)$, and when it is flowing over an obligation channel it is interpreted as an obligation, usually represented by $O(p)$.

The norm database does not have any input, we therefore assume that the norms of the normative system are fixed. E.g., it can be extended with an input to update the norm database. We also assume that there can be background knowledge in the form of institutional constraints. The output contains sets of conditional obligations, conditional permissions, counts-as conditionals, and institutional constraints, four sets of pairs of mixed descriptions.

The counts-as component directly encodes Searle's conditional "X counts as Y in context C". Examples of constitutive norms are "X counts as a presiding official in a wedding ceremony", "this bit of paper counts as a five euro bill" and "this piece of land counts as somebody's private property". The input contains counts-as conditionals, X, a brute description, and C, a mixed description. The output is Y, an institutional literal.

The counts-as component is part of a component for institutional constraints, acting like a wrapper for the counts-as conditionals. Its input are the brute facts, the counts-as conditionals, and the institutional constraints. The output are the brute and institutional facts, a mixed description.

The notorious contrary-to-duty paradoxes like Chisholm's and Forrester's paradox have led to the use of constraints in deontic logic [11]. The strategy is to adapt a technique that is well known in the logic of belief change - cut back the set of norms to just below the threshold of making the current situation inconsistent. Therefore, the input of the obligation component contains conditional obligations, a set of pairs of mixed descriptions, a context, a mixed description, and constraints, a set of mixed descriptions. The output contains obligations, a mixed description.

The permission component is like an obligation component without constraints, since the issue of contrary-to-duty does not play a role for permissions. The input of the permission component is a set of conditional permissions, and a context of brute and institutional facts. As mentioned above in the description of the whole normative system, another distinction is that the output is a set of mixed descriptions. The input contains conditional permissions, a set of pairs of mixed descriptions, and a context, a mixed description. The output contains permissions, a set of mixed descriptions.

Finally, consider again the whole architecture. When designing a normative system, there is a tradeoff between obligations, permissions and counts-as conditionals, in the sense that there are many norm bases which have the same global output. For example, the same output can be obtained for example using many counts-as conditionals, or no counts-as conditionals at all. Likewise there is a tradeoff between obligations and permissions. Consequently, there is a need for a methodology to represent normative systems in our architecture. It is relatively straightforward to represent a piece of legal code in the architecture, as in legal code the three kinds of norms are usually clearly distinguished, but in general it is more problematic.

3. COORDINATION

A coordination model defines the interaction among components. In software engineering coordination models and languages based on tuplespaces are very popular, but such a coordination model does not reflect the topology of the architecture. Therefore, in this paper the coordination of the normative components is based on a channel-based coordination model. Our discussion is influenced by the coordination language Reo [2], which has also been used to coordinate agents and agent organizations [8].

Most of the channels are straightforward. The output of the norm database is connected to the input of the three other components. The output of the wrapper of the counts-as component is the context of the obligation and permission component, *et cetera*. All channels are synchronous, and we assume that they are all synchronized. This can be made explicit in the architecture. For example, in the Reo coordination language this can be achieved by connecting so-called SyncDrain channels between all channels. However, we did not represent this in the architecture to keep it as simple as possible. This synchronicity may seem like an unrealistic assumption, but it simply means that the calculation is atomic. This is a common way to model coordination in coordination languages. In actual implementations they do not have to be synchronous, but there is a mechanism in the coordination language to ensure the atomicity.

When there are multiple output channels on the same port, then the output is replicated to all channels. For example, the output of the wrapper of the counts-as component is replicated to the obligation and permission component, and the output of the permission component is replicated to the output of the normative system as well as to an input of the obligation component. When there are multiple input channels which are joined, then there are special merge nodes (a simple kind of component). In Reo there is one predefined merge as a non-deterministic choice among the inputs, if inputs are available at the same time. In the normative system architecture there are two mergers, represented by black circles. One merges the output of the counts-as component with the output of the permission component to result in the constraints for the obligation component, and the other merges the output of the obligation and permission component such that obligations are permitted. They cannot be modeled as a non-deterministic choice, but they are defined by $d \times \{d_1, \ldots, d_n\} = \{d \cup d_1, \ldots, d \cup d_n\}$. Since there are no cycles, it is straightforward to calculate the coordination.

The relation between the counts-as component and its wrapper is as follows. When the counts-as component outputs at least Y for input X and C, then the wrapper also outputs Y (and possibly more) when the input contains both X and C.

4. LOGICAL SPECIFICATIONS

The architecture is a bridge between logical specifications, for example based on deontic logic, and normative multi-agent systems. In a normative multi-agent system the components can be implemented in a variety of ways. The abstract behavior of the components and channels can be given in terms of real-timed timed data streams [2], based on general results in the Reo coordination language. The principles of deontic logic can be interpreted as functionality descriptions of the components [15, 6].

Though initially our motivation to develop the architecture of the normative system was to build such computer systems, we noticed that the architecture can be used too in deontic logic research. In particular, it can be used as a logical architecture to study interaction among logical systems by varying the logics to describe the components. For example, we can use either the logic for counts-as conditionals of Jones and Sergot [10], Artosi *et al.* [3], Grossi *et al.* [9], or our own proposal [4], and study its interaction with input/output logic [11], or some other deontic logic. To do so, we just have to find the flat conditional fragment of the logics, and ignore the modal operator. This is subject of further study [5].

5. REFERENCES

[1] C. Alchourrón and E. Bulygin. *Normative systems*. Springer, 1971.

[2] F. Arbab. Reo: A channel-based coordination model for component composition. *Mathematical Structures in Computer Science*, 14(3):329–366, 2004.

[3] A. Artosi, A. Rotolo, and S. Vida. On the logical nature of count-as conditionals. In *Procs. of LEA 2004 Workshop*, 2004.

[4] G. Boella and L. van der Torre. Regulative and constitutive norms in normative multiagent systems. In *Procs. of KR'04*, pages 255–265. AAAI Press, 2004.

[5] G. Boella and L. van der Torre. A logical architecture of a normative system. In *Procs. of ΔEON'06*, LNAI. Springer, 2006.

[6] J. Broersen, M. Dastani, and L. van der Torre. Beliefs, obligations, intension and desires as components in agent architectures. *International Journal of Intelligent Systems*, 20:9:893–919, 2005.

[7] E. Bulygin. Permissive norms and normative systems. In A. Martino and F. S. Natali, editors, *Automated Analysis of Legal Texts*, pages 211–218. Publishing Company, Amsterdam, 1986.

[8] M. Dastani, F. Arbab, and F. de Boer. Coordination and composition in multi-agent systems. In *Procs. of AAMAS05*, pages 439–446. 2005.

[9] D. Grossi, F. Dignum, and J.-J. Meyer. Contextual taxonomies. In *Procs. of CLIMA-V*, LNAI 3487, pages 33–51. Springer, 2005.

[10] A. Jones and M. Sergot. A formal characterisation of institutionalised power. *Journal of IGPL*, 3:427–443, 1996.

[11] D. Makinson and L. van der Torre. Constraints for input-output logics. *Journal of Philosophical Logic*, 30:155–185, 2001.

[12] D. Makinson and L. van der Torre. Permissions from an input/output perspective. *Journal of Philosophical Logic*, 32 (4):391–416, 2003.

[13] J. Searle. *Speech acts: An essay in the philosophy of language*. Cambridge University, Cambridge, England, 1969.

[14] J. Searle. *The Construction of Social Reality*. The Free Press, New York, 1995.

[15] J. Treur. Semantic formalisation of interactive reasoning functionality. *International Journal of Intelligent Systems*, 17:645–686, 2002.

[16] G. von Wright. Deontic logic. *Mind*, 60:1–15, 1951.

What Agents Can Achieve under Incomplete Information

Wojciech Jamroga
Institute of Computer Science
Clausthal University of Technology, Germany
wjamroga@in.tu-clausthal.de

Thomas Ågotnes*
Department of Informatics
University of Bergen, Norway
agotnes@ii.uib.no

ABSTRACT

We propose a non-standard semantics for Alternating-time Temporal Logic (ATL) with incomplete information, for which no commonly accepted semantics has been proposed yet. In our semantics, formulae are still interpreted with respect to ATL structures, but are interpreted in sets of states rather than in single states. We also propose a new epistemic operator for "constructive" knowledge, and we show that the new language is strictly more expressive than existing solutions, while retaining the same model checking complexity.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*; I.2.4 [**Artificial Intelligence**]: Knowledge Repr. Formalisms and Methods—*Modal logic*

General Terms

Theory

Keywords

Alternating-time Temporal Logic, strategic ability, incomplete information, epistemic logic

1. INTRODUCTION

ATL [1] is probably the most important logic of strategic ability that has emerged in recent years. A combination of ATL and epistemic logic, called ATEL, was introduced in [9] to enable reasoning about agents acting under incomplete information. Still, it has been pointed out in several places that the meaning of ATEL formulae is somewhat counterintuitive. A number of ATEL revisions were proposed to overcome this problem [3, 5, 8, 6, 10, 2], yet none of them seems to be the ultimate definitive solution. Our aim is to come up with a logic of ability under incomplete information which is both general and elegant.

*Supported by Norw. Research Council grant 166525/V30

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

In this paper, we propose a non-standard semantics for the logic of strategic ability and incomplete information. In the semantics, formulae are interpreted over *sets of states* rather than single states. This reflects the intuition that the "constructive" ability to enforce φ means that the agents in question have a single strategy that brings about φ for *all* possible initial situations – and not that a successful strategy exists for *each* initial situation (because those could be different strategies for different actual situations). To do it in a flexible and general way, the type of the satisfaction relation in our proposal forces one to specify the set of initial states explicitly. In consequence, we write $M, Q \models \langle\langle A \rangle\rangle \varphi$ to express the fact that A must have a strategy which is successful for all states in a set Q. It should be emphasized that we do not propose new *models*: concurrent epistemic game structures have already been used for several years in ATEL-like logics. What we propose is a new *interpretation of formulae*. We also propose a new epistemic operator for "practical" or "constructive" knowledge that yields the set of states for which a single evidence (i.e., a successful strategy) should be presented (instead of checking if the required property holds in each of the states separately, like standard epistemic operators do). The new semantics and operators are presented in Section 3.

2. WHAT AGENTS CAN ACHIEVE

In this section, we present a very brief overview of ATL and its extensions for agents with incomplete information.

Alternating-time Temporal Logic. ATL [1] can be understood as a generalization of the branching time temporal logic CTL, in which path quantifiers are replaced with so called *cooperation modalities*. Formula $\langle\langle A \rangle\rangle \varphi$, where A is a coalition of agents, expresses that A have a collective strategy to enforce φ. ATL formulae include temporal operators: "$\bigcirc$" ("in the next state"), $\Box$ ("always from now on") and $\mathcal{U}$ ("until"). The semantics can be defined using *concurrent game structures*, each including: agents $\mathbb{A}\mathrm{gt} = \{1, \ldots, k\}$; states St; actions Act; atomic propositions Π; valuation $\pi : St \rightarrow \mathcal{P}(\Pi)$; function $d : \mathbb{A}\mathrm{gt} \times St \rightarrow \mathcal{P}(Act)$ defining actions available to an agent in a state; and a transition function o assigning an outcome state $q' = o(q, \alpha_1, \ldots, \alpha_k)$ to state q and tuple of actions $\langle \alpha_1, \ldots, \alpha_k \rangle$ that can be executed by $\mathbb{A}\mathrm{gt}$ in q. A *strategy* $s_a : St \rightarrow Act$ is a conditional plan that specifies what $a \in \mathbb{A}\mathrm{gt}$ is going to do in every possible situation. A *collective strategy* S_A is a tuple of strategies, one per agent from $A \subseteq \mathbb{A}\mathrm{gt}$. A *path* Λ in model M is an infinite sequence of states that can be effected by subsequent transitions. Function $out(q, S_A)$ returns the set of all

paths that may result from agents A executing strategy S_A from state q onward. Informally speaking, $M, q \models \langle\langle A\rangle\rangle\varphi$ iff there is a collective strategy S_A such that φ holds for every $\Lambda \in out(q, S_A)$.

Alternating-time Temporal Epistemic Logic. ATEL [9] adds to ATL operators for representing agents' knowledge: $K_a\varphi$ reads "agent a knows that φ". Additional operators $E_A\varphi$, $C_A\varphi$, and $D_A\varphi$ refer to "everybody knows", common knowledge, and distributed knowledge among the agents from A. Models for ATEL extend concurrent game structures with indistinguishability relations $\sim_1, ..., \sim_k \subseteq Q \times Q$ (one per agent) for modeling agents' uncertainty. Then: $M, q \models K_a\varphi$ iff φ holds for every q' such that $q \sim_a q'$.

Relations $\sim_A^E$, $\sim_A^C$ and $\sim_A^D$, used to model group epistemics, are derived from the individual relations of agents from A. First, $\sim_A^E$ is the union of relations $\sim_a$, $a \in A$. Next, $\sim_A^C$ is defined as the transitive closure of $\sim_A^E$. Finally, $\sim_A^D$ is the intersection of all the $\sim_a$, $a \in A$. Then, for $\mathcal{K} = C, E, D$: $M, q \models \mathcal{K}_A\varphi$ iff φ holds for every q' such that $q \sim_A^{\mathcal{K}} q'$.

Problems with ATEL. It has been pointed out in several places that the meaning of ATEL formulae is somewhat counterintuitive [3, 5, 6]. Most importantly, one would expect that an agent's ability to achieve φ should imply that the agent has enough control and knowledge to *identify* and *execute* a strategy that enforces φ. This problem is closely related to the distinction between knowledge *de re* and knowledge *de dicto*, well known in the philosophy of language [7]. Several variations on "ATL with incomplete information" have been proposed, yet none of them seems definitive. We summarize the most important proposals below.

ATL$_{ir}$. In the logic of ATL$_{ir}$ [8], cooperation modalities are presented with a subscript: $\langle\langle A\rangle\rangle_{ir}$ to indicate that they address agents with *imperfect information* and *imperfect recall*. Agents are required to use *uniform* strategies, i.e. ones that specify the same choices in indistinguishable states (if $q \sim_a q'$ then $s_a(q) = s_a(q')$). Formula $\langle\langle A\rangle\rangle_{ir}\varphi$ holds in M, q iff there is a uniform collective strategy S_A such that, for every $a \in A$, q' such that $q \sim_a q'$, and path $\Lambda \in out(q', S_A)$, we have that φ is true for Λ. In other words, there is a strategy such that *everybody in A knows* that executing this strategy will bring about φ. Note that it is not possible to express that A have common knowledge about the successful strategy, or that they can identify it if they share their knowledge, etc.

Alternating-time Observational Temporal Logic. ATOL, proposed independently in [5], follows the same perspective as ATL$_{ir}$. However, it includes also epistemic modalities in the object language (like ATEL), and it offers a richer language of strategic operators to express subtle differences between various kinds of collective abilities. The reading of $\langle\langle A\rangle\rangle_{\mathcal{K}(\Gamma)}\varphi$ is: "group A has a (memoryless, uniform) strategy to enforce φ, and agents Γ can identify the strategy as successful for A in the epistemic sense $\mathcal{K}$". That is, $M, q \models \langle\langle A\rangle\rangle_{\mathcal{K}(\Gamma)}\varphi$ iff there is a S_A such that for every q' such that $q \sim_\Gamma^{\mathcal{K}} q'$ and path $\Lambda \in out(q', S_A)$, φ is true for Λ. We observe that model checking ATL$_{ir}$ and ATOL is NP-complete in the size of the model and the formula [8, 4, 5].

"Feasible ATEL*".* The revision of ATEL from [6] extends ATEL with new modalities: $\langle\langle A\rangle\rangle^f$, $\langle\langle A\rangle\rangle^f_E$, $\langle\langle A\rangle\rangle^f_C$, $\langle\langle A\rangle\rangle^f_{K_a}$ and $\langle\langle A\rangle\rangle^f_{M_a}$, very similar to the ones of ATOL. The NP-completeness result carries over to "Feasible ATEL" (it subsumes ATL$_{ir}$ and can be seen as a subset of ATOL).

Other Approaches. Epistemic Temporal Strategic Logic [10] focuses on the concept of *undominated strategies*; in a way, $\langle\langle A\rangle\rangle\varphi$ in ETSL can be summarized as: "if A play *rationally* to achieve φ (meaning: they never play a dominated strategy), they will achieve φ". Another, very recent proposal [2] approaches the problem of strategic abilities within the framework of STIT (the logic of *seeing to it that*). We do not discuss these proposals further here due to lack of space.

In the original formulation of ATL, agents were assumed to have perfect recall of the game, in the sense that they could base their decisions on *sequences* of states rather than single states. Variants of ATL for perfect recall and incomplete information include ATL$_{iR}$ [8] and ATEL-R* [5]. However, as agents seldom have unlimited memory, and logics of strategic ability with incomplete information and perfect recall are believed to have undecidable model checking [1, 8], we do not investigate this kind of ability here.

3. A NEW SEMANTICS FOR ABILITY AND KNOWLEDGE, AND NEW EPISTEMIC OPERATORS

ATOL covers more cases than ATL$_{ir}$ and "Feasible ATEL", and it is not committed to any notion of rationality (unlike ETSL). One major drawback of ATOL is that it vastly increases the number of modal operators necessary to express properties of agents. For team A, a whole family of cooperation modalities $\langle\langle A\rangle\rangle_{\mathcal{K}(\Gamma)}$ is used to specify who should identify the right strategy for A, in what way etc. It would be much more elegant to modify the semantics of "simple" cooperation modalities $\langle\langle A\rangle\rangle$ and/or epistemic operators, so that they can be composed into sufficiently expressive formulae. However, the property of a strategy being successful (under incomplete information) with respect to goal φ is *not* local to the current state; *the same* strategy must be successful in all possible "opening" states. In order to capture this feature of strategic ability, we change the type of the satisfaction relation $\models$, and define what it means for a formula φ to be satisfied in a *set of states* $Q \subseteq St$ of model M. To our best knowledge, nobody has used this kind of semantics yet. Moreover, we extend the language of ATEL with unary "constructive knowledge" operators $\mathbb{K}_a$, one for each agent a, that yield the set of states, indistinguishable from the current state from a's perspective. Constructive common, "everybody's" and distributed knowledge is formalized via operators $\mathbb{C}_A, \mathbb{E}_A$, and $\mathbb{D}_A$.

3.1 Language and Semantics

The language is defined formally as follows, where $p \in \Pi$:

$$\varphi ::= p \mid \neg\varphi \mid\mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle \bigcirc \varphi \mid \langle\langle A\rangle\rangle \Box\varphi \mid \langle\langle A\rangle\rangle \varphi \,\mathcal{U}\, \varphi \mid C_A\varphi \mid E_A\varphi \mid D_A\varphi \mid \mathbb{C}_A\varphi \mid \mathbb{E}_A\varphi \mid \mathbb{D}_A\varphi.$$

The models are again concurrent epistemic game structures, and we consider only memoryless uniform strategies. Now, we define the notion of a formula φ being satisfied by a set of states Q in a model M, written $M, Q \models \varphi$. We will also write $M, q \models \varphi$ as a shorthand for $M, \{q\} \models \varphi$. Let $\mathrm{img}(q, \mathcal{R})$ be the set of all states q' such that $q\mathcal{R}q'$. Moreover, we write $out(Q, S_A)$ for $\cup_{q\in Q} out(q, S_A)$, and $\mathrm{img}(Q, \mathcal{R})$ for $\cup_{q\in Q}\mathrm{img}(q, \mathcal{R})$. The new semantics is given through the

clauses below. Individual knowledge operators can be derived as: $K_a\varphi \equiv C_{\{a\}}\varphi$ and $\mathbb{K}_a\varphi \equiv \mathbb{C}_{\{a\}}\varphi$.

$M, Q \models p$ iff $p \in \pi(q)$ for every $q \in Q$, when $p \in \Pi$;
$M, Q \models \neg\varphi$ iff $M, Q \not\models \varphi$;
$M, Q \models \varphi \wedge \psi$ iff $M, Q \models \varphi$ and $M, Q \models \psi$;
$M, Q \models \langle\langle A\rangle\rangle \bigcirc \varphi$ iff there exists S_A such that, for every $\Lambda \in out(Q, S_A)$, we have that $M, \{\Lambda[1]\} \models \varphi$;[1]
$M, Q \models \langle\langle A\rangle\rangle \Box\varphi$ iff there exists S_A such that, for every $\Lambda \in out(Q, S_A)$ and $i \geq 0$, we have $M, \{\Lambda[i]\} \models \varphi$;
$M, Q \models \langle\langle A\rangle\rangle \varphi\, \mathcal{U}\, \psi$ iff there exists S_A such that, for every $\Lambda \in out(Q, S_A)$, there is an $i \geq 0$ for which $M, \{\Lambda[i]\} \models \psi$ and $M, \{\Lambda[j]\} \models \varphi$ for every $0 \leq j < i$;
$M, Q \models \mathcal{K}_A\varphi$ iff $M, q \models \varphi$ for every $q \in \mathrm{img}(Q, \sim_A^{\mathcal{K}})$,
$M, Q \models \hat{\mathcal{K}}_A\varphi$ iff $M, \mathrm{img}(Q, \sim_A^{\mathcal{K}}) \models \varphi$
(where $\hat{\mathcal{K}} = \mathbb{C}, \mathbb{E}, \mathbb{D}$ and $\mathcal{K} = C, E, D$, respectively).

3.2 Expressing Agents' Strategic Abilities

$M, q \models \mathbb{K}_a\langle\langle a\rangle\rangle\varphi$ expresses the fact that a has a single strategy that enforces φ from *all* states indiscernible from q, instead of stating that φ can be achieved from *every* such state *separately*. Note that the latter property is very much in the spirit of standard epistemic logic, and indeed can be captured with the standard knowledge operator (as $K_a\langle\langle a\rangle\rangle\varphi$). More generally, the first kind of formulae refers to *having a strategy "de re"* (i.e. having a successful strategy and knowing the strategy), while the latter refers to *having a strategy "de dicto"* (i.e. only knowing that *some* successful strategy is available; cf. [5]). Note also that the property of having a winning strategy for the current state (but not necessarily even knowing *about* it) is simply expressed with $\langle\langle a\rangle\rangle\varphi$. Capturing different ability levels of *coalitions* is analogous, with various "epistemic modes" of collective recognizing the right strategy.

THEOREM 1. *Let φ be a formula of* ATL$_{ir}$, ATOL *or "Feasible* ATEL*", and let tr be as follows:*

$$tr(p) = p \qquad tr(\neg\varphi) = \neg tr(\varphi)$$
$$tr(\varphi \wedge \psi) = tr(\varphi) \wedge tr(\psi) \qquad tr(\bigcirc\varphi) = \bigcirc tr(\varphi)$$
$$tr(\Box\varphi) = \Box tr(\varphi) \qquad tr(\varphi\, \mathcal{U}\, \psi) = tr(\varphi)\, \mathcal{U}\, tr(\psi)$$
$$tr(\langle\langle A\rangle\rangle_{ir}\varphi) = \mathbb{E}_A\langle\langle A\rangle\rangle\varphi \qquad tr(\langle\langle A\rangle\rangle_{\mathcal{K}(\Gamma)}\varphi) = \hat{\mathcal{K}}_\Gamma\langle\langle A\rangle\rangle\varphi$$
$$tr(\langle\langle A\rangle\rangle^f\varphi) = \langle\langle A\rangle\rangle\varphi \qquad tr(\langle\langle A\rangle\rangle^f_{\mathcal{K}}\varphi) = \hat{\mathcal{K}}_A\langle\langle A\rangle\rangle\varphi$$
$$tr(\langle\langle A\rangle\rangle^f_{K_b}\varphi) = \mathbb{K}_b\langle\langle A\rangle\rangle\varphi \qquad tr(\langle\langle A\rangle\rangle^f_{M_b}\varphi) = \neg K_b\neg\langle\langle A\rangle\rangle\varphi$$
$$tr(\mathcal{K}_A\varphi) = \mathcal{K}_A tr(\varphi)$$

where $\mathcal{K} = C, E, D$ and $\hat{\mathcal{K}} = \mathbb{C}, \mathbb{E}, \mathbb{D}$, respectively. Then:

$$M, q \models \varphi \text{ iff } M, q \models tr(\varphi).$$

REMARK 2. *The new language is strictly more expressive than* ATL$_{ir}$, ATOL *etc.: for example, formula $\mathbb{E}_A\mathbb{E}_A\langle\langle A\rangle\rangle\varphi$ cannot be expressed in any of the former logics.*

3.3 Model Checking

We define *general model checking* as the problem that asks whether formula φ holds in model M and *set of states* Q. Let $mctl(\varphi, M)$ be a CTL model checker that returns the set of all states that satisfy φ in M. Below, we sketch algorithm $mcheck(\varphi, M, Q)$ that returns "yes" if $M, Q \models \varphi$ and "no" otherwise, running in nondeterministic polynomial time.

- Cases $\varphi \equiv p, \varphi \equiv \neg\psi, \varphi \equiv \psi_1 \wedge \psi_2, \varphi \equiv \mathcal{K}_A\psi$: straightforward (proceed as usually).
- Case $\varphi \equiv \hat{\mathcal{K}}_A\psi$: return $mcheck(\psi, M, \mathrm{img}(Q, \sim_A^{\mathcal{K}}))$.
- Case $\varphi \equiv \langle\langle A\rangle\rangle \bigcirc \psi$: run $mcheck(\psi, M, q)$ for every $q \in St$, and label the states in which the answer was "yes" with a new proposition p. Then, guess the strategy of A, and "trim" model M by removing all the transitions inconsistent with the strategy (yielding a sparser model M'). Return "yes" iff $Q \subseteq mctl(\mathsf{A}\bigcirc\mathsf{p}, M)$. [For other temporal operators: analogous.]

Note that all the relevant strategies can be guessed *beforehand*, as a single complex (but still polynomial) witness (cf. [4]), which gives us the following result:

THEOREM 3. *General model checking for our logic is NP-complete in the size of the model and the formula.*

4. CONCLUSIONS

In this paper, we propose a non-standard semantics for the modal logic of strategic ability under incomplete information, in which formulae are interpreted over *sets of states* rather than single states. Moreover, we introduce new epistemic operators for "constructive" knowledge. It turns out that, in this new semantics, simple cooperation modalities $\langle\langle A\rangle\rangle$ can be combined with "constructive" epistemic operators into sufficiently expressive formulae. Indeed, the new logic *is strictly more expressive than most existing* ATL *versions for incomplete information*, while it retains the same model checking complexity as the least costly of them. The philosophical dimension of constructive knowledge is also natural: the constructive knowledge operators capture the notion of *knowing "de re"*, while the standard epistemic operators refer to *knowing "de dicto"*. We believe that we have finally obtained a satisfying logic of agents' strategies under uncertainty, and at the same time have come up with novel, meaningful epistemic operators that capture important properties of the interaction between knowledge, action and ability.

5. REFERENCES

[1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.
[2] A. Herzig and N. Troquard. Knowing how to play: Uniform choices in logics of agency. *AAMAS*, 2006.
[3] W. Jamroga. Some remarks on alternating temporal epistemic logic. In *Proc. of FAMAS*, p. 133–140, 2003.
[4] W. Jamroga and J. Dix. Model checking strategic abilities of agents under incomplete information. In *Proceedings of ICTCS 2005*, pages 295–308, 2005.
[5] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63(2–3), 2004.
[6] G. Jonker. Feasible strategies in Alternating-time Temporal Epistemic Logic. Master thesis,2003.
[7] W. Quine. Quantifiers and propositional attitudes. *Journal of Philosophy*, 53:177–187, 1956.
[8] P. Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2), 2004.
[9] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *Proceedings of AAMAS-02*, pages 1167–1174, 2002.
[10] S. van Otterloo and G. Jonker. On Epistemic Temporal Strategic Logic. In *Proc. of LCMAS*, 2004.

[1] By $\Lambda[i]$, we denote the ith position in Λ (starting from 0).

Flexible Agent Protocols via Temporal and Resource-based Reasoning

Duc Q. Pham
RMIT University
GPO Box 2476V
Melbourne, 3001, Australia
qupham@cs.rmit.edu.au

James Harland
RMIT University
GPO Box 2476V
Melbourne, 3001, Australia
jah@cs.rmit.edu.au

ABSTRACT

In multi-agent systems, it has long been recognized that the traditional approaches to protocols which often used a fixed sequence of interactive actions as the basis are too brittle, and more flexible methods are needed. A crucial aspect of the development of such methods is the notion of commitments, which provides a mechanism for regulating interactions between agents. In this paper we investigate a rule-based approach to the specification of flexible protocols by means of a reasoning system which incorporates both temporal reasoning and resources-based reasoning and uses the notion of commitments.

Categories and Subject Descriptors

D.3.1 [**Programming Languages**]: Formal Definitions and Theory; I.2.5 [**Artificial Intelligence**]: Programming Languages and Software; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Representations (procedural and rule-based)*

General Terms

Languages, Theory

Keywords

Agent programming languages, Formal models of agency, Logics for agent systems

1. INTRODUCTION AND BACKGROUND

In traditional approaches to protocol specification, like those using Finite State Machines or Petri Nets in distributed computing or computer networks, protocols are often predetermined legal sequences of interactive behaviors. In frequently changing environments such as the Internet, such fixed sequences can quickly become outdated and are prone to failure. Hence, protocols in such environments need to be flexible, and in particular, interaction protocols should ensure that agents have autonomy over their interactive behaviors. Also, agents should be allowed to adjust their interactive actions to take advantages of opportunities or handle exceptions that arise during interaction.

Our approach is to specify protocols in a declarative manner, i.e. in terms of what is to be achieved rather then how agents should interact. The concept of *commitments* is used to allow agents to perform means-end reasoning to determine an appropriate course of action. We do this reasoning in *temporal linear logic* [3], which combines temporal and resource-sensitive reasoning. It is well-known that temporal logic, in particular, is quite suitable for describing and reasoning about temporal constraints, and has been used in many agent systems. It is also known that linear logic is suitable for modeling resources [1]. Hence we used temporal linear logic to construct a commitment based interaction framework which allows both temporal and resource-related reasoning for interaction protocols.

A commitment refers to a strong promise to undertake some courses of action. Persistence in commitments gives agents a certain level of predictability about the actions of other agents, which is fundamental for issues of interdependencies, global constraints or resource sharing [4]. Commitments have been adapted to make protocols more flexible via commitment machines [6]. Conventions are also used to provide monitoring mechanisms for commitments.

In linear logic, formulas are constrained to be used exactly once and hence treated as resources. The connective $\otimes$ also allows a natural expression of proportion. If A encodes "having one dollar" then $A \otimes A \neq A$ and means having two dollars. Moreover, as remarked in [2], connectives $\&$ and $\oplus$ allow choices to be made clear between internal choices (one's own), and external choices (others' choice). For instance, to specify that the choice of places A or B for goods' delivery is ours as the supplier, we use $A\&B$, or is the client's, we use $A \oplus B$. Linear logic is a natural mechanism to provide the ability to match consumption and supply of resources among agents and hence can simplify the specification of resource allocation [2].

Temporal Linear Logic (**TLL**) [3] is the result of introducing temporal logic into linear logic and hence is resource-conscious as well as deals with time. The temporal operators used are $\bigcirc$ (next), $\Box$ (anytime), and $\Diamond$ (sometime) [3]. Formulas with no temporal operators can be considered as being available only at present. Adding $\bigcirc$ to a formula A, i.e. $\bigcirc$A, means that A can be used only at the next time

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

and exactly once. Similarly, $\Box$A means that A can be used exactly once and at any time, and the choice is internally decided, as appropriate to one's own capability. $\Diamond A$ means that A can be used once at some time and the choice of time is externally decided by others.

Section 2 introduces our framework based around an example of online sale interactions. Then we discuss the advantages and limitations of using our framework to model interaction protocols and achieve flexibility as well as conclusions.

2. OUR FRAMEWORK

2.1 Encoding In Temporal Linear Logic

We consider encoding of resources, capabilities and commitments in agent interaction.

A unit of consumable resources is modeled as a proposition. We also include information about the location and ownership into the encoding of resources. We denote $A@\alpha_{\alpha}$ as that a resource A is located at (@) and owned (subscript $_{\alpha}$) by agent α and is equivalent to $A \otimes Locate(A, \alpha) \otimes Own(A, \alpha)$. Numeric figures are further used to abbreviate a multiplicative conjunction of the same instances. For instance, 2 $dollar = dollar \otimes dollar$. Moreover, such $\bigcirc^3 A$ is a shorthand for $\bigcirc \bigcirc \bigcirc A$.

The capabilities of agents refer to producing, consuming and relocating resources. The general representation form is $\Gamma \multimap$, in which Γ describes the conditions before and describes the condition after an application of the capability.

We model non-conditional commitments by negative formulas. For example, a commitment to provide a cricket bat is represented as $cricket\_b^{\perp}$. A non-conditional commitment is resolved when the committed agent brings about the respective resource or action as required by the commitment, which means there is a presence of a formula like $cricket\_b$, which can remove the negative formula.

A conditional commitment to bring about some effects given certain conditions can be modeled by connecting the conditions to the effects via a linear implication. A general form is $\Gamma \multimap$ where Γ is the condition part and is the effect part. If the condition Γ is derived, by consuming Γ, the linear implication will ensure that is resulted. If the conditions can not be achieved, the linear implication can not be applied and hence the commitment is then released.

Capabilities and conditional commitments, despite their distinct meanings, can be viewed as rules of interaction. Protocols can then be specified by appropriate interaction rules.

2.2 Example Scenario

We consider agents whose states are determined by a set of resources available, interaction rules and capabilities together with commitments. The scenario is an on-line sale of cricket bats from a merchant (Mer) to a customer (Cus).

At agent Cus

Cus has an amount of 50 dollars available at any time, and can pay by credit card: $\Box\$50@C \otimes \Box credit\_paym@C_C$.

Cus has a commitment of obtaining 2 cricket bats at some time: $[\Diamond(2cricket\_b@C_C)]^{\perp}$.

At agent Mer

Mer has available at any time 200 cricket bats for sale and 20 gifts: $200 \Box cricket\_b@M_M \otimes 20 \Box gift@M_M$.

Rule 1: Mer commits offering two cricket bats and a gift ($2cricket\_b@M_M^{\perp} \otimes gift@M_M^{\perp}$) to Cus ($2 \Box cricket\_b@C_C \otimes \Box gift@C_C$) if Cus pays 20 dollars ($20\$@C_C$) either via Paypal ($Paypal@C_C$) or by credit card ($credit\_paym@C_C$). The choice is at Cus. There are only 30 promotional offers like this.

$30 \Box [20\$@C_C \otimes (Paypal@C_C \oplus credit\_paym@C_C)$
$\multimap 20 \Box \$@M_M \otimes 2 \Box cricket\_b@C_C \otimes 2[cricket\_b@M_M]^{\perp} \otimes \Box gift@C_C \otimes gift@M_M^{\perp}]$

Rule 2: Similarly, Mer will offer a cricket bat to Cus if Cus pays 10 dollars either via Paypal or by credit card. Access to rule 2 can be times, up to Mer's policy.

$\Box [10\$@C_C \otimes (Paypal@C_C \oplus credit\_paym@C_C)$
$\multimap 10 \Box \$@M_M \otimes \Box cricket\_b@C_C \otimes cricket\_b@M_M^{\perp}]$

2.3 Interaction

To achieve goals, agents use TLL inference rules to construct proofs from state formulas and interaction rules. A simple model of negotiation is used. A rule can be proposed to another agent as a commitment it is willing to take. If the other agent agrees, both agents are involved in the commitment. Otherwise, the other agent can reject the proposal or counter-propose another rule.

The interaction begins by Cus trying to fulfill its pending commitment.

1. Cus asks Mer for two cricket bats to be delivered to Cus at some time.

C to M: **REQUEST** $[\Diamond(2cricket\_b@C_C)]^{\perp}$

To meet the request, Mer looks for applicable rules. One application of rule 1 or two applications of rule 2 at present or any future time t can derive $\bigcirc^t 2 \Box cricket\_b@C_C$ or $2 \bigcirc^t \Box cricket\_b@C_C, t \geq 0$ and either of them can remove $[\Diamond(2cricket\_b@C_C)]^{\perp}$.

2. Mer will propose each rule to Cus. n is any time up to Mer's choice. With similar analyzes, Cus determines that given their conditions can be satisfied, the proposals can help to derive its request.

We assume that Cus decides that it prefers having a gift and hence it will accept the proposal of rule 1 and refuse the proposal of rule 2.

M to C: **PROPOSE** $\bigcirc^n [20\$@C_C \otimes (Paypal@C_C \oplus credit\_paym@C_C) \multimap 20 \Box \$@M_M \otimes 2 \Box cricket\_b@C_C \otimes 2[cricket\_b@M_M]^{\perp} \otimes \Box gift@C_C \otimes gift@M_M^{\perp}]$

C to M: **ACCEPT**

M to C: **PROPOSE** $\bigcirc^n [10\$@C_C \otimes (Paypal@C_C \oplus credit\_paym@C_C) \multimap 10 \Box \$@M_M \otimes \Box cricket\_b@C_C \otimes cricket\_b@M_M^{\perp}]$

C to M: **REFUSE**

The conditions of the accepted proposal at the application time $n1$ is $\bigcirc^n (20\$@C_C \otimes (Paypal@C_C \oplus credit\_paym@C_C))$. One way to satisfy the conditions is for Cus, at the next n time points, to derive 20 dollars($\bigcirc^n 20\$@C_C$); and to pay via Paypal ($\bigcirc^n Paypal@C_C$) OR to pay by credit card ($\bigcirc^n credit\_paym@C_C$).

3. $\bigcirc^n 20\$@C_C$: as Cus has 50 dollars, it can use 20 dollars: $20 \Box \$@C_C \vdash \Box 20\$@C_C \vdash \bigcirc^n 20\$@C_C$.

There are two options for payment, the choice ($\oplus$) is at agent Cus.

4. We assume that Cus decides to use credit payment. We have $\Box credit\_paym@C_C \vdash \bigcirc^n credit\_paym@C_C$

5. (3) and (4) together allow the initial proposal to be applied to derive $\bigcirc^n \Box gift@C_C$ and $\bigcirc^n 2 \Box cricket\_b@C_C$ for Cus and commitments of Mer of $\bigcirc^n gift@M_M^{\perp}$ and $\bigcirc^n 2[cricket\_b@M_M]^{\perp}$, which are also resolved by the re-

sources $\Box gift@M_M$ and $2\Box cricket\_b@M_M$ available at Mer ($\bigcirc^n gift@M_M^{\perp} \otimes \Box gift@M_M \vdash \perp$ and $\bigcirc^n 2[cricket\_b@M_M]^{\perp} \otimes 2 \Box cricket\_b@M_M \vdash \perp$). Also, any value of n that satisfies $n \geq 0$ will allow Mer to fulfill Cus's initial request of $[\Diamond(2cricket\_b@C_C)]^{\perp}$. The interaction succeeds as all commitments are resolved.

2.4 Flexibility

The flexibility desired has been achieved in the example. Mer can choose the time n to apply the sale proposal. In step 2, Cus can choose among the sale proposals offered by Mer to explore the opportunity in the promotional period. Also, it is Cus's decision to proceed with the preferred payment method - step 4. Moreover, attempts at deriving 20 dollars in step 3 or carrying out payment - step 4 - can be switched order, as can be the attempts of Mer at fulfilling the commitments of giving 2 cricket bats and giving the gift in step 5.

3. DISCUSSION AND FURTHER WORK

The basic execution mechanism can be defined by proof construction, i.e. building a proof of the goal formula from the current state and the interaction rules. This means that an appropriate interaction sequence for the agents can be extracted from this proof.

As we have seen, TLL allows a natural and expressive specification of sequences of agent interaction. Temporal operators ($\bigcirc$, $\Box$ and $\Diamond$) and their combinations help to allocate resources or actions in time and express the time order of events. Linear implication ($\multimap$) can express a removal or a consumption together with its consequences. Hence, "consumed resources $\multimap$ produced resources" models resource transformation; "a resource at one agent $\multimap$ that at another agent" models resource relocation; "a resource owned by one agent $\multimap$ one owned by another agent" models ownership change; and "one state $\multimap$ another" models state update. Linear implication also reinforces the commitments that effects achieved only when their conditions are fulfilled.

Furthermore, given that flexibility includes the ability to make a sensible choice, having the choices expressed explicitly ($\&, \oplus, \Diamond$ and $\Box$) in the specification of interaction protocols provides agents with an opportunity to reason about the right choices during interaction and hence explore the flexibility in them.

In addition, by having the order of rules' applications be determined dynamically by the proof construction rather than be predefined, agents can gain flexibility. Moreover, because changes in the environment can be regarded as removing or adding formulas onto the current state formulas, the proof construction by agents on the current state formulas may require a different set of interaction rules to be used for interaction. This alteration of interactive behaviors to deal with changes brings flexibility into agent interaction.

Moreover, as the proof construction selects a sequence of interaction rules for interaction, selecting from all the possible combinations of interaction rules as in our framework gives more chances and flexibility than selecting from a few fixed (predefined) sequences. It is also more likely for agents to handle exceptions or explore opportunities that arise.

However, as all the temporal operators in TLL refer to concrete time points, one drawback is that we can not express durations in time faithfully. One major disadvantage of simulating a duration of an event by spreading copies of that event over adjacent time points continuously (like $\bigcirc A \otimes \bigcirc^2 A \otimes \ldots \bigcirc^{10} A$) is that it requires the time range to be provided explicitly.

Another disadvantage of our approach is that the rules for interaction require some detailed knowledge of the formulas of temporal linear logic. Clearly it would be beneficial to have a visually-based tool similar to UML diagrams which would allow non-experts to specify the appropriate rules without having to learn the details of the formulas themselves.

Similar works in [5] explore the advantages of TLL by using partial deduction techniques to help agents figure out the missing capabilities or resources and based on that, to negotiate with other agents about cooperation strategies. Our approach differs in bringing the concept of commitment into the modeling of interaction, and providing a more natural and detailed map for specifying interaction, especially about choices, time and updating using the full propositional TLL.

Many other approaches like using Commitment Machines [6] to model protocols use the commitment concept to bring more meaning into agents' interactive actions and hence allow them to be more flexible. However, these approaches use logic systems that are limited in their expressiveness to model resources.

Our further work will include using TLL to verify various properties of interaction protocols such as liveness and safety. Also, we will investigate modeling commitments that can be breached together with the conventions that monitor agents' handling of commitments. This also includes managing conflicts among commitments.

Acknowledgments

We are very thankful to Binh Tran and Michael Winikoff for many stimulating and helpful discussions of this material.

4. REFERENCES

[1] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[2] J. Harland and M. Winnikoff. Agent negotiation as proof search in linear logic (poster). In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002)*, Bologna, July 2002.

[3] T. Hirai. *Temporal Linear Logic and Its Applications*. Doctoral Dissertation, Graduate School of Science and Technology, Kobe University, Japan, 2000.

[4] N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.

[5] P. Küngas. Temporal linear logic for symbolic agent negotiation. *Lecture Notes in Artificial Intelligence*, 3157:23–32, 2004.

[6] P. Yolum and M. P. Singh. Commitment machines. In *Proceedings of the 8th International Workshop on Agent Theories, Architectures, and Languages (ATAL-01)*, pages 235–247. Springer-Verlag, 2002.

Run-Time Model Checking of Interaction and Deontic Models for Multi-Agent Systems

Nardine Osman
School of Informatics
The University of Edinburgh
Edinburgh EH8 9LE, UK
N.Osman@sms.ed.ac.uk

David Robertson
School of Informatics
The University of Edinburgh
Edinburgh EH8 9LE, UK
dr@inf.ed.ac.uk

Christopher Walton
School of Informatics
The University of Edinburgh
Edinburgh EH8 9LE, UK
cdw@inf.ed.ac.uk

ABSTRACT

This paper is concerned with the problem of obtaining predictable interactions between groups of agents in open environments when individual agents do not expose their BDI logic. The most popular approaches to this in practise have been to model interaction protocols and to model the deontic constraints imposed by individual agents. Both of these approaches are appropriate and necessary but their combination creates the practical problem of ensuring that interaction protocols come into contact with agents that possess compatible deontic constraints. This is essentially an issue of property checking dynamically at run-time. We show how model checking can be applied to this problem.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; F.3.1 [**Specifying and Verifying and Reasoning about Programs**]: Specification techniques; D.2.4 [**Software / Program Verification**]: Model checking

General Terms

Verification

Keywords

dynamic model checking, interaction models, deontic models

1. INTRODUCTION

One of the most fundamental challenges of multi-agent system engineering is to enable predictable, reliable interaction amongst groups of agents without requiring a deep standardisation of the way in which they are engineered and while preserving as much as possible of their autonomy in individual reasoning. It is not plausible that agents built independently and with no agreement on forms of interaction could be predictably reliable, so researchers have searched for ways of standardising some aspects of coordination in the hope that this small amount of standardisation would provide predictability sufficient for important tasks. Two contrasting approaches to this problem have emerged:

- The use of explicit **models of interactions** (in a generic process or state-machine language) along with mechanisms by which agents can locate, reason about and participate in models of interaction that they judge appropriate. These are built to describe forms of interaction (thus are detached from individual agents) and typically are accessed when an agent anticipates it wants to initiate or join that type of interaction.
- The specification of constraints imposed by individual agents on the interactions they will allow: **deontic constraints**. These are built locally for an individual agent and typically accessed when a specific interaction is anticipated with an individual agent.

These two approaches are compatible, in the sense that they attack different aspects of a similar problem. They have not, however, been combined. Hitherto, this has not been a major problem, because neither approach had significant user communities. Those now are developing so that it is of practical importance to have a way of answering the question "Given an interaction model and an agent with given deontic constraints wishing to participate in that model, could that combination work?" This is a difficult question because both interaction models and deontic constraints are sophisticated logical objects, so the inference involved is not (say) simple term matching or subsumption. It also is difficult because in practise the question must be answered automatically and in real time, so inference mechanisms must be self contained and efficient. We show how a lightweight dynamic model checker can tackle this problem by allowing agents to invoke it at interaction time for verifying both the interaction and deontic models.

2. MOTIVATING EXAMPLE

Let us consider the travel agency scenario [1], where a customer agent, interested in making flight and hotel reservations, contacts a broker to find the most suitable travel agent for this scenario. The travel agent will have to deal with airline web services, query a hotel directory, and confirm customer's payments via a credit card web service.

The interaction model will specify the global rules of interaction. For example, it sates that the interaction starts

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

when the customer agent sends it vacation details (dates / destination) to the travel agent. Then, the travel agent sends a query to the involved airline web services with the given vacation details, and so on. The interaction model may be thought of as a state-space graph that specifies the allowed order of message passing actions between agents.

On the other hand, the deontic model specifies local individual constraints imposed on the various agents involved in the interaction. For example, a constraint on the broker agent is to not accept a travel agent that has no access to the selected hotel directory. Several constraints may be imposed on the customer agent. For example, it might not have access to a *MasterCard*, or it may be forced to encrypt its messages in a specific technology, say OpenPGP only. Similarly, other agents would have other sets of constraints.

The system is modelled through the combination of interaction and deontic models (Figure 1). Note that unlike the global interaction model which constructs a unique state-space graph for coordinating interactions, the deontic model is a collection of the involved agents' local constraints. These constraints could address a diverse collection of issues. They are usually used to tackle issues such as access control, authorisation, authentication, trust, security, etc.

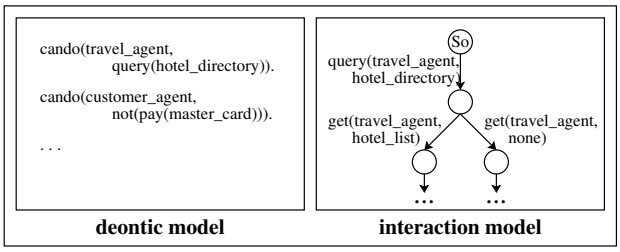


Figure 1: A conceptual model of the system model

Now let us consider the following complication. The customer agent contacts the broker agent requesting an appropriate travel agent. With the broker being responsible for finding suitable agents for a given interaction protocol, it should also be capable of verifying, at interaction time, that the protocol it has instantiated with agents is likely to work. This, however, relies on the correctness of the interaction protocol as well as the compatibility of the chosen agents. For example, trying to ally the customer agent with a travel agent that does not have access to a hotel directory will result in a scenario failure, regardless of whether the interaction protocol itself is error free or not. This requires a verifier that can handle both interaction and deontic constraints, and is capable of operating automatically at run-time.

3. DESIGN & IMPLEMENTATION PLANS

Our goal is to achieve a verifier which could be used by agents at interaction time for verifying MAS through the verification of the interaction and deontic rules. Figure 2 illustrates our design plan.

Because open systems consist of autonomous agents, it is necessary for agents to be capable of automatically verifying, at run-time, dynamic protocols affected by dynamic deontic rules. We choose model checking from amongst other verification techniques because it provides a fully automatic verification process which could be carried out by the agents

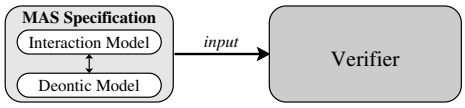


Figure 2: Design plan

during interaction time.

The Model Checking Problem. The model checking problem can be defined as follows: *Given a finite transition system S and a temporal formula ϕ, does S satisfy ϕ?* The model checking process is divided into three stages: modelling, specification, and verification. The system to be verified must first be modelled in the language of the model checker S. The properties (ϕ) to which the system model is verified upon should be specified using the model checker's temporal logic. Both the system model and the properties specification are fed to the model checker for verification. The model checker is, essentially, an algorithm that decides whether a model S satisfies a formula ϕ. In our case, the system model is a combination of interaction and deontic models. Figure 3 illustrates our model checking process.

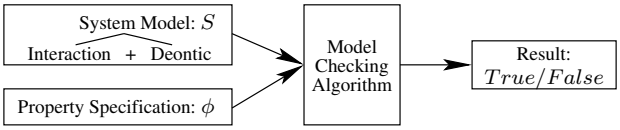


Figure 3: The model checking process

We choose the lightweight coordination calculus (LCC) [4] for specifying our interaction model, some policy language for specifying the deontic model, and the μ-calculus temporal logic for property specifications. The model checking technique implemented is a logic-based local model checking technique. The LCC language provides us with an executable specification that allows interaction models to be portable, and hence shared and global. This also supports dynamic model checking, where agents can automatically retrieve the protocol state and feed it to the model checker for verification. LCC also links cleanly to traditional process calculi used in model checking. The combination of μ-calculus and a logic-based local model checking technique results in an extremely compact size model checker, which actually throws the burden of the state-space search on the underlying tabled Prolog system. While [2] explores these issues in detail, this paper presents a higher-level description of the dynamic model checker irrespective of the specification details.

4. DYNAMIC MODEL CHECKING

At interaction time — when the conditions for verification are met and the related deontic models are obtained — the agent, interested in the satisfaction of property ϕ with respect to a given scenario, feeds the lightweight model checker with the following:

- the system model S (e.g. Figure 1), which consists of:
  - a global interaction model
  - a set of local deontic models
- the property ϕ to be verified

4.1 Example revisited

Figure 4 provides a section of the file that is fed to the model checker by the broker agent for verifying the travel agency scenario it is about to instantiate. The interaction model is a shared global model that specifies the rules of the interaction. The interaction model section presented in Figure 4 deals with a section of the interaction between the travel agent and the customer agent. More specifically, it shows that when the customer agent sends its chosen flight, the travel agent reacts by querying the hotel directory and sending a list of available hotels for the customer to choose from. The interaction model, written in LCC, is essentially a specification of the interaction's state-space graph.

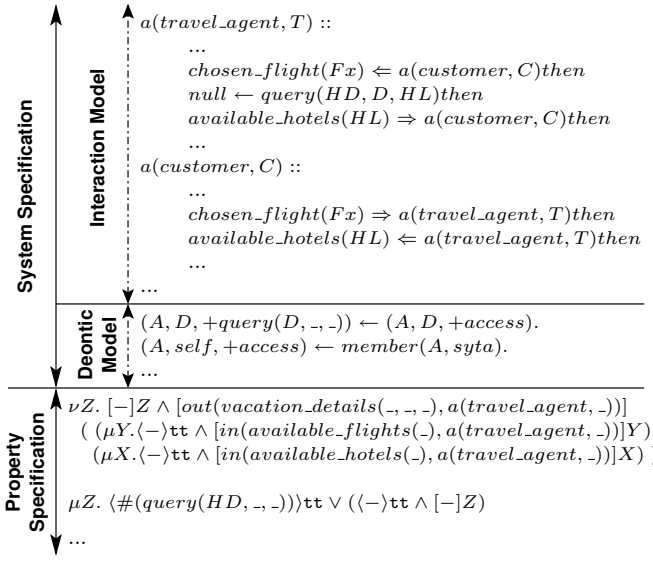


Figure 4: System model fed to the model checker

The deontic rules are constraints on the various agents engaged in the interaction. Unlike the interaction model, these do not impose constraints on the message passing action, but on any other action an agent can take. The first rule in the sample presented by Figure 4 specifies that an agent can query a directory only if it has access to it. This rule would belong to the broker agent who is interested in picking the right travel agent. The second rule specifies that an agent might access the directory only if it was a member of the student youth travel association (*stya*). This rule would belong to the hotel directory for specifying the appropriate authorisations.

Finally, the system model is to be verified against a given set of properties. The 'property specification' section of Figure 4 provides a sample of the properties that could be verified for our travel agency scenario. The first specifies that if the customer agent requests a vacation package, it will always eventually receive a list of flights and hotels. The second property specifies that in every run of the interaction, it is always the case that the hotel directory is eventually queried.

Now when the broker needs to verify that the interaction protocol will work properly for the selected set of agents, it will feed the file presented by Figure 4 to the dynamic model checker. The verification process is outlined below.

4.2 The verification process

The interaction model, specified with the LCC process calculus, is essentially a specification of the interaction's state-space graph. While global model checking is based on generating the whole state-space, local model checking partially constructs the state-space one step at a time until a solution is reached. Verification, in the case of our local model checker, starts at the initial state s_0 and tries to verify that the property ϕ is satisfied at s_0[1]. If it succeeds, the verifier terminates and the property is said to be *satisfied*. Otherwise, the verifier attempts to make a transition(s) to the next state(s) in the state-space[2]. If the transition(s) violates any of the deontic rules, then the verification process terminates and the property is *not satisfied*. Otherwise, the satisfaction of the property ϕ is verified with respect to the new state(s), and the whole process is repeated all over again.

This method raises two questions: (1) how do we ensure termination? and (2) how do we ensure correctness? Inspired by the XMC model checker [3], we build our model checker on top of XSB [5], a tabled Prolog system. Caching in tabled Prolog ensures termination, avoids redundant subcomputations, and computes the well-founded model of normal logic programs. Our model checker, unlike XMC, accepts LCC interaction models in parallel with deontic models.

5. CONCLUSION

This paper presents a verification technique for predicting and preventing failure — due to errors within the interaction protocol, conflicts between the interaction protocol and agents' rules and requirements, as well as clash of interests between agents — as early as possible in MAS scenarios. This is made possible by allowing the agents to invoke the model checker at runtime, when the conditions for verification are met, and verify that certain properties hold for the given combination of a shared interaction model and agents' local deontic models.

6. REFERENCES

[1] H. He, H. Haas, and D. Orchard. Web services architecture usage scenarios. *W3C Working Group Note*, Feb 2003. http://www.w3.org/TR/2004/NOTE-ws-arch-scenarios-20040211/.

[2] N. Osman, D. Robertson, and C. Walton. Run-time model checking of interaction and deontic models for multi-agent systems. In *Proceedings of the Third European Workshop on Multi-Agent Systems*, 2005.

[3] C. R. Ramakrishnan, I. V. Ramakrishnan, S. A. Smolka, Y. Dong, X. Du, A. Roychoudhury, and V. N. Venkatakrishnan. XMC: a logic-programming-based verification toolset. In *Computer Aided Verification*, Lecture Notes in Computer Science, 2000.

[4] D. Robertson. A lightweight coordination calculus for agent social norms. In *Proceedings of Declarative Agent Languages and Technologies workshop*, 2004.

[5] K. Sagonas, T. Swift, and D. S. Warren. XSB as an efficient deductive database engine. In *Proceedings of the 1994 ACM SIGMOD international conference on Management of data (SIGMOD '94)*, 1994.

[1]Satisfaction is verified by applying the modal μ-calculus proof rules presented in [2].

[2]Transitions are made based on the LCC transition rules presented in [2].

Modeling Mental States in the Analysis of Multiagent Systems Requirements

Alexei Lapouchnian
Department of Computer Science
University of Toronto
Toronto, ON, M5S 3G4, Canada
alexei@cs.toronto.edu

Yves Lespérance
Department of Computer Science and Engineering
York University
Toronto, ON, M3J 1P3, Canada
lesperan@cs.yorku.ca

ABSTRACT
This paper presents an agent-oriented requirements engineering approach that combines informal *i** models with formal specifications in the CASL language through the use of Intentional Annotated Strategic Rationale diagrams. In the resulting framework, agent goals and knowledge are represented as their mental states, which allows for the formal analysis of, among other things, agent interactions and incomplete knowledge. CASL models can also serve as high-level specifications for multiagent systems.

Categories and Subject Descriptors
D.2.1 [**Software Engineering**]: Requirements/Specifications – *methodologies*; I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence – *Multiagent Systems*

General Terms
Theory, Design, Verification.

Keywords
Agent-Oriented Software Engineering, Logics for Agent Systems, Goal-Oriented Requirements Engineering.

1. INTRODUCTION
Agent technology is well-suited for constructing applications that are open, distributed, and adaptable to change. We need software engineering methods that are tailored for designing systems with multiple communicating agents that strive to achieve their assigned goals thereby delivering the desired application.

In requirements engineering (RE), *goal-oriented approaches* (e.g., Tropos [1]) have become prominent. In Goal-Oriented Requirements Engineering (GORE) high-level stakeholder objectives are modeled as goals and refined into fine-grained requirements or environment expectations. Their reliance on goals makes GORE methods and agent-oriented software engineering a great match. Therefore, it is natural to use a goal-oriented requirements engineering approach when developing MAS. With GORE, it is easy to make the transition from the requirements to the high-level MAS specifications. In this context, while it is possible to informally analyze small systems, formal analysis is needed for any realistically-sized one. Thus, the aim of this work is to devise an agent-oriented requirements engineering approach with a formal component that supports reasoning about agents' goals (and knowledge), thereby allowing for rigorous formal analysis of the requirements expressed as the objectives of the agents in a MAS.

In our approach, we integrate the *i** goal-oriented modeling framework [11] with CASL [8], a formal agent-oriented specification language that supports the modeling of agent mental states. This gives the modeler the flexibility and intuitiveness of the *i** notation as well as the powerful formal analysis capabilities of CASL. To bridge the gap between informal *i** diagrams and formal CASL specifications we propose an *intermediate notation* that can be easily obtained from *i** models and then mapped into CASL. With our *i**-CASL-based approach, a CASL model can be used both as a requirements analysis tool and as a formal high-level specification for a multiagent system that satisfies the requirements. This model can be formally analyzed using the CASLve [9] tool or other tools and the results can be fed back into the requirements model.

2. BACKGROUND
*i** [11] is an agent-oriented modeling framework used for requirements engineering, etc. The *i** framework has two main components. Firstly, the *Strategic Dependency (SD) model* provides an external view of actors and the intentional dependencies between them. There can be four types of dependencies based on what is being delegated – a goal, a task, a resource, or a softgoal dependency. Secondly, the *Strategic Rationale (SR) model* describes the internals of actors (see Figure 1). The same four types of nodes are used in SR models along with task decompositions and means-ends links. Softgoals are used to represent non-functional requirements. Softgoal contribution links in SR models specify how process alternatives fare with respect to quality concerns. In Figure 1, we see a fragment of an (annotated) SR model showing the interactions between a meeting initiator agent (MS) and a meeting scheduling system agent (MS). The MS depends on the MI for information on the suggested meeting date range. The diagram shows how some of the agents' tasks are decomposed.

The Cognitive Agents Specification Language (CASL) [8] is a formal specification language that combines theories of action [6] and mental states [7] expressed in the situation calculus [5] with ConGolog [2], a concurrent, non-deterministic agent-oriented

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

programming language with a formal semantics. CASL specifications consist of two parts: the model of the domain and its dynamics (the declarative part) and the specification of the agents' behaviour (the procedural part). The domain is modeled in terms of *agents*, *primitive actions* executed by agents, *situations*, which are states of the domain that result from the execution of sequences of actions, and *fluents*, which are predicates and functions that may change from situation to situation. Axioms of the following types are used: *action precondition axioms*, *successor state axioms* (SSA) [6] that specify how primitive actions affect fluents, and *initial state axioms* describing the initial state of the domain and the initial mental states of the agents. Agent behaviour is specified using a rich high-level programming language with recursive procedure declarations, loops, conditionals, non-determinism, concurrency, and interrupts [2].

CASL supports formal modeling of agents' goals and knowledge based on a *possible worlds semantics* incorporated into the situation calculus, where situations are viewed as possible worlds [7]. **Know***(agt,φ,s)* specifies that an agent knows some formula *φ*, while **Goal***(agt,ψ,s)* indicates that in situation *s* the agent *agt* has the goal that *ψ* holds. We commonly use the formula **Goal***(agt,***Eventually***(ψ),s)* to state that *agt* has the goal that *ψ* is eventually true. Primitive communication actions such as `request`, `inform`, and `cancelRequest` are defined, which affect the mental states of the agents involved in the obvious way.

3. THE APPROACH

3.1 Increasing Precision with iASR Models

Our goal is to tightly associate *i** models with formal specifications in CASL. The standard SR diagrams are geared to informal analysis and are imprecise. For instance, they do not provide any information on whether subtasks are to be executed sequentially or concurrently, under certain conditions, etc. However, CASL is a precise language. To handle this mismatch we use Intentional Annotated SR (iASR) models, which are precise graphical representation for the procedural part of CASL specifications.

To develop an iASR diagram for an actor, one starts with the regular SR diagram for that actor. It then can be appropriately transformed to become an iASR model every element of which can easily be mapped into CASL. The main steps for producing iASR models from SR ones are discussed below.

Annotations. The main tool for disambiguating SR models is *annotations* – textual constraints on SR models [10]. *Composition annotations* (sequence (";"), concurrency ("||"), prioritized concurrency (">>"), and alternative ("|")) are applied from right to left and specify how the subtasks/subgoals are to be combined to execute the supertask or achieve the goal. *Link annotations* are applied to subtasks/subgoals and specify how/under what condition they are supposed to be achieved/executed. These annotations are: *while* loop, *for* loop, *if* condition, the pick (non-deterministic choice of argument), the interrupt, and the guard (it blocks execution until its condition becomes true). The latter two are commonly used modify agents' behaviour based on changes in their mental states. *Applicability condition annotations* specify when the goal achievement alternatives are applicable.

Softgoals. Softgoals are imprecise and thus cannot be handled in a formal specifications language. Therefore, softgoals are used to choose the best process alternatives and then are removed before iASR models are produced. Alternatively, softgoals can be operationalized or metricized, thus becoming hard goals.

Providing agent interaction details. While *i** abstracts from modeling agent interaction details, CASL models requests for services, monitoring for such requests, etc., which are important for formally analyzing MAS. Thus, high-level aspects of agent interaction must be specified in iASR models. In the iASR model in Figure 1 we can see that the interaction between the MS and the MI to exchange the meeting date range information has been specified. Note the use of the interrupt annotation that triggers when the MI acquires the goal to inform the MS about the meeting the date range following the MS's request. We can see here how the goals of individual agents are explicitly modeled. iASR models can also refer to agents' knowledge. For example, the guard annotation on the MS ensures that it can only proceed when it knows the date range.

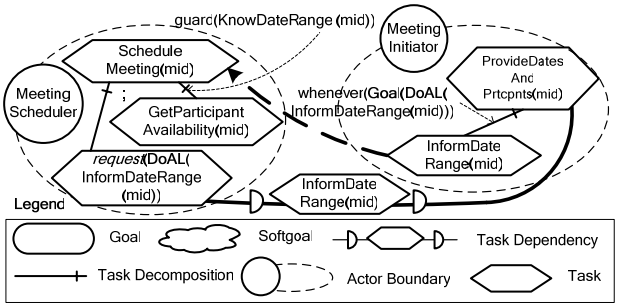


Figure 1. Adding agent interaction details to iASR model.

3.2 Mapping iASR Diagrams into CASL

Once all the necessary details have been introduced into an iASR diagram, it can be mapped into the corresponding formal CASL model, thus making the iASR model amenable to formal analysis.

The modeler defines a mapping **m** that maps every element (except for intentional dependencies) of an iASR model into CASL. This mapping associates iASR model elements with CASL procedures, primitive actions, and formulas so that a CASL program can be generated from an iASR model. Specifically, agents are mapped into CASL procedures that specify their behaviour; roles and positions (these can be played/occupied by agents) are mapped into similar procedures with an agent parameter so that they can be instantiated by individual agents; leaf-level task nodes are mapped into CASL procedures or primitive actions; composition and link annotations are mapped into the corresponding CASL operators, while conditions present in the annotations map into CASL formulas. Non-leaf task nodes are mapped into procedures that are automatically constructed from procedures/actions corresponding to the subtasks, which are combined based on the annotations used. For example, the task `ScheduleMeeting` in Figure 1 will be mapped into the following CASL procedure:

```
proc ScheduleMeeting(mid)
  request(MS,MI,DoAL(InformDateRange(mid)));
  guard(KnowDateRange(mid)) do
    GetParticipantAvailability(mid)
  endGuard
endProc
```

A goal node is mapped into a formula (its formal definition) and an achievement procedure that is automatically constructed based on the means for achieving the goal specified in the iASR model.

Intentional dependencies are not mapped into CASL per se – they are established by the associated agent interactions. iASR tasks requesting services from agents will be mapped into actions of the type `request(FromAgt,ToAgt,`**`Eventually`**`(φ))` for achievement goals φ. To ask an agent to execute a certain known procedure, the depender must request it with: `request(FromAgt,ToAgt,`**`DoAL`**`(Procedure))`. This requests the execution of the procedure while allowing `ToAgt` to perform other activities. The service provider must monitor for its mental state changes with appropriate interrupts (see Figure 1).

3.3 Formal Verification

Once an iASR model is mapped into CASL, it is ready to be formally analyzed. One tool that can be used is CASLve [9], a theorem prover-based verification environment that can be employed to verify executability of programs, their termination, etc.

Other approaches can also be used. However, tools based on techniques like simulation work with much less expressive languages than CASL requiring simplification of CASL specifications before these methods can be used on them. For example, most simulation tools cannot handle mental state specifications; these would then have to be operationalized before simulation is performed. The ConGolog interpreter can be used to directly execute such simplified specifications, as done in [10].

If expected properties of the system are not entailed by the CASL model, it means that the model is incorrect and needs to be fixed. The source of an error found during verification can usually be traced to a portion of the CASL code and to a part of its iASR model since our systematic mapping supports traceability.

4. DISCUSSION

In the approach presented in this paper and in [4], we produce CASL specifications from *i** models for formal analysis and verification. The approach is related to the Tropos framework [1] in that it is agent-oriented and is rooted in RE concepts. Our method is not the first attempt to provide formal semantics for *i** models. For example, Formal Tropos (FT) [3] supports formal verification of *i** models through model checking. Also, in the *i**-ConGolog approach [10], on which our method is based, SR models are associated with formal ConGolog programs for simulation and verification. The problem is that in these methods, goals of the agents are abstracted out of formal specifications. This is done due to the fact that the formal components of these approaches (the model checker input language for FT and the restricted situation calculus theories for the others) do not support reasoning about agent goals (or knowledge). However, most of the interactions among agents involve knowledge exchange and goal delegation since multiagent systems are developed as social structures. Thus, complementing informal modeling techniques such as *i** with formal analysis of agent goals and knowledge is very important in the design of multiagent systems. We propose a framework where goals are not removed from the agent specifications, but are modeled formally and can be updated following requests. This allows agents to reason about their objectives. Information exchanges among agents are also formalized as changes in their knowledge state. In our approach, goals are not system-wide properties, but belong to concrete agents. The same applies to knowledge. This subjective point of view provides support for new types of formal analysis on CASL specifications. Thus, this method is more agent-oriented and allows for precise modeling of stakeholder goals. Modeling of conflicting stakeholder goals, a common case in requirements engineering, and agent negotiations is also possible. All of the above can be useful for the later stages of AOSE since most specifications of agent communications model them as changing agents' mental states. Moreover, many agent implementation languages/tools are based on models of mental states.

The full paper is available at: www.cs.toronto.edu/~alexei.

5. REFERENCES

[1] J. Castro, M. Kolp and J. Mylopoulos. Towards Requirements-Driven Information Systems Engineering: The Tropos Project. *Information Systems*, 27(6):365-389, 2002.

[2] G. De Giacomo, Y. Lespérance and H. Levesque. ConGolog, A Concurrent Programming Language Based on the Situation Calculus. *Artificial Intelligence*, 121:109-169, 2000.

[3] A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, P. Traverso. Specifying and Analyzing Early Requirements in Tropos. *Requirements Engineering Journal*, 9(2):132-150, May 2004.

[4] A. Lapouchnian. Modeling Mental States in Requirements Engineering - An Agent-Oriented Framework Based on *i** and CASL. M.Sc. Thesis. Department of Computer Science, York University, Toronto, 2004.

[5] J. McCarthy and P. Hayes. Some Philosophical Problems From the Standpoint of Artificial Intelligence. *Machine Intelligence*, Volume 4, pp. 463-502, Edinburgh University Press, 1969.

[6] R. Reiter. The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression. In V. Lifschitz, Ed. Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy, pp. 359-380, Academic Press, 1991.

[7] R.B. Scherl and H. Levesque Knowledge, Action, and the Frame Problem. Artificial Intelligence, 144(1-2):1-39, 2003.

[8] S. Shapiro and Y. Lespérance. Modeling Multiagent Systems with the Cognitive Agents Specification Language - A Feature Interaction Resolution Application. Proc. *ATAL-2000*, pp. 244-259, LNAI 1986, Springer, 2001.

[9] S. Shapiro, Y. Lespérance and H. Levesque. The Cognitive Agents Specification Language and Verification Environment for Multiagent Systems. In C. Castelfranchi and W.L. Johnson Eds., Proc. *AAMAS'02*, Bologna, Italy, July 15-19, 2002, ACM Press.

[10] X. Wang and Y. Lespérance. Agent-Oriented Requirements Engineering Using ConGolog and *i**. Proc. *AOIS-01*, 2001.

[11] E. Yu. Towards modeling and reasoning support for early requirements engineering. In Proc. *RE'97*, Annapolis, USA.

A Stochastic Language for Modelling Opponent Agents

Gerardo Simari Amy Sliva Dana Nau V.S.Subrahmanian
University of Maryland
College Park, Maryland, USA
{gisimari,asliva,nau,vs}@cs.umd.edu

ABSTRACT

There are numerous cases where a reasoning agent needs to reason about the behavior of an opponent agent. In this paper, we propose a hybrid probabilistic logic language within which we can express what actions an opponent may take in a given situation. We present the syntaxis and semantics of the language, and the concept of a Maximally Probable Course of Action.

Categories and Subject Descriptors

I.2 [**Computing Methodolgies**]: Artificial Intelligence

Keywords

Logic for agent systems, Formal models of agency

1. INTRODUCTION

There are numerous multiagent applications where we need to build a *model* of an opponent agent. For example, consider a computer implementation of a multi-player game such as *Diplomacy* or *Civilization* — in such a game, a human (or agent) player may wish to have a model of how another player might play in order to decide on the best move or moves. Likewise, in strategic decision making applications such as negotiations, it is a good idea to have one or more models of how an opponent might behave.

Alternatively, we have been building an application for reasoning about the actions that certain tribes involved in the opium business in the Pakistan Afghanistan borderlands might take. We have been developing a set of rules that would allow us to predict what possible reactions could arise when US forces take certain actions. This has clear benefits as US forces would like to take those actions that have a desirable outcome with a high probability.

The two classes of applications - games and real world cultural reasoning scenarios - have one significant difference. Games are limited in scope and are governed by strict rules. Cultural models are less strict and operate under very different rules. Second, and more important, in the case of games, one can try to mine rules automatically from large collections of past games. In the case of cultural models, this is far harder. We are yet to see a single mined collection of rules dealing with any culture at all.

Moreover, most opponent models are *handcrafted* in a language that is selected on a case by case basis. In this paper, we develop a single uniform language called SOMA within which opponent models may be expressed *by a human* for a wide variety of applications ranging from automated game playing programs to strategic adversarial reasoning problems. Of course, in the case of games, automatically extracting models from past histories is relatively easy compared to performing similar extractions in cases where relevant histories are harder to find.

In Section 2, we develop a formal syntax for SOMA-programs. Section 3 develops a formal model-theoretic semantics for SOMA-programs, and Section 4 includes a declarative definition of the 'most probable course of action" (MPCOA) for the opponent.

2. SYNTAX OF SOMA-RULES

Throughout this paper, we assume the existence of a *reasoning agent* which wishes to model and reason about an *opponent agent*. At any given point in time, the reasoning agent may have a state. This state may allow it to, for example, record the actions she has observed an opponent making and/or include the beliefs of the reasoning agent. Without loss of generality, we will assume that the state of the reasoning agent is logically describable using an alphabet consisting of a finite set of *predicate symbols* (each with an associated arity), a finite set of *constant symbols*, and an infinite set of variable symbols. We assume that no function symbols are present. As usual, a *term* is either a constant symbol or a variable symbol. If p is an n-ary predicate symbol and $t_1, \ldots, t_n$ are all terms (resp. constants), then $p(t_1, \ldots, t_n)$ is an atom (resp. *ground* atom). The *observable state* at any given point in time is a set of ground atoms.

In addition, the reasoning agent believes in the existence of some finite set of elementary actions that the opponent can take. This set is represented by a *finite* set $\mathcal{A}$ of symbols $\mathcal{A}$ called *action symbols* disjoint from the set of predicate symbols. Every $a \in \mathcal{A}$ has an associated arity; if a is an n-ary action symbol, and $t_1, \ldots, t_n$ are all terms (resp. constants), then $a(t_1, \ldots, t_n)$ is an *action atom* (resp. *ground* action atom).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

DEFINITION 2.1 (ACTION FORMULA). *Every action atom is an action formula. If P and Q are action formulas, then so are $(P \wedge Q), (P \vee Q)$, and $\neg P$.*

An action formula denotes a complex action. For example, $\neg a$ is the action where the opponent does not perform a. $(a \wedge b)$ is the complex action where the opponent does both a and b.

DEFINITION 2.2 (SOMA-RULE). *If $B_1, \ldots, B_n$ are atoms, $0 \leq \ell \leq u \leq 1$ are probability values, and P is an action formula, then*

$$P : [\ell, u] \quad \leftarrow \quad B_1 \wedge \ldots \wedge B_n$$

The above rule is an elementary SOMA-rule *if P is an action atom.*

Intuitively, suppose $\mathcal{S}$ is the current state of the reasoning agent (including any beliefs about the opponent agent). The above rule states that if the state $\mathcal{S}$ satisfies all the atoms $B_1, \ldots, B_n$, then the opponent agent will take action(s) P with a probability between ℓ and u.

DEFINITION 2.3 (SOMA-PROGRAM). *A (elementary)* SOMA-*program is a finite set of (elementary)* SOMA-*rules.*

Here is an (admittedly simplistic) example of a simple SOMA-program – a more sophisticated and carefully developed program of this kind that extracts the rules using sample data such as *Gallup* poll data might enable an unprincipled politician to determine what stances will get him votes.

EXAMPLE 2.1. *Consider a simple example involving* politics. *By examining a set of surveys, we might be able to write down a set of rules applicable to an arbitrary member m of the population (opponent) as follows:*

$$\begin{aligned} vote\_myway : [0.2, 0.3] &\leftarrow democrat. \\ vote\_myway : [0.3, 0.6] &\leftarrow promise\_cut\_tax. \\ \neg vote\_myway : [0.3, 0.9] &\leftarrow promise\_cut\_tax. \end{aligned}$$

The first rule says that I (the reasoning agent) expect m to vote for me with 20 to 30% probability if I am a democrat. The probability goes up to 30 to 60% if I promise a tax cut (independently of whether I am a democrat or not). The probability that the example citizen either does not vote for me is 30 to 90% if I promise a tax cut.

SOMA-programs are variations of generalized probabilistic (gp) logic programs [Ng and Subrahmanian 1993] - the heads of SOMA-rules are more expressive than gp-logic program rule heads, but conversely the bodies of gp-logic programs are more expressive. The semantics of SOMA-programs builds on that of gp-programs and is briefly described below.

3. SEMANTICS OF SOMA-RULES

A reasoning agent can use a SOMA-program to represent its beliefs about an opponent agent and estimate what the opponent might do in a given situation.

A *course of action* (COA for short) that the opponent might take is any finite set of ground action atoms. Suppose $\mathcal{C}$ is the set of all possible courses of action. We assume the existence of a function ϕ which takes a course of action and a state as input, and returns either 0 or 1 denoting whether the course of action is feasible or not (a course of action may be infeasible, for example, if the actions in it are mutually incompatible or if the actions cannot be executed in the current state due to pre-conditions that do not hold or if the the new state that results does not satisfy some integrity constraints). In the following, we will write $gr(\mathcal{A})$ to denote the set of all ground action atoms. We first define what it means for a COA to satisfy an action formula P.

DEFINITION 3.1. *A course of action C* ***satisfies*** *a* ground *action formula P, denoted $C \mapsto P$ if:*

1. *P is an action atom and $P \in C$, or*
2. *P has the form $P_1 \wedge P_2$ and C satisfies P_1 and P_2, or*
3. *P has the form $P_1 \vee P_2$ and C satisfies P_1 or P_2, or*
4. *P has the form $\neg P_1$ and C does not satisfy P_1.*

A SOMA-interpretation defined below assigns a probability to each COA.

DEFINITION 3.2 (SOMA-INTERPRETATION). *A* SOMA-interpretation *I is a mapping from $\mathcal{C}$ to $[0, 1]$ such that $\Sigma_{X \in \mathcal{C}} I(X) = 1$.*

Intuitively, $I(X)$ represents the probability that the opponent will perform exactly those actions in X. Clearly, the opponent must choose some course of action which is why $\Sigma_{X \in \mathcal{C}} I(X) = 1$. Let us now return to our political example.

EXAMPLE 3.1. *We have two* COA*s: $C_1 = \{\}$; $C_2 = \{vote\_myway\}$. A* SOMA-*interpretation I_0 may assign 0.3 and 0.7 respectively to these two* COA*s.*

DEFINITION 3.3. *Suppose $\mathcal{S}$ is a state and I is a* SOMA-*interpretation. I is said to* ***satisfy*** *a ground rule*

$$P : [\ell, u] \quad \leftarrow \quad B_1 \wedge \ldots \wedge B_n$$

with respect to state $\mathcal{S}$ iff

1. *$\{B_1, \ldots, B_n\} \not\subseteq \mathcal{S}$ or*
2. *$\ell \leq \Sigma_{\phi(C, \mathcal{S}) = 1 \wedge C \mapsto P} I(C) \leq u$.*

We say that I satisfies a rule if and only if it satisfies all ground instances of the rule.

Let us return to our political example.

EXAMPLE 3.2. *Consider the first rule of example 2.1 and suppose the current state $\mathcal{S}_0 = \{democrat, promise\_cut\_tax\}$. The first rule is* not satisfied *by the interpretation I_0 of Example 3.1 because $I_0(vote\_myway) = 0.7 \notin [0.2, 0.3]$. Had the probability interval in the head of this rule been $[0.5, 0.8]$, then the rule would have been satisfied by I_0.*

We use the usual notion of consistency.

DEFINITION 3.4 (CONSISTENCY). *A* SOMA-*program Π is* consistent *with respect to state $\mathcal{S}$ if there is at least one* SOMA-*interpretation that satisfies all rules in Π.*

The following definition shows that we can associate a set of linear constraints with any SOMA-program.

DEFINITION 3.5 (CONS($\Pi, \mathcal{S}$)). *Suppose* Π *is a* SOMA*-program and* $\mathcal{S}$ *is a state. The* constraints associated with Π*, denoted* CONS($\Pi, \mathcal{S}$) *associates a variable* p_i *with each course of action* C_i *(denoting the probability of* C_i*) and are defined as follows:*

1. *If*

$$P : [\ell, u] \quad \leftarrow \quad B_1 \wedge \ldots \wedge B_n$$

is a ground instance of a rule in Π *and* $\{B_1, \ldots, B_n\} \subseteq \mathcal{S}$*, then the constraint:*

$$\ell \leq \Sigma_{\phi(C_i, \mathcal{S})=1 \wedge C_i \mapsto P} p_i \leq u$$

is in CONS($\Pi, \mathcal{S}$)*.*

2. $\Sigma_{\phi(C_i, \mathcal{S})=1} p_i = 1$ *is in* CONS($\Pi, \mathcal{S}$)*.*

EXAMPLE 3.3. *Consider the "political" example shown in Example 2.1 and suppose the state in question is* $\mathcal{S}_0 = \{democrat, promise\_tax\_cut\}$*. Let* C_1 *and* C_2 *be the two COAs from Example 3.1. The constraints in this case are:*

$0.2 \leq p_2 \leq 0.3$*;*
$0.3 \leq p_2 \leq 0.6$*;*
$0.3 \leq p_1 \leq 0.9$*;*
$\Sigma_{\phi(C_i, \mathcal{S})=1} p_i = 1$.

The following result shows that a SOMA-program Π is consistent w.r.t. a given state iff the set CONS($\Pi, \mathcal{S}$) of constraints has a solution.

4. MAXIMALLY PROBABLE COAS

When a reasoning agent builds a SOMA-program about another opposing agent, his goal is to determine what the opponent will do in a given state. We define below, the lower probability that the opponent will take a given course of action.

DEFINITION 4.1 (LOWER PROBABILITY OF A COA). *Let* Π *be a* SOMA*-program and* $\mathcal{S}$ *be a state. The* lower probability*,* low(C_i) *of course of action* C_i *is defined as follows:*

1. low(C_i) $= 0$ *if* $\phi(C_i, \mathcal{S}) = 0$*, i.e.* C_i *is not valid.*
2. *Otherwise,* low(C_i) $=$ **minimize** p_i ***subject to*** CONS($\Pi, \mathcal{S}$).

The *upper probability*, $up(C_i)$ of course of action C_i can be defined in a similar manner. Note that for an given course of action C, we cannot *exactly* determine the probability that the opponent will do C. This is true even if all rules in Π have a point probability in the head — this is because our framework does not make any simplifying assumptions (e.g. independence) about the probability that the agent will perform action b, given that he will do action a.

The following example explains what happens in the situation of our political example and the state $\mathcal{S}_0$ introduced in example 3.1.

EXAMPLE 4.1. *As we saw in example 3.3, there are 4 constraints that must be satisfied at once in order to find an interpretation that satisfies all rules in* Π *with respect to* $\mathcal{S}$*. It can be seen that* low(C_1) $= 0.3$*, and* low(C_2) $= 0.3$*. Similar calculations lead to* up(C_1) $= 0.9$*, and* up(C_2) $= 0.3$.

5. RELATED WORK

We are not aware of much work on probabilistic agent reasoning - the work that exists focuses on agents that are reasoning about an uncertain state and trying to figure out how best to act in such a situation. However, there are a few relevant works that we build on top of.

[Gmytrasiewicz and Durfee 1992] have developed a logic of knowledge and belief to model multiagent coordination. Their framework permits an agent to reason not only about the world and its own actions, but also to simulate and model the behavior of other agents in the environment. In an earlier paper, they show how one agent can reason with a probabilistic view of the behavior of other agents so as to achieve coordination.

[Dix et al. 2000] develop a model of probabilistic agents on top of legacy code — in their framework, an agent reasons about uncertainty in its state. However, there is no uncertainty about what the agent will do. A subsequent work by these authors [Dix et al. 2005] extends this framework to the case of temporal probabilistic reasoning agents that can reason with uncertainty about when certain things will be true in the state.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a hybrid logic/statistical language called SOMA within which a reasoning agent can represent knowledge about the probability that an opponent agent may take one or more actions. We have developed a formal model theoretic semantics for such agents.

In the full version of this paper, we make several additional contributions: first, we prove that checking consistency of SOMA-programs is NP-hard. We have identified syntactic fragments of SOMA-agents for which consistency checking is polynomial. We show that checking if a given course of action has a probability exceeding a given threshold is NP-hard. We have developed an exact algorithm to find an MPCOA and several heuristic algorithms that can approximate the most probable course of action very quickly and with high degrees of accuracy.

7. REFERENCES

DIX, J., NANNI, M., AND SUBRAHMANIAN, V. 2000. Probabilistic agent reasoning. *ACM Transactions of Computational Logic 1,* 2, 201–245.

DIX, J., KRAUS, S., AND SUBRAHMANIAN, V. 2004. Heterogeneous temporal probabilistic agents. *ACM Transactions of Computational Logic 5(3).*

GMYTRASIEWICZ, P. AND DURFEE, E. 1992. A Logic of Knowledge and Belief for Recursive Modeling. In *Proceedings of the 10th National Conference on Artificial Intelligence.* AAAI Press/MIT Press, San Jose, CA, 628–634.

HANKS, S. AND MCDERMOTT, D. 1994. Modeling a dynamic and uncertain world I: Symbolic and, probabilistic reasoning about change. *Artificial Intelligence 65(2)*, 1–55.

NG, R. AND SUBRAHMANIAN,V.S. 1993. A Semantical Framework for Supporting Subjective and Conditional Probabilities in Deductive Databases. *Journal of Automated Reasoning 10 (2)*, pps 191–235.

Agent planning and search

Decentralized Planning under Uncertainty for Teams of Communicating Agents

Matthijs T.J. Spaan
University of Amsterdam
Kruislaan 403, 1098 SJ
Amsterdam, The Netherlands
mtjspaan@science.uva.nl

Geoffrey J. Gordon
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
ggordon@cs.cmu.edu

Nikos Vlassis
University of Amsterdam
Kruislaan 403, 1098 SJ
Amsterdam, The Netherlands
vlassis@science.uva.nl

ABSTRACT

Decentralized partially observable Markov decision processes (DEC-POMDPs) form a general framework for planning for groups of cooperating agents that inhabit a stochastic and partially observable environment. Unfortunately, computing optimal plans in a DEC-POMDP has been shown to be intractable (NEXP-complete), and approximate algorithms for specific subclasses have been proposed. Many of these algorithms rely on an (approximate) solution of the centralized planning problem (i.e., treating the whole team as a single agent). We take a more decentralized approach, in which each agent only reasons over its own local state and some uncontrollable state features, which are shared by all team members. In contrast to other approaches, we model communication as an integral part of the agent's reasoning, in which the meaning of a message is directly encoded in the policy of the communicating agent. We explore iterative methods for approximately solving such models, and we conclude with some encouraging preliminary experimental results.

General Terms

Theory, Algorithms

Keywords

Artificial intelligence, planning under uncertainty, cooperative multiagent systems, decentralized POMDPs

1. INTRODUCTION

Planning refers to the process of computing a conditional sequence of actions that fulfill a given task as well as possible. It is a crucial part of any intelligent agent; human, robot or software agent alike. We are interested in planning under various forms of uncertainty for cooperative multiagent systems. First of all, the agent might be uncertain regarding the exact consequence of executing a particular action. Furthermore, the agent's sensors may be noisy or provide only a limited view of the environment. As the agent is part of a team, a third source of uncertainty are its teammates, as it should consider their actions as part of its own decision making. In this paper, we study scenarios in which agents have the ability to communicate, but bandwidth is limited and communication has a certain cost. The range of possible application domains is broad, including cooperative robotics [6], sensor networks [13], distributed space applications [4], communication networks [16] and supply chain management [7].

Single-agent planning under uncertainty has been formalized in the partially observable Markov decision process (POMDP) framework [12]. A POMDP defines stochastic models to capture the uncertain effects of actions and the fact that sensors are imperfect and have a limited scope. In particular, the transition model governs how the system's state is influenced by actions, and the observation model stochastically relates states to observations. Solving a POMDP provides the agent with an optimal plan for a given task, but it is intractable to compute exact solutions in general (as it is known to be PSPACE-complete [17]). However, recent years have seen much progress in the development of approximate methods [19, 20, 23, 24].

Planning for a team of agents proves to be significantly harder than planning for just a single agent. In order for the team to act optimally, each agent not only has to consider the consequences of its own action, but at the same time also the actions its teammates will execute. As each agent does not know what the other agents observe, it will be hard to predict their actions. Decentralized POMDPs (DEC-POMDPs) generalize the POMDP framework to cooperative multiagent settings, but solving finite horizon DEC-POMDPs has been proven to be NEXP-complete [2], a considerable jump in complexity from the single agent case. Besides brute-force enumeration and evaluation of all possible policy tuples, only one algorithm is known for computing optimal solutions [11], which in practice is limited to very small problems. As the general setting is very hard to solve optimally, research has focused on heuristic methods, which often also make additional assumptions on the planning domain [27, 8, 1, 10, 18].

Allowing agents to communicate with each other can increase team performance, as it mitigates two sources of uncertainty: a message from a teammate can either provide information about the state of the system, or about the teammate's intentions. In particular, if one assumes that

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

the agents have unlimited free access to a noise-free communication channel, the multiagent planning problem reduces to a single-agent one: each agent shares its observations with all teammates, which allows each one of them to select the optimal joint action, and execute its part of the joint action vector. However, although optimal, such an approach is not desirable for several reasons. First of all, in reality a communication channel is likely to be bounded in bandwidth, and sending a message will have a particular cost. Second, the resulting single agent POMDP will still be hard to solve, as the size of the problem description grows exponentially in the number of agents. Therefore, we choose to tackle the problem in a decentralized manner, in which each agent considers only its local state plus some shared uncontrollable state features, instead of the cross-product of the local states of all agents and the uncontrollable state space. Furthermore, we consider communication as an integral part of the system, and model it directly in the DEC-POMDP, in contrast with other approaches in which communication decisions are governed by a separate algorithm [8, 5, 14, 22].

We propose a decentralized model for tackling cooperative multiagent planning problems, which incorporates a communication channel. In contrast to most work on distributed POMDPs, we do not define an explicit communication language, but instead treat the semantics of communication as part of the optimization problem. We will present an iterative method for computing a joint policy for the team in a decentralized fashion. Given a set of fixed policies for all agents but agent i, we convert the DEC-POMDP to a POMDP from i's perspective which incorporates the expected contribution to the joint team reward of the other agents' policies. We propose a heuristic method for computing a communication policy and we conclude with encouraging preliminary experimental results in two domains.

2. PLANNING FOR MULTIAGENT TEAMS

We will start by reviewing the cooperative multiagent planning problem from a decision-theoretic perspective. First we will briefly introduce partially observable Markov decision processes (POMDPs), a rich mathematical framework for single-agent planning under uncertainty. Next, we will look at decentralized POMDPs which constitute a multiagent extension of the POMDP model, and at existing exact and approximate methods for solving them.

2.1 Partially observable Markov decision processes

A partially observable Markov decision process (POMDP) models the repeated interaction of an agent with a stochastic environment, parts of which are hidden from the agent's view [12]. The agent's goal is to perform a task by choosing actions which fulfill the task best. At each time step the environment is in a state $s \in S$, and the agent executes an action $a \in A$, which will move the environment to a new state s' according to a stochastic transition model $p(s'|s,a)$. As a by-product the environment will generate a scalar reward signal $R(s,a)$, which defines the agent's task. Next the agent receives an observation $o \in O$ which is stochastically related to s' according to $p(o|s',a)$, but in general will not uniquely identify s'. As such, the agent is uncertain about the true state of the system, which is often modeled as a probability distribution over all states. The probability distribution, or belief $b(s)$ in POMDP terminology, summarizes all relevant history and is updated using Bayes' rule:

$$b^{ao}(s') = \frac{p(o|s',a)}{p(o|a,b)} \sum_{s \in S} p(s'|s,a)b(s), \qquad (1)$$

where $p(o|a,b) = \sum_{s' \in S} p(o|s',a) \sum_{s \in S} p(s'|s,a)b(s)$ is a normalizing constant.

The goal of the agent is to compute an optimal policy $\pi(b)$ that maps beliefs to actions. Optimal is defined as maximizing a certain performance measure, for instance the expected discounted sum of rewards $E\big[\sum_{t=0}^{h} \gamma^t R\big]$, where h is the planning horizon and γ is a discount rate, $0 < \gamma < 1$. Without going into details of exact POMDP solving, it suffices to say that computing an optimal policy is intractable in general [17]. In recent years a number of approximate algorithms including PBVI [19], BPI [20], HSVI [23], and Perseus [24] have been proposed, which allow for solving larger POMDP domains by, e.g., exploiting structure in the domain.

2.2 Decentralized POMDPs

A straightforward extension of the POMDP framework to cooperative multiagent settings is called the decentralized POMDP (DEC-POMDP) model [2]. Instead of a single agent we now consider a set of agents, which share the same reward function. If the agents have individual (and differing) reward functions, the model is known as a partially observable stochastic game (POSG) [11]. A DEC-POMDP is defined as follows:

- $I = \{1, \ldots, n\}$ is a set of n agents.
- S is a finite set of states.
- A_i is a finite set of actions for agent i.
- O_i is a finite set of observations for agent i.
- $p(s'|s,a)$ is the joint transition model which defines the probability of jumping to state $s' \in S$ given that the team executed joint action $a \in A_1 \times \ldots \times A_n$ in $s \in S$.
- $p(o|s,a)$ is the joint observation model which defines the probability that after taking joint action a and ending up in state s results in joint observation $o \in O_1 \times \ldots \times O_n$.
- $R : S \times A_1 \times \ldots \times A_n \rightarrow \mathbb{R}$ is the reward function which gives a scalar reward signal to the team.
- $b_i^0 \in \quad (S)$ is the initial state distribution.
- γ is a discount rate, $0 \leq \gamma < 1$.
- h is the planning horizon.

Contrary to the single agent case, in a DEC-POMDP it is not possible in general to compute a belief state as an agent only knows its local observation o_i but not the complete observation vector o, which is required for computing the belief update (1). As such, each agent chooses its action a_i at time t based on its policy $\pi_i : \times_{t-1}(A_i \times O_i) \rightarrow A_i$, i.e., it only considers its own history of actions taken and observations received until time t. The goal of the team of agents is to compute a joint policy $\pi = \{\pi_1, \ldots, \pi_n\}$ which maximizes the future discounted reward $V^\pi = E_\pi\big[\sum_{t=0}^{h} \gamma^t R\big]$. Unfortunately, computing an optimal policy π^* is NEXP-complete [2], and there is only one known algorithm for computing an exact solution (apart from brute-force), which we will describe next [11].

2.3 Solving decentralized POMDPs

Hansen et al. [11] present a dynamic programming operator for finite horizon POSGs, which allows for incremental construction of policies: starting from a solution for horizon t it computes a solution for horizon $t+1$. When the POSG is a DEC-POMDP, i.e., the agents share the same reward function, the algorithm will result in a set of policies optimal for a certain planning horizon. A belief is maintained over the cross-product of the state of system and the future conditional plans, or policy trees, of all other agents. However, the set of possible policy trees grows doubly exponential in the planning horizon, which renders a brute-force approach infeasible in practice. To combat the growth, at every iteration (very weakly) dominated policy trees are pruned. A policy tree for agent i is dominated if its removal does not decrease the value of any belief for agent i. However, even with pruning at every iteration the algorithm can solve only very small problems before running out of memory.

As solving a DEC-POMDP is intractable in the general case, methods have been proposed for approximately solving more restricted DEC-POMDP variations. Simplified models that have been studied include extra assumptions regarding the observation model, in particular that an agent's observations are independent from those of its teammates [8] or that each agent can observe its own local state with full certainty, which reduces the problem to a decentralized MDP [27, 1, 10]. The latter assumption requires a factored state representation in which the state space is the cross-product of each agent's local state space and optionally a set of shared external state features which each agent can observe but not influence. Examples include time or any other external information relevant to the team's task. Given a factored state space one can further assume that the agents' transitions are independent [27, 1, 18], or impose a particular joint reward structure [1]. Apart from restricting the model one can also limit the expressive power of the solution method, for instance by only searching in the space of finite state controllers of a particular size [3, 25]. Another way of computing a locally optimal solution is to optimize one agent's policy while keeping the other policies fixed, and iterate over all agents until convergence [15, 5, 3].

A different dimension to tackle decentralized POMDP models is to assume the agents have a way to communicate with each other (which is not modeled in the DEC-POMDP). As we noted above, if a team of agents is allowed to communicate for free and without limitations, then each agent can broadcast at every time step its perceived observation to all of its teammates. As a result every agent knows the joint observation vector which it uses to update the joint belief of the team, based on which the next joint action is selected. As such, free communication reduces the distributed control problem to a centralized, single-agent one, which can be solved using standard POMDP solution techniques. However, in reality communication is often not free or unlimited, and modeling a team of agents in this way might not be desirable. Nevertheless, several methods treat the multiagent system as if communication were truly free, solve the large centralized POMDP, and execute the resulting joint policy in a distributed fashion while imposing communication constraints [5, 14, 22]. While executing the policy a limited form of communication is used to attempt to preserve coherent behavior of the team. Note that the communication decisions are not part of the centralized policy, but are made by a separate algorithm, which for instance monitors the uncertainty regarding the joint belief. We refer to [21, 9] for a detailed analysis of communication issues arising in DEC-POMDP settings.

In contrast, Xuan et al. [27] propose a model that incorporates the communication decision directly in the agent's policy. It adds a communication sub-stage to the decision process, in which an agent decides whether it will communicate with its teammates. In this way an agent can explicitly reason about communication, instead of relying on an independent instrument to handle communication issues. Extending this framework, Goldman and Zilberstein [8] present a formal model for decentralized control with communication decisions based on the DEC-POMDP model. In such decentralized models agents reason over their local state, instead of considering the state of all teammates. A decentralized model seems more appropriate for agents who cannot observe each other's state nor have free and unrestricted communication at their disposal. Decentralized models also do not suffer from the exponential growth in the size of the centralized state space when the number of agents increases. Limited communication abilities, however, can be exploited to improve the team's performance, by allowing agents to share information at a certain cost.

3. PROPOSED MODEL

In [8, 9] a framework for modeling cooperative multiagent systems as a decentralized POMDP with a global reward function is proposed. The agents reason explicitly over their communication decisions, but the observations of each agent are independent. We propose a different decentralized model which draws on [8, 9], but in which a communication channel is established by allowing agents to send messages as part of their action vector, and messages are received in the next time step as part of the recipients' observation vectors. In contrast to most work on distributed POMDPs, we do not manually define a communication language, but treat the semantics of communication as part of the optimization problem.

We refine the DEC-POMDP model as described in Sec. 2.2 as follows:

- The state space S factors into $n+1$ components: $S = S_0 \times S_1 \times \ldots \times S_n$, where S_i is the set of state features relevant to agent i and S_0 the set of external state features, which are shared by all agents but which cannot be controlled. A local state of an agent i is defined as $\bar{s}_i = (s_0, s_i)$, where $s_0 \in S_0, s_i \in S_i$.

- Σ_i is a finite set of possible messages, and $\_{}_i \in \Sigma_i$ denotes a message sent by agent i. Not sending any message is defined as sending the empty message $\_{}_i = \emptyset$.

- $A_i = A_i^d \times A_i^\sigma$ is the action set for agent i, where A_i^d is the set of domain level actions and A_i^σ the set of communication acts.

- $O_i = O_i^d \times O_i^\sigma$ is the observation set for agent i, where O_i^d is the set of domain level observations the agent receives through its sensors, and $O_i^\sigma = \Sigma_1 \times \ldots \times \Sigma_{i-1} \times \Sigma_{i+1} \times \ldots \times \Sigma_n$ the set of possible messages the agent can receive from any of its teammates.

- The joint transition model $p(s'|s,a)$ depends only on the domain level component of each agent's action (sending messages does not influence the state of the environment). We assume that the agents' transitions are independent, allowing us to factor the joint transition model:

$$p(s'|s,a) = p(s'_0|s_0) \prod_{i=1\ldots n} p(s'_i|s_i, a^d_i), \qquad (2)$$

where $s'_0, s_0 \in S_0$, $s'_i, s_i \in S_i$, and $a^d_i \in A^d_i$.

- The local observation model $p(o_i|\bar{s}_i, a_i)$ specifies the probability of receiving observation $o_i = (o^d_i, o^\sigma_i)$ after executing action a_i and ending up in $\bar{s}_i$.

- We define $p(o^\sigma_i|^-)$ as the probability that a message vector $^- = (\ _1, \ldots,\ _{i-1},\ _{i+1}, \ldots,\ _n)$ will be interpreted as a particular observation o^σ_i, which can be used to model noise in the communication channel.

- The reward function $R(s,a)$ gives the team of agents a scalar reward at each time step for taking joint action a in state s. It is defined as the sum of each agent's local reward function and the joint reward of the team:

$$R(s,a) = \sum_{i=1\ldots n} R_i(\bar{s}_i, a_i) + R(s_0, \ldots, s_n, a_1, \ldots, a_n). \qquad (3)$$

The local reward $R_i(\bar{s}_i, a_i)$ is defined as the sum of the reward for the domain level action $R_i(\bar{s}_i, a^d_i)$ and the communication cost $R_i(\bar{s}_i, a^\sigma_i)$:

$$R_i(\bar{s}_i, a^\sigma_i) = \begin{cases} 0 & \text{if } \ _i = \emptyset \\ r_c < 0 & \text{otherwise} \end{cases} \qquad (4)$$

- Initial belief b^0_i, discount rate γ, and planning horizon h as in Sec. 2.2.

In the remainder of this paper we shall limit our discussion to two agents, but our methods and experiments generalize to more than two agents. Broadcast communication is used, and we will focus on noise-free communication, fixing $p(o^\sigma_i|^-)$ to be the identity matrix.

3.1 Example: Multiagent Heaven or Hell

To give some intuition on the range of problems our model captures we will now discuss a very simple example application. The "Multiagent Heaven or Hell" problem (inspired by [26]) is a scenario in which two agents have to meet in one of two locations, one called "heaven" and the other one "hell", see Fig. 1. At the start of each trial heaven is positioned either in the bottom-left state ($s_0 = \textit{left}$) or in the bottom-right state ($s_0 = \textit{right}$) with equal probability, and hell will be located in the other state. If the agents meet in heaven the team receives a reward of $r_h > 0$, but meeting in hell is penalized with a reward of $-r_h$. However, each agent does not know the location of heaven or hell until it visits the "priest" state, in which it receives an observation indicating whether heaven is left or right. Visiting the priest results in an individual negative reward of $r_p < 0$ for the particular agent, and we assume that the reward obtainable in heaven makes visiting the priest a viable option, i.e., $r_h > -nr_p$.

If no communication is possible, or the communication cost ($-r_c$) is prohibitively large, the optimal policy could be for each agent to visit the priest, find out where heaven is located and go there. However, when communication is relatively cheap, i.e., $r_c > r_p$, more interesting scenarios become feasible. For instance, only agent i could ask the priest and inform agent j by sending it a message. This would save agent j the cost of visiting the priest and still be able to tell heaven from hell. However, instead of defining such semantics a priori, we would like the agents to learn when and how to communicate, and how to interpret incoming messages. In particular, the agents should optimize their use of communication with respect to the cost of sending a message r_c.

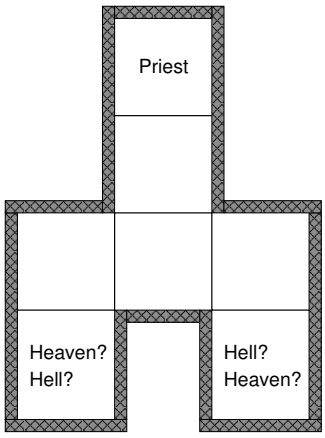


Figure 1: Multiagent Heaven or Hell environment. The priest (top state) can tell the agents where heaven is located (bottom-left or bottom-right state).

3.2 Belief tracking

In a general DEC-POMDP setting it is not possible for an agent to maintain an exact belief over the true state of the system, as it only receives its own part of the observation vector. As such, agent i has to reason about what observation agent j might have observed, but at the same time agent j is modeling agent i, which leads to the very high complexity class of solving DEC-POMDPs. As a result of the lack of independence between agents, only approximate beliefs can be computed. Alternatively, one can assume that agent i's observations are independent from those of agent j [15], which together with a common transition independence assumption precludes any use of direct communication in the DEC-POMDP model, as the agents cannot influence each other. In contrast, in our decentralized model the domain level component o^d_i of an observation $o_i = (o^d_i, o^\sigma_i)$ only depends on an agent's local state $\bar{s}_i$, but the communication component o^σ_i does depend on agent j, as follows

$$p(o_i|\bar{s}_i, a_i) = p(o^d_i, o^\sigma_i|\bar{s}_i, a_i) \qquad (5)$$

$$= p(o^d_i|\bar{s}_i, a_i) p(o^\sigma_i), \qquad (6)$$

where $p(o^\sigma_i)$ is the probability that the other agents send agent i a particular message vector in the previous time step, which we will approximate based on their policies. Consequently, we can only perform an approximate belief update, but it allows us to model a built-in communication channel.

Analogous to (1), agent i's belief b_i is updated as follows

when it takes action a_i and receives observation o_i:

$$b_i^{a_i o_i}(\bar{s}_i') = \frac{p(o_i^d|\bar{s}_i', a_i)p(o_i^\sigma)}{p(o_i^d, o_i^\sigma|a_i, b_i)} \sum_{\bar{s}_i \in (S_0 \times S_i)} p(\bar{s}_i'|\bar{s}_i, a_i) b_i(\bar{s}_i), \tag{7}$$

where $p(o_i^d, o_i^\sigma|a_i, b_i)$ is a normalizing constant. The probability that $b_i^{a_i o_i}$ will be the successor of b_i when agent i executes action a_i is defined as

$$p(b_i^{a_i o_i}|b_i, a_i) = p(o_i|b_i, a_i) \tag{8}$$

$$= \sum_{\bar{s}_i \in (S_0 \times S_i)} p(o_i|\bar{s}_i, a_i) b_i(\bar{s}_i). \tag{9}$$

Since individual beliefs can be defined through (7), local policies π_i can be defined as in a standard POMDP setting, i.e., as mappings $\pi_i : \quad (S_0 \times S_i) \rightarrow A_i$ from local beliefs b_i to actions. The only term not yet defined in (7) is $p(o_i^\sigma)$: the probability that agent i receives a message $\quad_j$ and interprets it as observation o_i^σ. It is defined by two factors: the probability that agent i receives $\quad_j$ as o_i^σ, as defined by the channel noise model $p(o_i^\sigma|\quad_j)$, times the probability that agent j actually sends $\quad_j$. The latter probability is estimated by computing the set of possible beliefs $\{b_{j,t}^\sigma\}$ in which agent j sends $\quad_j$ according to π_j, and summing the probabilities $p(b_{j,t})$ that any of those beliefs in $b_{j,t} \in \{b_{j,t}^\sigma\}$ will occur:

$$p_{t+1}(o_i^\sigma|\pi_j) = p(o_i^\sigma|\quad_j) \sum_{b_{j,t} \in \{b_{j,t}^\sigma\}} p(b_{j,t}), \quad \text{with} \tag{10}$$

$$\{b_{j,t}^\sigma\} = \{b_{j,t}|(a_j^d, a_j^\sigma) = \pi_j(b_{j,t}), a_j^\sigma = \quad_j\}. \tag{11}$$

where $p(b_{j,t})$ is the probability that agent j is in belief $b_{j,t}$ at time t. The following equation computes $p(b_{j,t})$ as a recursion from time step 0, at which we know agent j is in its initial belief b_j^0 with probability 1:

$$p(b_{j,t}) = \begin{cases} 1 & \text{if } t = 0 \\ p(b_{j,t}|b_{j,t-1}, \pi_j(b_{j,t-1}))p(b_{j,t-1}) & \text{otherwise} \end{cases} \tag{12}$$

which uses (9). The set $\{b_{j,t}\}$ of all possible beliefs which agent j could believe to be true at a particular time step is finite for any given t, as we have assumed finite models.

3.3 Communication

The previous section defined how each agent updates its belief and how j's policy affects i's belief update, and here we provide the intuition how this implements a communication channel. For a particular incoming message $\quad_j$ agent i can use its knowledge of π_j and the set of beliefs $\{b_{j,t}\}$ for agent j to compute the set $\{b_{j,t}^\sigma\}$ (11) from which agent j would send the particular message $\quad_j$. In this way, receiving message $\quad_j$ at time $t+1$ informs agent i about agent j's local state at time t:

$$p_{t+1}^{\sigma_j}(\bar{s}_j) = \sum_{b_{j,t} \in \{b_{j,t}^\sigma\}} b_{j,t}(\bar{s}_j) p(b_{j,t}), \tag{13}$$

combining (12) and (11). The crucial point is that, as agent j's local state $\bar{s}_j$ includes S_0, receiving message $\quad_j$ will provide agent i with information about s_0:

$$p_{t+1}^{\sigma_j}(s_0) = \sum_{s_j \in S_j} p_{t+1}^{\sigma_j}(\bar{s}_j) \quad \text{with} \quad \bar{s}_j = (s_0, s_j). \tag{14}$$

In summary, the fact that s_0 is shared by all agents allows for communication-based learning, by enabling an agent j to modify the observation model of some other agent i, via the term $p(o_i^\sigma)$ in (7).

Depending on the quality of agent j's communication policy, communication can improve team performance. For instance, if agent j communicates uniformly at random, agent i will not benefit from communication as the information content of the message is zero. However, in the Multiagent Heaven or Hell example described above, if agent j communicates $\quad_j = 1$ if it knows heaven is left, communicates $\quad_j = 2$ if it knows heaven is right, and does not communicate otherwise, then agent i can use $\quad_j$ to update its belief about the location of heaven in a useful manner and proceed to heaven without visiting the priest.

3.4 Discussion

When comparing with other recent approaches to planning for cooperative multiagent systems in partially observable environments, there are two main advantages to the proposed model. First of all, we treat communication as an integral part of an agent's reasoning, not as an add-on. In particular, we do not define a priori semantics for a message, but treat it as part of the optimization problem. The meaning of a particular message $\quad_i$ is defined by the situations in which agent i will send $\quad_i$. Of course, assuming a message set Σ_i with user-defined, high quality and fixed semantics simplifies the optimization problem, but we believe it is worthwhile to explore optimizing communication as part of (approximately) solving the DEC-POMDP.

The second feature of the proposed model is that it is decentralized, and does not require an (approximate) solution to the large single-agent POMDP resulting from truly free communication (as described in Sec. 2.3). The centralized state, action and observation spaces grow exponentially in the number of agents, rendering even approximate solutions infeasible. In our decentralized model, only the observation set for each agent grows with the number of agents, which can be limited by manipulating the size of possible messages Σ_i. One other possible solution to combat this exponential growth would be to assume that messages from different agents are conditionally independent given the state, which would allow us to factorize each agent's observation model. A factorized observation model induces a particular structure in an agent's POMDP (see next section), which can facilitate solving the POMDP.

We trade off exploiting an (approximately) optimal centralized solution for a potentially more scalable decentralized setting, in which each agent only considers it own local state (and some shared uncontrollable state features). Furthermore, a decentralized framework is more natural for modeling agents who cannot observe each other's state nor have truly free communication available.

4. SOLUTION METHOD

After proposing a decentralized model for planning for a team of communicating agents, we will now present an approach for computing a plan in our model. We propose to compute a joint policy for the team of agents in an iterative fashion [15, 5, 3]. We start with a (randomly selected) policy π_i for each agent i. We fix the policy for all agents except for one, our protagonist agent i. From i's perspective we can treat the other agents as part of the environment, and i can build a POMDP model of this environment, given that it knows the policies of the other agents. For the re-

sulting POMDP model it computes a policy using a standard (approximate) POMDP solver which replaces its current policy π_i. We then move on to the next agent and repeat the procedure from its perspective. This process is iterated until some convergence criterion is met, at which time the joint policy will have reached a local maximum. Multiple random restarts are necessary in general to reach good performance. We will solve the POMDP models using PERSEUS [24]. It computes a policy in the form of a value function $V_i : \quad (S_0 \times S_i) \rightarrow \mathbb{R}$, which for every belief estimates the amount of future discounted reward the agent can obtain.

4.1 Computing an agent's POMDP model

A crucial component is building the POMDP model from the known problem description and the policies of all teammates. In this POMDP model the reward function for agent i should not only consist of its local reward function R_i but also the expected contribution of the joint reward function R. For instance, in the Multiagent Heaven or Hell example described in Sec. 5.1, if agent i predicts that agent j's policy directs j to the heaven location, agent i's local reward function should report that if i goes to heaven too, it can receive the joint reward of r_h. We summarize the influence of the other agent by computing two statistics: $p(a_j|\bar{s}_j)$ is the probability agent j will execute a_j in $\bar{s}_j$, and $p(\bar{s}_j)$ is the probability that agent j will be in a particular state $\bar{s}_j$. We approximate these probabilities by simulating the belief tree for the team, given the pair of policies $\{\pi_i, \pi_j\}$. The root of the belief tree at $t = 0$ is the starting belief b_i^0 of each agent i. We compute the set of possible $t + 1$ beliefs for each agent by propagating each of its beliefs at time t:

$$\{b_{i,t+1}\} = \{b_{i,t}^{a_i o_i} | a_i = \pi_i(b_{i,t}), o_i \in O_i\}. \qquad (15)$$

In order to keep the memory requirements in check in general pruning will be necessary. From the belief sets at time step $\{b_{i,t}\}$ and the probability that each of the beliefs will be encountered (according to (12)) we can approximate the required statistics $p(a_j|\bar{s}_j)$ and $p(\bar{s}_j)$.

We use these statistics to define the local reward function R_{i,π_j} for agent i, which takes into account the expected contribution from agent j's policy π_j:

$$R_{i,\pi_j}(\bar{s}_i, a_i) = R_i(\bar{s}_i, a_i) + \sum_{\bar{s}_j \in (S_0 \times S_j), a_j \in A_j} R(s_0, s_i, s_j, a_i, a_j) p(a_j|\bar{s}_j) p(\bar{s}_j), \qquad (16)$$

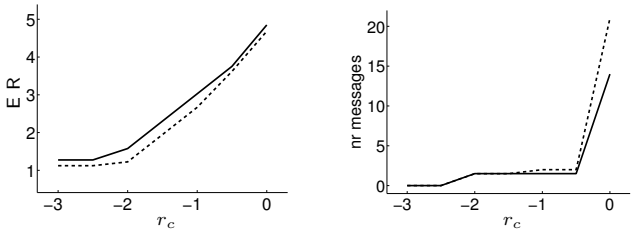
with $(s_0, s_i) = \bar{s}_i$. From the same statistics the observation model for agent i can be computed, which will encode the information contained in incoming messages from agent j, as we described in Sec. 3.3. To complete the POMDP model, the transition model for agent i is given by

$$p(\bar{s}_i'|\bar{s}_i, a_i) = p(s_0', s_i'|s_0, s_i, a_i^d, a_i^\sigma) \qquad (17)$$

$$= p(s_0'|s_0) p(s_i'|s_i, a_i^d). \qquad (18)$$

4.2 Learning to communicate

The iterative scheme described above is able to react to any incoming communications, which are incorporated in the local reward model and exploited by the POMDP solver. However, it will not learn from scratch to send any messages, as the other agent's policy has not yet learned how it can benefit from incoming messages. Therefore we begin by equipping each agent with a fixed heuristic communication policy and iteratively optimize the agents' domain level component of their policies to exploit the incoming messages. Next we aim to optimize the communication component of each agent's policy with respect to the communication reward $r_c < 0$. In particular, initially all agents could at each time step send the last domain level observation they received, and we could improve team performance by only sending messages that are useful.



(a) Reward. (b) Nr. of messages.

Figure 2: Multiagent Heaven or Hell results. Solid lines indicate results for 2 agents, and dashed lines show results for a team of 3 agents. Plot (a) shows the expected discounted reward (E R) obtained for different communication rewards (r_c), and figure (b) plots the average number of messages sent.

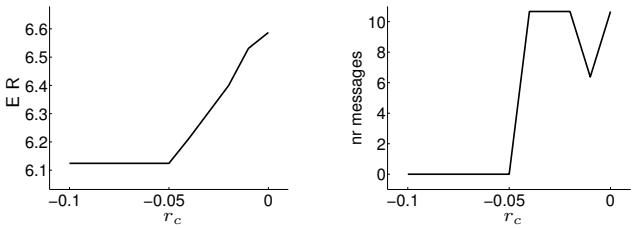
A message is useful when it conveys some information which the other agent can use to improve team performance. As the meaning of a message is defined over s_0, the entropy of $p^{\sigma_j}(s_0)$ (14) measures the information content of a particular message. However, if a message has low entropy does not necessarily mean it is useful to send. One proposal for testing its value is to modify each agent's local reward model (16) to prefer communicating low entropy messages to not sending them, but only at states in which it would send the messages under the initial heuristic communication policy. This ensures the other agent will know how to respond to the incoming message. Next we repeat the process of iterative optimization, allowing the agents to adjust to the new communication behavior.

5. EXPERIMENTS

We will present experimental results in the Multiagent Heaven or Hell domain and in a game of tag, in which agents have to cooperate in order to tag an opponent. In these experiments we included the time step of the system in s_0, which allows for a single message to have a different meaning, depending on which time step it was sent. Adding time to s_0 requires no extra assumptions, but does allow the agents to more closely correlate their behavior. Providing correlation for agents in a DEC-POMDP has been shown to improve team performance [3].

5.1 Multiagent Heaven or Hell

First we will consider the Multiagent Heaven or Hell domain as introduced in Sec. 3.1. The environment contains seven grid cells ($|S_i| = 7$) and the shared state s_0 is the cross-product of the time index and {"heaven is in bottom-left grid cell","heaven is in bottom-right grid cell"}, where we set the horizon of the problem to 15. Each agent has a domain level action set $A_i^d = \{$*north, east, south, west,*



(a) Reward. (b) Nr. of messages.

Figure 3: Pack Tag results. Plot (a) shows the expected discounted reward (E R) obtained for different communication rewards (r_c), and figure (b) plots the average number of messages sent.

stay in place}, where the first four are movement actions that transport the agent one grid cell in the corresponding direction, and the last action is a no-op which has no effect on the agent's location. Each agent can observe whether there is a wall to the left or to the right of it, and in the priest state it can observe the location of heaven, resulting in six possible observations O_i^d ={*left, right, both, neither, heaven-left, heaven-right*}. Meeting in heaven by all agents is rewarded by $r_h = 10$ and meeting in hell by $-r_h$. Visiting the priest incurs a negative reward $r_p = -2$, every movement action a reward of -0.1, and the discount factor γ is set to 0.95. Both the transition and observation model are deterministic, and the agents always start in the center grid cell (the cell at the T-intersection).

Fig. 2 shows results obtained by using the local reward modification scheme described in Sec. 4.2, for teams of 2 and 3 agents. First, we see in Fig. 2(a) that the team performance increases when the penalty for communication decreases, i.e., r_c becomes less negative. In all cases one agent i will visit the priest, but it will only send a message indicating the location of heaven if the communication cost is low ($r_c \geq -2$), see Fig. 2(b). Otherwise the other agent(s) will go to one of the heaven/hell locations, and agent i will only join them if it is indeed the correct location. When communication is free, performance is best and a large number of messages are communicated, as one would expect. On the other extreme, when communication is expensive ($r_c = -3, -2.5$), the performance remains stable.

5.2 Pack Tag

The "Pack Tag" problem models a game of tag, in which two robots have to tag an opponent robot [19, 5]. The game is played on a $k \times k$ grid, with one robot starting in the bottom-left corner and the other one in the top-right corner. At the beginning of each game the opponent is positioned uniformly at random in one of the grid cells and remains stationary. The two robots have to occupy the same grid cell as the opponent and execute the *tag* action simultaneously in order to tag the opponent. The robots can move in any of the four compass directions, and their actions are deterministic. The two robots have a sensor which detects the opponent with full certainty when the opponent is sharing the same grid cell. Each robot receives a null observation otherwise, but remains perfectly localized due to the known starting position and noise-free motion. Each action incurs a reward of -0.1 and successfully tagging the opponent results in a reward of 10. The agents have to tag the opponent in 10 time steps, and γ was set to 0.95.

As in Sec. 5.1 we varied the communication cost while employing the local reward modification method, and using a grid size $k = 3$. Fig. 3 shows that when the communication cost is high, $r_c \leq -0.05$, the agents converge to a non-communicating policy pair. They both search the grid, and when one of them sees the opponent it waits for the other to join it before jointly tagging it. However, when communication is cheaper the agents communicate when they observe the opponents, which indicates to the other agent the location of the opponent. Here the benefit of including time in s_0 is manifested, as receiving the same message at a different time step has a different meaning: agent i knows agent j's policy, and therefore knows j's search pattern, and in this way knows in what grid cell agent j observed the opponent when j send its message. After receiving an informative message an agent abandons its search pattern and moves directly to its teammate's position, where they jointly tag the opponent. The policy pair computed for $r_c = -0.01$ has converged to better local maximum, which uses a more efficient form of communication.

6. CONCLUSIONS

In this paper we studied the problem of computing a plan for a team of agents. We considered a general problem setting, in which agents inhabit a stochastic environment that is only partially observable to them. The agents have the capability to communicate, but sending messages has a certain cost and the available bandwidth is limited. Decentralized partially observable Markov decision processes (DEC-POMDPs) formalize such multiagent planning problems for groups of cooperating agents. We presented a decentralized model for tackling these kinds of planning problems, which incorporates a communication channel. Decentralized models potentially scale better and are a more natural paradigm for modeling agents that cannot observe each other's state. Contrary to other recent approaches, we treat communication as an integral part of the model, and model it directly in the DEC-POMDP. We proposed an approximate iterative method for computing policies in the proposed model, and obtained encouraging preliminary experimental results in two domains.

As future work we would like to examine possibilities of simultaneous learning of communication-derived POMDPs for two or more agents, which could potentially improve upon the alternating-maximization paradigm. Furthermore, we would like to experiment on problems with more agents, which are too large to be solved by centralized methods. For problems that are still solvable by centralized algorithms, we intend to compare the centralized solution to the decentralized one, to gain more insight in how optimality is traded off for more efficient computation.

Acknowledgments

M.T.J. Spaan and N. Vlassis are supported by PROGRESS, the embedded systems research program of the Dutch organization for Scientific Research NWO, the Dutch Ministry of Economic Affairs and the Technology Foundation STW, project AES 5414. G.J. Gordon's work was supported in part by NSF grant EF-0331657.

7. REFERENCES

[1] R. Becker, S. Zilberstein, V. Lesser, and C. Goldman. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*, 22:423–455, 2004.

[2] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[3] D. S. Bernstein, E. A. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized POMDPs. In *Proc. Int. Joint Conf. on Artificial Intelligence*, 2005.

[4] M. B. Dias, A. Stentz, and D. Goldberg. Market-based multirobot coordination for complex space applications. In *Proc. of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2003.

[5] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

[6] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Game theoretic control for robot teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.

[7] M. S. Fox, M. Barbuceanu, and R. Teigen. Agent-oriented supply-chain management. *International Journal of Flexible Manufacturing Systems*, 12(2,3), 2000.

[8] C. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, 2003.

[9] C. V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. In *Journal of Artificial Intelligence Research*, volume 22, pages 143–174, 2004.

[10] A. Guo and V. Lesser. Planning for weakly-coupled partially observable stochastic games. In *Proc. Int. Joint Conf. on Artificial Intelligence*, 2005.

[11] E. Hansen, D. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *Proc. of the National Conference on Artificial Intelligence*, San Jose, 2004.

[12] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[13] V. Lesser, C. L. Ortiz, and M. Tambe, editors. *Distributed sensor networks: A multiagent perspective*. Kluwer Academic Publishers, 2003.

[14] R. Nair, M. Tambe, M. Roth, and M. Yokoo. Communication for improving policy computation in distributed POMDPs. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

[15] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proc. Int. Joint Conf. on Artificial Intelligence*, Acapulco, Mexico, Aug. 2003.

[16] J. Ooi and G. Wornell. Decentralized control of a multiple access broadcast channel: Performance bounds. In *Proc. of the 35th Conference on Decision and Control*, pages 293–298, 1996.

[17] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[18] S. Paquet, L. Tobin, and B. Chaib-draa. An online POMDP algorithm for complex multiagent environments. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, 2005.

[19] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. Int. Joint Conf. on Artificial Intelligence*, Acapulco, Mexico, Aug. 2003.

[20] P. Poupart and C. Boutilier. Bounded finite state controllers. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[21] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[22] M. Roth, R. Simmons, and M. Veloso. Decentralized communication strategies for coordinated multi-agent policies. In A. Schultz, L. Parker, and F. Schneider, editors, *Multi-Robot Systems: From Swarms to Intelligent Automata*, volume IV. Kluwer Avademic Publishers, 2005.

[23] T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *Proc. of Uncertainty in Artificial Intelligence*, 2004.

[24] M. T. J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[25] D. Szer and F. Charpillet. An optimal best-first search algorithm for solving infinite horizon DEC-POMDPs. In *European Conference on Machine Learning*, 2005.

[26] S. Thrun. Monte Carlo POMDPs. In *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.

[27] P. Xuan, V. Lesser, and S. Zilberstein. Communication decisions in multi-agent cooperation: Model and experiments. In *Proc. of the Fifth Int. Conference on Autonomous Agents*, 2001.

Integrating Parallel Interactions into Cooperative Search

Efrat Manisterski [1]
[1] Department of Computer Science
Bar-Ilan University
Ramat-Gan 52900, Israel
maniste@cs.biu.ac.il

David Sarne [2]
[2] Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA
sarned@eecs.harvard.edu

Sarit Kraus [1, 3]
[3] Institute for Advanced Computer Studies
University of Maryland
College Park, MD
sarit@cs.biu.ac.il

ABSTRACT

In this paper we incorporate autonomous agents' capability to perform parallel interactions into the cooperative search model, resulting in a new method which outperforms the currently used ones. As a framework for our analysis we use the electronic marketplace, where buyer agents have the incentive to search cooperatively. The new search technique is quite intuitive, however its analysis and the process of extracting the optimal search strategy are associated with several significant complexities. These difficulties are derived mainly from the unbounded search space and simultaneous dual affects of decisions taken in different world states. We provide a comprehensive analysis of the model, highlighting, demonstrating and proving important characteristics of the optimal search strategy. Consequently, we manage to come up with an efficient modular algorithm for extracting the optimal cooperative search strategy for any given environment. A computational based comparative illustration of the system performance using the new search technique versus the traditional methods is given, emphasizing the main differences in the optimal strategy's structure and the advantage of using the proposed model.

Categories and Subject Descriptors

K.4.4 [**Computing Milieux**]: Computers And Society —*Electronic Commerce*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence —*Multiagent systems*

General Terms

Algorithms, Design, Economics, Performance, Theory

Keywords

parallel, cooperative search, search cost

1. INTRODUCTION

Coalition formation is well recognized as a key process in a multi-agent systems, mostly desirable in environments where a group of agents can perform a task more efficiently than any single agent can [10]. In recent years many coalition formation models have been suggested, for various domains [18, 5], particularly for electronic commerce [19, 20, 16]. In the latter context, the most common coalition concerned is a coalition of buyers, derived mainly from the potential of obtaining volume discounts [19, 15] and the ability to search cooperatively for market opportunities in a more efficient manner [16].

The cooperative search incentive derives principally from the existence of search costs found in MAS, reflecting the resources (not necessarily monetary), that need to be invested/consumed while searching for opportunities in the environment (e.g. searching for an opportunity to buy a product in the context of the electronic marketplace) [16]. The scenario of having search costs is common in MAS where the agent needs (for its decision making process) immediate information concerning market opportunities. Given the richness of opportunities and the dynamic and open nature of these environments, central mechanisms are usually incapable of supplying such information with the level of completeness and accuracy required by the agent, certainly not without a cost. Thus the agent needs to spend some of its resources on search related activities. Despite the reduction in the magnitude of these search costs in the electronic commerce era, the continuous growth in the number of retailers and virtual stores over the Internet, followed by a phenomenal increase in the number of opportunities available, makes the overall search cost an important parameter affecting buyers' search strategy [4, 9, 16, 1].

By forming a coalition and delegating the search task to a representative agent (acting on behalf of the coalition[1]), the cooperative search offers the advantage of sharing, reusing and re-allocating opportunities among the coalition members (e.g. exploiting opportunities which would have been discarded otherwise if each of the agents would have performed an alternative separate search) [16]. Nevertheless, the process of forming and maintaining the coalition induces some overhead, derived mainly from the communication and coordination activities [15], thus the representative agent should set its search strategy in a cost/effective manner.

A classic example of the above in traditional markets is the procurement management officer of a corporation. Instead of having each individual in the cooperation spend time and resources on locating its specific requested supplies, the task is delegated to the procurement management officer. Here, in addition to the price discounts obtained for aggregated demands of identical items, the procurement management officer becomes highly updated with the

[1]As discussed in detail in [16], the use of a representative agent is a common and efficient means for executing the coalition's goal. Furthermore, given the option of side-payments the overall utility maximization strategy taken by the representative agent is always the preferred one by all coalition members (i.e. no conflict of interests), regardless of the pre-set coalition's payoff division protocol.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

different offerings and specific supplies available by the different merchants in the markets. As a result the cost of locating the right deal for each request becomes significantly smaller (in comparison to the equivalent search conducted by each of the individuals).

The basic concepts by which a representative agent should manage the cooperative search, including an analysis and computational means for extracting its optimal search strategies are given in [16]. Nevertheless, the assumption used in that model for constructing the representative agent's strategy is that the representative agent interacts with one seller agent at a time as part of its search process. Such an assumption ignores an inherent strength of autonomous agents, which is their capability to efficiently interact with several other agents in parallel (enhanced by their improved communication capabilities and their ability to process an enormous amount of information in a short time, compared to people). The parallel interaction is preferable when an agent's search cost combines fixed components (e.g. operational costs) and/or non-linear dependency on the number of interactions maintained (e.g. advantage of size). In such cases the adoption of the parallel technique by the representative agent suggests a reduction in the average cost per interaction with seller agents. While the integration of parallel interactions technique into a single search process is quite intuitive, its integration into a coalition's search is not trivial at all. The major difficulty derives from the fact that different coalition members may have heterogeneous multi-attribute utility functions. In this paper we supply a comprehensive analysis of the parallel cooperative search model and present an algorithm which can facilitate the calculation process of the coalition strategy. The new searching technique results in at least as good (and in many environments significantly better) expected utility for the coalition.

Similar to the model introduced in [16], we apply the multi-attribute utility theory (MAUT) [8], to analyze preferences with multiple attributes in our agent based search mechanism. This enables a set of preferences to be represented by a numerical utility function. We consider the agents to be heterogeneous, each having its own utility function. The model is general, though several specific implementation aspects relating to the B2C (Business-to-Consumer) market, where sellers can supply almost any demanded volume, and the C2C (Consumer-to-Consumer) market, where sellers offer single items for sale, are emphasized. Based on the proposed analysis, the representative agent can calculate its optimal strategy given the utility functions of the coalition members and the specific environment in which it operates.

Notice that among the three basic stages that are common to all coalition formation models [14, 19]: coalition structure generation (where the agents form/join the coalition), executing the coalition task, and dividing the generated value among the coalition members, our focus is on finding the optimal search strategy for the coalition, given its structure and the opportunity distribution. As suggested in [16], the representative agent operates in its environment alongside many other agents that represent coalitions differing in their size, their members' utility functions and the products they are seeking. These other coalitions, as well as the different individual utility functions play an important role when studying the stability of a coalition and issues of revealing the true utility function (truth telling). The analysis of these important issues is based on the ability to properly derive the coalition's utility given any specific self structure (i.e. number of agents it represents and their reported, not necessarily true, utility functions) and the environment it is operating in. This paper aims to supply this functionality, laying the foundations and enabling research of many of the important aspects of coalition formation given above in the context of cooperative search (truth telling, stability, payoff division, etc.).

The main contributions of this paper are threefold: First, we formally model and analyze the parallel cooperative search problem of agents operating in a costly environment. This model is a general search model and can be applied in various domains in addition to the electronic marketplace that is being used as a framework for our work. Second, we show that in many environments the parallel cooperative search outperforms the traditional search strategies (either when each agent searches by itself or when using a cooperative sequential search). Furthermore, we draw attention to scenarios where traditional cooperative search is proven to be non-beneficial, however parallel cooperative search is a favorable technique. Finally, we supply an algorithm that facilitates the calculation of the coalition's optimal strategy, and significantly reduces the complexities associated with the attempt to extract this strategy from an appropriate set of equations.

In the following section we address relevant multi-agent and parallel search literature. The parallel cooperative search model assumptions and a formal description are given in sections 3 and 4, respectively. In section 5, we explore the unique characteristics of the representative agent's optimal strategy when using cooperative parallel search, leading to the introduction of an efficient algorithm for extracting it. In section 6 we demonstrate the unique properties of the parallel search method using as a reference two of the classical search models that our suggested model generalizes. We conclude and suggest directions for future research in section 7.

2. RELATED WORK

In many scenarios autonomous agents in multi-agent environments may cooperate in order to perform tasks. The need for cooperation may arise either when an agent is incapable of completing a task by itself or when operating as a group can improve their overall performance [3, 10, 19]. Group based cooperative behavior can be found in various domains, such as solving complex optimization problems [18], military and rescue domains [5], e-business applications [19, 20] and many more. In the electronic market domain, most authors focus on coalitions formed to obtain volume discount [19, 20]. Additional coalition formation models for the electronic marketplace consider extensions of the transaction-oriented coalitions into long-term ones [3], and for large-scale electronic markets [10]. Traditionally, the majority of this research effort has focused on issues concerning optimal division of agents into disjoint exhaustive coalitions [14, 20], division of coalition payoffs [20] and enforcement methods for interaction protocols. Only a few authors have considered the coalition's problem of determining its strategy in the electronic commerce domain, once the coalition is formed [7]. Nevertheless, other than in [16], none of the proposed models have considered a coalition's search in a costly environment, and in particular none of them (including [16]) have made use of the agent's capabilities to maintain parallel interactions.

The problem of a searcher operating in a costly environment, seeking to maximize his long term utility is addressed in classical search theory ([11, 12], and references therein). The three main search models that can be found in the literature are the fixed sample size model, the sequential model and the variable sample size model. In the fixed sample size model, introduced in [17], the searcher draws a single sample where all observations are taken simultaneously. In the sequential search model [11] the searcher draws one observation at a time, allowing multiple search stages. The third search model [2, 6, 13] suggests a combined approach in which several observations may be made at any period. This latter method, which outperforms the other two, is in fact the single agent's equivalent to our cooperative search model considered in this paper. In addition to the establishment of some general properties of a search rule, most of the papers mentioned above considered

mainly issues of uncertainty associated with the availability of the inspected offers [6], recall and fallback utilities [13], and the effect of a final decision horizon on the search strategy [2]. Nevertheless, the focus of the above works from the search theory domain is on a single agent's search, and the analysis of a cooperative search is lacking. As shown in [16], the transition from a single agent search to a cooperative search concept as the one we base our model on, is not trivial and encapsulates many complexities resulting in a different strategy structure. In the cooperative search model, the representative agent needs to take into consideration the affect of new opportunities found on any subset of utility functions associated with the different coalition members. Though the process of extracting the optimal strategy is very different from the one used by a single agent that tries to maximize a simple or an aggregated utility function with a single opportunity.

3. THE MODEL

We base our model description and formulation on the definitions given in [16] and extend them to reflect agents' parallel search capabilities. We consider an electronic marketplace where numerous buyer and seller agents can be found. Each agent is interested in buying or offering to sell a well defined product. A product can be offered by many different seller agents under various terms and policies (including price). We assume that while buyer agents are ignorant of individual seller agents' offers, they are acquainted with the overall distribution of opportunities (defining an opportunity as the option to buy the product under specific terms and policies) in the marketplace.

Assuming there are no central mechanisms or mediators which can supply the agents with full immediate information concerning current market opportunities, they need to search for appropriate opportunities to buy their requested product. Throughout the search the buyer agents locate seller agents and learn about their offers by interacting with them. Each buyer agent evaluates opportunities using its own multi-attribute utility function. Buyer agents may have heterogeneous preferences and thus the utility from a given opportunity differs according to the evaluating buyer agent.

In its most basic form, each buyer agent searches in such a way that it interacts with several sellers in parallel at each stage of its search thus learns about a new set of opportunities. Based on the agent's evaluation of the utility that can be gained from each opportunity in the set, the agent makes a decision whether to exploit any of the opportunities it encountered throughout its search (i.e. buy from any of the sellers) or resume its search in a similar method. A decision of resuming search is always accompanied with the number of parallel interactions to be executed next.

The search activity is assumed to be costly [4, 9, 16, 1]. For each search stage in which the buyer agent locates, interacts and evaluates seller agents, the process induces a search cost. This cost is a function of the number of parallel interactions initiated and maintained by the agent. The search cost structure is principally a parameter of the market's liquidity and volatility, and thus shared by all buyer agents operating in the specific marketplace. Recognizing the benefits of a cooperative search, buyer agents, interested in similar products or interchangeable products, may form coalitions [16]. Any coalition formed will always be represented by a representative agent, which conducts its search on behalf of the coalition in a similar method (encountering sellers and accumulating new opportunities). The representative agent's search cost increases both as a function of the number of parallel interactions it forms and the number of buyer agents it represents[2]. We assume a buyer agent's utility from a given opportunity may be interpreted into monetary terms. Thus the utilities are additive and the total search utility can be obtained by subtracting the search cost that the process induces from this value.

As part of its search process, the representative agent needs to set a strategy for determining, given any set of known opportunities, whether to terminate or resume its search. In the latter case, the agent also needs to determine the number of parallel interactions to be used in the next search round. The optimal strategy is the one maximizing its expected total search utility (opportunities utility minus search costs). Given the representative agent's goal of maximizing the overall coalition utility, its decision is not influenced by the payoff division protocol, nor by coalition stability considerations, but rather influences these two factors [16]. Any of the agents' pre-determined portion of the coalition's utility will increase in its absolute value along with the increase of the net coalition utility, thus the overall utility maximization strategy is the preferred strategy by all agents at every stage of the search.

4. PROBLEM FORMULATION

Let $B = (B_1, B_2, ..., B_k)$ be the set of the attributes defining any of the potentially available opportunities in the market, where each attribute B_i can be assigned a value from the finite set $(b^i_{min}, ..., b^i_{max})$. An opportunity's type is defined by the vector $\overrightarrow{o_i} = (b_1, b_2, ...b_k)$, assigning a value b_i to each specific attribute B_i. We denote by O the space of potential opportunity types the coalition may encounter. The opportunity types' distribution in the marketplace, is denoted by the probability function $p(\overrightarrow{o})$.

We consider a coalition $A = \{A_1, A_2, ..., A_m\}$ of a general size, where A_j is the $j-th$ buyer agent in the coalition. Each buyer agent, A_j, evaluates different opportunities using a utility function $U_j : O \rightarrow R$, where $U_j(\overrightarrow{o})$ is the agent's utility from opportunity type $\overrightarrow{o}$. The search cost associated with having a coalition of m agents maintaining w simultaneous interactions with seller agents over a search round is denoted by the function $c(w, m)$, satisfying $\lim_{w\rightarrow\infty} c(w, m) = \infty$ and $\frac{dc(w,m)}{dw} > 0, \frac{dc(w,m)}{dm} > 0$.

The set of opportunities known to the representative agent at a given stage of its search is denoted by θ_{known}. It is possible to have several opportunities of the same type in θ_{known}, as similar opportunities can be offered by several seller agents or a single seller agent may offer opportunities of a specific type with a quantity greater than one unit.

5. ANALYSIS

5.1 The search strategy

Let Θ be the collection of all possible sets of opportunities. Consider a function $alloc : \Theta \rightarrow O^m$ that maps a given set of opportunities θ to the coalition members in A (i.e. an allocation)[3] in a way that the aggregated agents' utility with such allocation is maximized[4]. Let $alloc(\theta) = (\overrightarrow{y_1}, ...\overrightarrow{y_m})$, $\overrightarrow{y_i} \in (\theta \cup \{\emptyset\})$ be the allocation, resulting from operating the function $alloc$ over the set θ, where $\overrightarrow{y_i}$ denotes the opportunity associated with agent A_i and $y_i = \emptyset$ denotes that no opportunity was allocated to agent A_i. The immediate utility for the coalition if it terminates the search at the

[2]The reasoning for correlating the coalition's search cost with the number of members being represented is mainly associated with some coordination overhead. See [16] for details.

[3]In B2C markets the same opportunity may be allocated to more than one agent, while in C2C markets each opportunity is restricted to only one agent.

[4]If there is more than one allocation that maximizes the overall coalition utility then the function $alloc$ will always choose one of them according to a pre-defined ordering.

current point given a set of known opportunities θ_{known}, is denoted as $U^s(\theta_{known})$ and can be calculated as:

$$U^s(\theta_{known}) = \sum_{j=1}^{m} U_j(\overrightarrow{y_j}) \quad (1)$$

where $U_j(\emptyset) = 0, \forall j$.

Notice that up until this point, the world states space on which the agent needed to define its strategy (reflected by the set Θ) was potentially infinite. Nevertheless, since there are only m agents represented by the coalition, any additional occurrence of an opportunity of type o_i in θ_{known} beyond the first m occurrences can neither improve the overall coalition utility nor the representative agent's strategy. As a result, the representative agent can reduce the set of known opportunities according to which it determines its strategy to a subset, s, of θ_{known} where each opportunity of a given type appears at most m times. We refer to s as a $state$, and define S as the set of all potential states. Given the $state$ definition, we define a strategy as a function $x : S \rightarrow N$, where $x(s) = 0$ if the agent decides to terminate its search; otherwise $x(s)$ is the number of parallel searches the representative agent should maintain next, when in state s. We denote the optimal strategy by x^*.

We define $V(s, w)$ as the expected utility when using w parallel searches when in state s (assuming any search decision taken at a future state $s' \neq s$ will make use of the optimal number of parallel searches). The term $V(s, 0)$ denotes the immediate utility obtained, if the agent decides to terminate the search at state s, thus: $V(s, 0) = U^s(s)$. The value w ($w \in N$, $w \geq 0$) that maximizes the coalition expected utility $V(s, w)$, is denoted as $x^*(s)$:

$$x^*(s) = \arg\max_w V(s, w) \quad (2)$$

In order to formulate the appropriate equation for $V(s, w)$ (from which $x^*(s)$ can be derived) we make use of several additional notations and definitions. Consider a search round in which the representative agent interacts simultaneously with w seller agents, yielding a set $\theta_w = \{\overrightarrow{o_1}, ..., \overrightarrow{o_w}\}, \overrightarrow{o_i} \in O$ of opportunities. Let Θ_w be the collection of all w-sized sets of opportunities that can be produced in the environment the representative agent is operating. We denote by $p_w(\theta_w)$ the probability of encountering a specific set of opportunities θ_w, when maintaining w random interactions with seller agents.

We define $next\_s(s, \theta_w)$ as a function that returns the new state s' of a representative agent, upon encountering the set θ_w given its initial state s.[5] For a given number of simultaneous interactions, w, and a given state s, let Θ^s_w be the collection of all w-sized sets of opportunities, θ_w, that change the agent's current state (formally stated as: $\Theta^s_w = \{\theta_w | \theta_w \in \Theta_w \; and \; next\_s(s, \theta_w) \neq s\}$). We denote the complementary set of Θ^s_w by $\bar{\Theta}^s_w$ (the set that includes all w-sized sets of opportunities θ_w that does not change the agent's current state). The expected utility when using w parallel searches while in state s, $V(s, w)$, can now be expressed as ($\forall w > 0$):

$$V(s,w) = -c(w,m) + \sum_{\theta_w \in \Theta^s_w} p_w(\theta_w) V(s', x^*(s')) + \sum_{\theta_w \in \bar{\Theta}^s_w} p_w(\theta_w) V(s,w) \quad (3)$$

where $s' = next\_s(s, \theta_w)$. This is derived from the stationary nature of the problem - if no better state was reached, the search resumes using the same strategy, yielding the same expected utility. After some simple math manipulations of the above term we obtain:

$$V(s,w) = \frac{-c(w,m) + \sum_{\theta_w \in \Theta^s_w} p_w(\theta_w) V(s', x^*(s'))}{\sum_{\theta_w \in \Theta^s_w} p_w(\theta_w)} \quad (4)$$

[5]The set returned by $next\_s$ is the union of opportunities in s and θ_w, where each opportunity type appears at most m times.

Notice that in the case where no new better state can be reached, the denominator becomes zero, and $V(s, w) = -\infty$. This is quite straightforward as the representative agent infinitely maintains a costly search. Here, the coalition's optimal strategy is inevitably to terminate the search. This important characteristic will be used later for designing the proposed solution algorithms.

At this point, one may attempt to compute the coalition's strategy x^* by solving a set of equations of types 1, 2 and 4. Nevertheless, this solution approach is accompanied by many inherent complexities, derived from the structure of the equations. First, notice that equation 4 is a recursive equation and one needs to know the optimal strategy taken in future states s' when extracting the optimal strategy of a given state s. Second, the computation of $V(s, w)$ in equation (4) is exponential in the number of parallel searches, w, used (affecting the number of sets in Θ^s_w, both in the denominator and numerator). Last, according to the above formulation, the potential number of parallel searches that may be used is not bounded, thus reaching a local maximum does not guarantee that a higher utility cannot be obtained.

In the next subsection we present a comprehensive analysis of the problem, emphasizing some unique characteristics of the representative agent's optimal strategy. These findings lead to an algorithm with a polynomial complexity (in the number of potential interactions, w) for computing $V(s, w)$ (which is the key component for computing $x(s)$).

5.2 Analysis

We begin our analysis by facilitating the state definition in a way that allows us to consider simple divisions of the search space into improving and non-improving areas with regard to the coalition's utility. For this purpose we introduce the concept of equivalence between different sets of opportunities within the context of cooperative search. We say that two sets of opportunities $\theta', \theta'' \in \Theta$ are equivalent sets $\theta' \equiv \theta''$, if the following hold: (a) $U^s(\theta'') = U^s(\theta')$; and (b) $U^s(\theta' \cup \theta) = U^s(\theta'' \cup \theta)$ for any set of opportunities $\theta \in \Theta$ that the agent may encounter in the future. For any two equivalent sets θ', θ'', the representative agent is indifferent to knowing the opportunities in θ' and the opportunities in θ''. This is because any set of opportunities the representative agent will encounter in the future results in a similar utility thus the overall coalition's utility will be the same in both cases. Moreover, since the agent's decisions are merely determined by the overall coalition utility and in both cases similar utilities are reached with similar probabilities, the agent will use the same number of parallel interactions for both sets.

Notice that according to the definition above $equivalent$ is a transitive relation ($\theta' \equiv \theta''$, $\theta'' \equiv \theta''' \rightarrow \theta' \equiv \theta'''$). Moreover, $\theta' \equiv \theta''$ implies that $(\theta' \cup \theta) \equiv (\theta'' \cup \theta), \forall \theta \in \Theta$. Given an allocation $\ell = (\overrightarrow{y_1}, ..., \overrightarrow{y_m})$ of a set θ, we denote the set of opportunities that appear in ℓ by $\{\ell\}$.

THEOREM 1. *Any set of opportunities θ is equivalent to the set of opportunities returned by the function $alloc(\theta)$. Formally stated: $\theta \equiv \{alloc(\theta)\}$.*

The above theorem is certainly non-trivial, particularly when considering the C2C market, where the optimal allocation is permutation based.

An immediate result obtained from theorem 1 above is that the representative agent's strategy is affected only by the subset of θ_{known} defined by $\{alloc(\theta_{known})\}$. Therefore we can now redefine $state$ to be the set of opportunities that are members in $alloc(\theta_{known})$. This latter definition significantly simplifies our analysis and enables us to suggest an efficient algorithm for extracting x^*. It is

notable that the computation method used by the function $alloc$ is market-dependent. While in B2C markets the function assigns each agent with the opportunity that maximizes its utility, $\vec{y_j}^* = \arg\max_{\vec{y} \in \theta} U_j(\vec{y}), j = 1, ..., m$, in C2C markets $alloc$ can be computed by solving a maximum weighted matching in a bipartite graph[6].

Given the analysis above, the set of all possible states, S, is given by $S = \{s | \exists \theta \in \Theta \; where \; s = \{alloc(\theta)\}\}$. Also, we can now denote by $S_s = (s_1, ...s_{|S|})$ the states constituting S ordered by their utilities, where s_1 is the state with the highest utility in S. Notice that now the function $next\_s(s, \theta_w)$ simply returns the set of opportunities in $alloc(s \cup \theta_w)$.

The above changes in the definition of a $state$ does not affect the definition of Θ^s_w and $\bar{\Theta}^s_w$. We denote by $p^{stay}(s, w)$ the probability the agent stays at the same state s after conducting w parallel searches. This can be calculated as the probability of having none of the encountered w opportunities change the representative agent's state:

$$p^{stay}(s, w) = (p^{stay}(s, 1))^w = (\sum_{\{\vec{o}\} \in \bar{\Theta}^s_1} p(o))^w \quad (5)$$

The term $1 - p^{stay}(s, w)$ can now be used as a better structured representation of the element $\sum_{\theta_w \in \Theta^s_w} p_w(\theta_w)$ that appears in equation (4). Similarly, we may consider the use of $V^{new}(s, k)$, defined as the coalition's expected utility obtained by potentially reaching new states (e.g. different than s) after maintaining k parallel interactions, while using the optimal strategy $x^*(s')$ for each new state s'. The term $V^{new}(s, k)$ does not take into account the cost associated with the current k interactions, however it does consider the search costs associated with any further search stages, originated in new (better) states. Notice that if the agent stays in state s during all k searches $v^{new}(s, k)$ is equal to zero. The term $V^{new}(s, w)$ is actually a representation of the element $\sum_{\theta_w \in \Theta^s_w} p_w(\theta_w) V(s', x^*(s'))$ in equation (4). Therefore, equation (4) can now be formulated as:

$$V(s, w) = \frac{-c(w, m) + V^{new}(s, w)}{1 - p^{stay}(s, w)} \quad (6)$$

The use of $V^{new}(s, w)$ in the above equation is a primary concept used in the algorithm suggested in the next section. Nevertheless, in order to extract $x^*(s)$ it is essential to supply the representative agent with an upper bound, w^s_{max}, for the optimal number of parallel searches to be used when in state s. The following proposition suggests an efficient bound for $x^*(s)$.

PROPOSITION 5.1. *For each state, s_i, an upper bound, $w^{s_i}_{max}$, to $x^*(s_i)$ can be calculated using $w^{s_i}_{max} = \lceil w \rceil$, where w is the solution of the following equation:*

$$c(w, m) = (U^s(s_1) - U^s(s_i))(1 - p^{stay}(s_i, w)) \quad (7)$$

The suggested bound is valid simply because for every value of w greater than $w^{s_i}_{max}$ the search cost associated with the following immediate search round is greater than any possible future improvement in the coalition's utility. Later on, we show that the above upper bound value is a by product of the main loop in our proposed algorithm, thus it does not even need to be directly calculated.

[6]For a set of opportunities $\theta \in \Theta$ found in the C2C market we construct a graph $G_\theta = (V_1, V_2, E)$, where each vertex of V_1 corresponds to an agent in A and each vertex in $V2$ corresponds to an opportunity $\vec{o} \in \theta$. An edge is connected between each agent A_j in V_1 and each opportunity $\vec{o}$ in V_2. The weight of such an edge is the utility for agent A_j from opportunity $\vec{o}$, $U_j(\vec{o})$. Here $alloc(\theta) = (\vec{y_1}, .., \vec{y_m})$, where $\{(A_1, \vec{y_1}), .., (A_m, \vec{y_m})\}$ is a maximum weighted matching in G_θ.

5.3 Algorithmic Approach

Recall that when attempting to solve the problem as a set of equations (see section 5.1) the potential number of parallel searches that may be used is unbounded. Furthermore even if we do manage to come up with a bound for $x^*(s)$ then the calculation of $V(s, w)$ (from which $x^*(s)$ can be derived) is exponential in the number of parallel interactions used, w. Our analysis, which is based on the restructuring of the different elements composing $V(s, w)$, allows us to bypass these two main complexities through the introduction of a finite algorithm with a polynomial computational complexity in w that will necessarily identify the optimal strategy for the representative agent.

In order to efficiently compute $V^{new}(s, w)$ in equation (4) we consider the w simultaneous interactions as w sequential interactions, associated with no search cost. This fully complies with the definition of $V^{new}(s, w)$ as given above (as the cost of the w interactions is already considered). The justification for the above representation method is given in the following lemma 1 which follows directly from theorem 1.

LEMMA 1. *A new state reached by obtaining a new set of opportunities is equivalent to a state reached by sequentially obtaining pairwise disjoint subsets of this set. Formally stated, given a set θ_w and any number of subsets $\theta^1_{w_1}, ..., \theta^r_{w_r}$, $\theta^i_{w_i} \subseteq \theta_w$, $\theta^i_{w_i} \cap \theta^j_{w_j} = \emptyset, i \neq j$, $\theta^1_{w_1} \cup ... \cup \theta^r_{w_r} = \theta_w$ and an initial state s, then the following holds:*

$$ns(s, \theta_w) \equiv ns(ns(...ns(ns(s, \theta^1_{w_1}), \theta^2_{w_2})..., \theta^{r-1}_{w_{r-1}}), \theta^r_{w_r}) \quad (8)$$

where $ns(s, \theta^i_{w_i}) = next\_state(s, \theta^i_{w_i})$. The proof is given in the full paper.

A specific case of the above lemma 1 is where each subset consists of a single opportunity. Thus the calculation of $V^{new}(s, k)$ can recursively rely on the values of $V^{new}(s', k-1)$ where s' represents any of the new states reached after obtaining one additional opportunity from O. Therefore in order to compute $V^{new}(s, k)$, we merely consider the expected utility after conducting one interaction from the planned k interactions. Here, with a probability of $p^{stay}(s, 1)$ the agent remains in the same state s, where the expected utility (having $k-1$ more interactions to go) is $V^{new}(s, k-1)$ (Recall that $V^{new}(s, 0) = 0$ according to the definition of this function that was given in the previous section). Otherwise, if a new state s' is reached after the single interaction, then there is the possibility of reaching further new states in the remaining $k-1$ interactions (taking these states' utility into consideration by the term $V^{new}(s', k-1)$) or with a probability of $p^{stay}(s', k-1)$ the agent will remain in the new state s' even after the additional $k-1$ interactions (in which case the utility is the one obtained by resuming the search from this state using the optimal strategy, $V(s', x^*(s'))$). The above description is encapsulated in the following recursive equation:

$$V^{new}(s, k) = p^{stay}(s, 1) V^{new}(s, k-1) + \quad (9)$$
$$+ \sum_{\{\vec{o}\} \in \Theta^s_1} p(\vec{o})(V^{new}(s', k-1) + p^{stay}(s', k-1) V(s', x^*(s')))$$

where $s' = next\_state(s, \overrightarrow{o})$. Notice that when repeating the calculation using the above equation with increasing k value, starting from $k = 1$, each iteration includes only a single unknown parameter, $(V^{new}(s, k))$.

The above analysis leads to the following algorithm for computing the agent's optimal strategy.

ALGORITHM 5.1. *An algorithm for computing the optimal search strategy x^*. **Input**: O - set of potential opportunity types in the*

market; $p(\vec{o})$ - opportunity types' probability function; m - coalition's size; $U_j()$, $j = 1, ..., m$ - coalition members' utility functions; $c(w, m)$ - search cost function;
***Output**: $x^*(s)\ \forall s \in S$ - the representative agent's optimal strategy.*

1. *Build the set of ordered states S_s;*
2. *For i=1 to $|S|$ {*
3. *Set $V(s_i, 0) = U^s(s_i)$ using equation 1*
4. *Set $w = 1$;*
5. *While $c(w, m) \leq (U^s(s_1) - U^s(s_{|S|}))$ {*
6. *Compute $V^{new}(s_i, w)$ using equation (9).*
7. *Compute $V(s_i, w)$ using equation (6); w++; }*
8. *Set $x^*(s_i) = \arg\max_{w' \in (0,..w)} V(s_i, w')$ }*
9. *Return($x^*(s_i)$, $i = 1, ..., s_{|S|}$)*

Notice that at each stage of its execution, algorithm 5.1 reuses components computed in earlier stages. For example, $V^{new}(s, w)$ appears both in the computation of $V(s, w)$ (equation (6)), in the computation of $V(s', w + 1)$ (equation (9)), where $\exists \vec{o}' \in O$ such that $next\_s(s', \vec{o}') = s$ and in the computation $V(s'', w+1)$ equation (9), where $\exists \vec{o}'' \in O$ such that $next\_s(s'', \vec{o}'') = s$. Storing in memory the result for each such computational element for the purpose of reusing it at later stages significantly improves the efficiency of the algorithm. This is accomplished by using two matrixes V and V^{new} of size $|S| \times (w_{max}^{s_{|S|}} + 1)$, where the corresponding $V(s, w)$ and $V^{new}(s, w)$ values are stored for each pair (s, w), representing a state and the correlated result for each number of simultaneous interactions used for calculation. Additionally, we store x^* values for reusing $x^*(s')$ in the computation of $v^{new}(s, w)$.

THEOREM 2. *Algorithm 5.1 returns the optimal strategy of the representative agent in a polynomial time of $w_{max}^{s_{|S|}}$.*

PROOF. In order to prove that the algorithm is polynomial in $w_{max}^{s_{|S|}}$, it is sufficient to prove that for each state s the computation of steps 3-8 are polynomial in $w_{max}^{s_{|S|}}$. Notice that the first elements being calculated are for state s_1 (i.e. the one with the maximum utility) according to the loop in step 2. Here, as explained in section 5.1, the expected utility from any strategy in which the search is resumed (i.e. using $w \geq 1$) is $V(s, w) = -\infty$ (formally, since $\Theta_w^{s_1} = \emptyset\ \forall w$ then $p^{stay}(s_1, w) = 1$ and $V^{new}(s_1, w) = ... = V^{new}(s_1, 1) = 0$, thus $V(s_1, w) = -\infty\ \forall w \in (1, .., w_{max}^{s_{|S|}})$). Thus the optimal strategy in state s_1 is to terminate the search, i.e. $x^*(s_1) = 0$ and $V(s_1, 0) = U^s(s_1)$. For any other state, s, when reaching step 6 of the algorithm, the agent has already computed both $V(s', x^*(s'))$ and $V^{new}(s', w)\ \forall w \in (0, ..w_{max}^{s_{|S|}})$ for all potential future new state s' of the state s. This is due to the fact that all future states of a state s appear before s in the set of ordered states S_s (either because having a higher utility, or equal utility yet sorted before s according to the function $alloc$). In addition, for any number of parallel interactions $w \geq 1$ the agent has already computed $V(s, w - 1)$. Therefore the computation time in step 6 does not depend on w. Then, when reaching step 7 of the algorithm, the numerator computation takes a constant time and the computation time of the denominator is polynomial in w. In step 8 the agent chooses the maximum value between $w_{max}^{s_{|S|}}$ values that have already been computed, therefore it is polynomial in $w_{max}^{s_{|S|}}$. The algorithm uses $w_{max}^{s_{|S|}}$ as an upper bound for the optimal number of interactions $x^*(s_i)$,$\forall i = 1, ..., |S|$ (Notice that for $s_{|S|}$ the term $(1 - p^{stay}(s, w))$ in equation 7 is 1). This bound is valid since according to equation 7 the value of $w_{max}^{s_i}$ decreases as a function of i. □

Further significant improvement of the above algorithm's performance can be achieved by calculating and using the specific upper bound, $w_{max}^{s_i}$, correlated with each state, s_i in step 5, according to equation 7. This might require at some points re-computation of $V(s_i, w)$ for w values that were not used in former algorithm execution stages; however the total number of calculations for each state s_i will significantly decrease[7].

Before we demonstrate some important properties of our search model, we wish to emphasize that the cooperative parallel search is a generalization of both the single agent parallel search and the cooperative sequential search models[8]. it is clear that the algorithm suggested in section 5.3 results in the same strategy as in the cooperative sequential search and in the single agent's parallel search, for the specific cases in which $\forall s_i\ w_{max}^{s_i} = 1$ parallel interactions or $m = 1$ agents, respectively.

Finally we wish to note the case where the agents are fully homogeneous (in terms of their utility functions) and operate in B2C markets. Here, we can prove that the optimal strategy of the representative agent is stationary (i.e. does not change according to the current state). Furthermore, we show that the stationary strategy characteristic holds not only for fully homogeneous agents but also when the agents have correlated preferences. We say that two agents A_i and A_j have correlated preferences if agent A_i prefers $\vec{o}'$ over $\vec{o}''$ if and only if agent A_j prefers opportunity $\vec{o}'$ over $\vec{o}''$, (i.e $\forall \vec{o}', \vec{o}'' \in O\ U_j(\vec{o}') \leq U_j(\vec{o}'') \leftrightarrow U_i(\vec{o}') \leq U_i(\vec{o}'')$).

THEOREM 3. *In B2C market if all agents have correlated preferences, the search strategy is based on a reservation value[9] U_{rv}. In such scenario the number of parallel interactions the representative agent uses according to the optimal strategy (in case it resumes the search) is fixed during the search ($\forall s \in S$ such that $U^s(s) < U_{rv}$ exists $x^*(s) = w_{fixed}$).*

PROOF. In the above scenario, the search can be represented as the search of a single agent with a utility function which equals the sum of the different agents' utilities, $U = U_1 + U_2 + ... + U_m$. In such case after terminating the search, each of the coalition members will always be assigned with the same opportunity. Therefore, the search strategy is reservation value based, and the search will be terminated upon reaching an opportunity with a utility exceeding some reservation value U_{rv}. Since the probability of reaching such opportunity does not depend on the agent's state, the number of parallel interactions used throughout the search is fixed. □

6. ILLUSTRATIVE EXAMPLES

Having an efficient means for calculating the representative agent's optimal strategy when using parallel search, we can now demonstrate some specific properties of this search method. As a reference we use the single agent's parallel search and the cooperative sequential search models introduced in [2, 6, 13] and [16], respectively.

When taking the cooperative sequential search as a baseline, in many cases the use of the cooperative parallel search results in a significantly improved joint expected utility for the agents constituting the coalition (in the worst case scenario, the optimal cooperative search strategy will yield equal utility). When considering

[7]The extent of the achieved improvement is highly correlated with the specific environment in which the representative agent is operating.

[8]Notice that in this context the single agent sequential search is a specific case of the single agent parallel search, where the agent interacts with a single seller agent at a time.

[9]See [16] for definition and discussion concerning reservation value based strategies in cooperative search.

single agents' parallel search, the cooperative parallel search has the potential to produce a significantly better expected utility (in comparison to the aggregated utility of the single searches) however, the decision of which of the two methods to use depends on the amount of coalition overhead costs induced by the cooperative search.

Figure 1 illustrates the above, depicting the expected utility per agent in any of the search methods[10] in the C2C market (left hand-side) and the B2C market (right hand-side) as a parameter of the similarity level (α) between the utility functions of the agents constituting the coalition. The results are based on the following environment, which was used originally for evaluating the performance of the cooperative sequential search [16]:

ENVIRONMENT 1. *A coalition of two agents, A_1 and A_2, searching for opportunities defined by two attributes, B_1 and B_2, where each attribute can have a value from the discrete range $(1, ..., 5)$ with an equal probability for each of the values. The agents are heterogeneous in respect to the way they evaluate each potential opportunity: agent A_1 is associated with the utility function $U_1(\overrightarrow{o}) = B_1 + B_2$ while agent A_2 with the utility function $U_2(\vec{x}) = 2(1-\alpha)B_1 + 2\alpha B_2$. Thus the parameter α indicates the level of agents' similarity/heterogeneity. The search cost of any single agent was taken to be $c(w, 1) = 0.15 + 0.05w$ and for a coalition $c(w, m) = c(w, 1)ln(m + 1)$, $\forall(m > 1)$.*

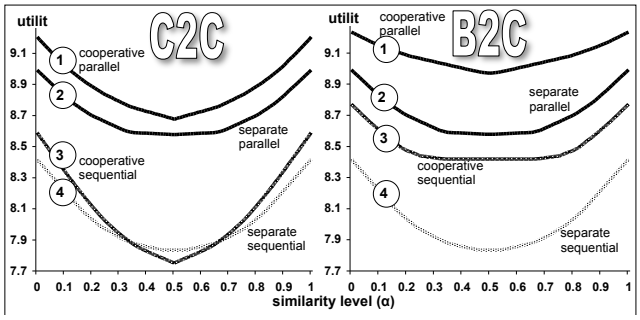


Figure 1: Average utility per buyer agent for different models in different markets

Curve 1 in each graph depicts the average expected utility when the two agents form a coalition, making use of the suggested parallel cooperative search. As expected our model outperforms the cooperative sequential search model (represented by curve 3) in terms of the expected utility for the agents. The other two curves describe the average expected utility of the agents when each is searching separately using the parallel (represented by curve 2) and the sequential (represented by curve 4) methods. In this specific environment the use of the cooperative parallel search also outperforms the single agent parallel search model though it is not always the case. Notice that the results obtained for the cooperative parallel search are consistent with a general characteristic of cooperative search [16] in a way that the usage of the method in the B2C market results with a better expected utility than in the C2C market. In the case of the separated single searches the market type does not affect the strategy structure nor the expected utility.

Figure 1 also reflects an interesting insight which contradicts an important strategy domination relationship found between single and cooperative sequential search techniques of fully homogeneous agents (i.e. with the same utility function, as in the case where $\alpha = 0.5$ in our environment). While for the sequential search the use of single agents' searches always outperforms cooperative search in C2C markets (when considering fully homogeneous agents) [16], here we have a true evidence that when parallel search is concerned, the cooperative search technique may outperform the aggregated result of the single homogeneous agents' search.

Next we introduce and make use of a simpler sample environment for demonstrating some additional properties of the cooperative parallel search.

ENVIRONMENT 2. *Similar to environment 1 above, except for the following changes: (1) each attribute has only two possible values $(1, 2)$ with an equal probability of $\frac{1}{2}$; (2) The utility functions used are $U_1(\overrightarrow{o}) = 1.9B_1 + 0.1B_2$ and $U_2(\overrightarrow{o}) = 0.1B_1 + 1.9B_2$. The search cost of any single agent was assumed to be $c(w, 1) = 0.5 + 0.05w$ and for a coalition $c(w, m) = c(w, 1) * ln(m + 1)$, $\forall(m > 1)$.*

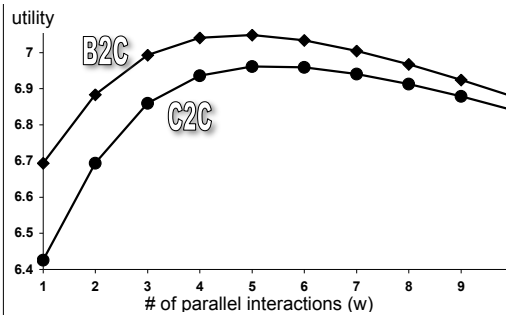


Figure 2: Coalition's overall utility as a function of w

Figure 2 depicts the expected coalition's overall utility with respect to the number of searches conducted at the beginning of the search (i.e. the first search stage, before the agent knows of any of the opportunities), assuming that in all the other states the agent uses the optimal number of parallel interactions, $x^*(s)$. Here, we can see the affect of two conflicting forces: as the number of parallel interactions the agent uses in this stage increases, the probability of associating a better opportunity with any of the two coalition members increases, however the overall search cost associated with the search stage increases. From the figure, we learn that the optimal number of parallel interactions to be used in this stage is $x*(\emptyset) = 5$.

An additional important characteristic of the cooperative parallel search we wish to emphasize concerns the number of parallel interactions used as part of the optimal search strategy along the search. While in a single agent's parallel search the search strategy is stationary (i.e. the number of parallel interactions used does not change along the search process) in our model the number of simultaneous interactions along the search that needs to be maintained depends on the agent's state (i.e. the set of known opportunities known to the representative agent). This is demonstrated in the directed acyclic graph (DAG) given in figure 3, which represents the feasible transitions from one state to other states in a B2C market described by environment 2. The state is determined according to the relevant set of known opportunities, correlated with the definition given in section 5.2. The relatively large number of parallel interactions used in the optimal strategy, allows the representative agent to reach all better states that are defined in this example. Notice that when reaching states $\{(1, 2), (2, 1)\}$ and $\{(2, 2), (2, 2)\}$ the optimal strategy of the agent is to terminate the search. For comparison purposes, notice that in any of the single agents' separate search (i.e. single agent parallel search model) the optimal strategy is to constantly use 4 parallel interactions (as long as resuming search).

[10]For the cooperative models the average expected utility per coalition member measure is used.

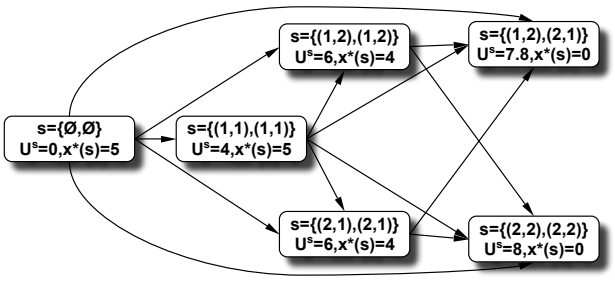


Figure 3: Potential transitions between states in a simple B2C market

7. DISCUSSION AND CONCLUSIONS

The capability of using parallel interactions as part of a search process is inherent in the infrastructure of autonomous information agents. When using the cooperative parallel search, the representative agent should use a new strategy, different in its structure in comparison to the optimal strategy used in the cooperative sequential search and in a single agent's parallel search. As expected, the use of the new model has the potential of significantly improving the coalition's expected utility as demonstrated in the previous section. Furthermore, we emphasize that the coalition's expected utility will never decrease when using our proposed mechanism in comparison to pure sequential cooperative search. This is mainly because in the case where maintaining more than a single interaction in some of the world states is not favorable the suggested algorithm will converge to one interaction at a time strategy, as used in the sequential cooperative search (which is a specific case of our model). Obviously if the search cost is linear and depends solely on the number of interactions being maintained, then the sequential cooperative search is the dominating strategy. Nevertheless, a scenario in which the representative agent's search cost combines additional fixed components and/or non-linear dependency on the number of interactions maintained are much more realistic [16]. In those scenarios the parallel cooperative search yields great benefits for searchers.

The novelty of the analysis given in this paper is twofold. First it supplies us with a better understanding of the opportunities space, dividing it into improving and non-improving areas, thus instead of having dual simultaneous dependencies between states we can now define a single directional dependency for each pair of states. Second, it supplies a bound for the optimum number of parallel interactions that the representative agent uses in each state in its optimal strategy. These two features allow us to overcome the main complexities associated with the attempt to solve the problem as a set of equations and offer a finite algorithm with a polynomial (rather than current exponential) computational complexity that inevitably reaches the optimal strategy. The model does not depend on attribute preferences or a specific cost function and with appropriate modification of the allocation function it can be generalized to additional markets other than C2C and B2C.

It is notable that the same considerations discussed in [16] concerning the potential for an actual implementation of cooperative search hold for the parallel (general) model presented in this paper. Similarly, future research in the area of parallel cooperative search should extend the scope of research to include additional topics associated with the coalition formation process such as the coalition stability, given the MAS settings and the division of payoffs (in terms of side payments) between the coalition members.

8. ACKNOWLEDGMENTS

This work was supported in part by NSF #IIS208608 and ISF #8008. In memory of Matat Rosenfeld-Adler.

9. REFERENCES

[1] Y. Bakos. Reducing buyer search costs: Implications for electronic marketplaces. *Management Sci.*, 42:1676–92, 1997.

[2] J. Benhabib and C. Bull. Job search: The choice of intensity. *J. of Political Economy*, 91:747–764, 1983.

[3] S. Breban and J. Vassileva. Long-term coalitions for the electronic marketplace. In *B. Spencer, ed., Proceedings of E-Commerce Applications Workshop*, 2001.

[4] S. Choi and J. Liu. Optimal time-constrained trading strategies for autonomous agents. In *Proc. of MAMA'2000*, 2000.

[5] M. Dias. *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*. PhD thesis, Robotics Institute, CMU, 2004.

[6] S. Gal, M. Landsberger, and B. Levykson. A compound strategy for search in the labor market. *International Economic Review*, 22(3):597–608, 1981.

[7] T. Ito, H. Ochi, and T. Shintani. A group-buy protocol based on coalition formation for agent-mediated e-commerce. *International Journal of Computer and Information Science (IJCIS)*, 3(1):11–20, 2002.

[8] R. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. John Wiley & Sons, New York, 1976.

[9] J. Kephart and A. Greenwald. Shopbot economics. *JAAMAS*, 5(3):255–287, 2002.

[10] K. Lermann and O. Shehory. Coalition formation for large scale electronic markets. In *Proceedings of ICMAS'2000*, pages 216–222, Boston, 2000.

[11] S. Lippman and J. McCall. The economics of job search: A survey. *Economic Inquiry*, 14:155–189, 1976.

[12] J. McMillan and M. Rothschild. Search. In Robert J. Aumann and Amsterdam Sergiu Hart, editors, *Handbook of Game Theory with Economic Applications*, pages 905–927. 1994.

[13] P. Morgan. Search and optimal sample size. *Review of Economic Studies*, 50(4):659–675, 1983.

[14] T. Sandholm, K. Larson, M. R. Andersson, O. Shehory, and F. Tohme. Coalition structure generation with worst case guarantees. *Artificial Int. Journal*, 111:209–238, 1999.

[15] D. Sarne and S. Kraus. The search for coalition formation in costly environments. In *Proc. of CIA*, pages 117–136, 2003.

[16] D. Sarne and S. Kraus. Cooperative exploration in the electronic marketplace. In *Proceedings of AAAI 2005*, pages 158–163, 2005.

[17] G. Stigler. The economics of information. *Journal of Political Economy*, 69(3):213–225, 1961.

[18] S. Talukdar, L. Baerentzen, A. Gove, and P. S. de Souza. Asynchronous teams: Cooperation schemes for autonomous agents. *Heuristics*, 4(4):295–321, 1998.

[19] N. Tsvetovat, K. Sycara, Y. Chen, and J. Ying. Customer coalitions in electronic markets. In *Proceedings of AMEC 2000*, pages 121–138, Barcelona, 2000.

[20] J. Yamamoto and K. Sycara. A stable and efficient buyer coalition formation scheme for e-marketplaces. In *Proc. of the Fifth International Conference on Autonomous Agents*, pages 576–583, 2001.

A Multi-Agent System that Facilitates Scientific Publications Search

Aliaksandr Birukou
Department of Information and Communication Technology
University of Trento - Italy
birukou@dit.unitn.it

Enrico Blanzieri
Department of Information and Communication Technology
University of Trento - Italy
enrico.blanzieri@unitn.it

Paolo Giorgini
Department of Information and Communication Technology
University of Trento - Italy
paolo.giorgini@unitn.it

ABSTRACT

It is very difficult for beginners to define and find the most relevant literature in a research field. They can search on the web or look at the most important journals and conference proceedings, but it would be much better to receive suggestions directly from experts of the field. Unfortunately, this is not always possible and systems like CiteSeer and GoogleScholar become extremely useful for beginners (and not only). In this paper, we present an agent-based system that facilitates scientific publications search. Users interacting with their personal agents produce a transfer of knowledge about relevant publications from experts to beginners. Each personal agent observes how publications are used and induces behavioral patterns that are used to create more effective recommendations. Feedback exchange allows agents to share their knowledge and virtual communities of cloned experts can be created to support novice users. We present a set of experimental results, obtained using CiteSeer as a source of information, that show the effectiveness of our approach.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering, Relevance feedback, Search process*; I.2.6 [**Artificial Intelligence**]: Learning—*Knowledge acquisition*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*

General Terms

Design, Experimentation

Keywords

Multi-agent system, implicit culture, learning agents, personal agents, publication search, CiteSeer, learning from observations

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

The Web is a huge source of information where it is possible to find publications and scientific literature. However, the majority of papers are usually irrelevant to individual researchers. Beyond conventional search engines, users need specific tools and methods for an effective use of all the available scientific resources. For example, special search engines for scholar articles, like GoogleScholar [7], have recently gained a lot of popularity among professionals and specialists. Online libraries like CiteSeer [9] and ACM digital library [16] for computer science are more and more used by researchers to search and download papers that are relevant to their scientific area. These services provide search and browsing facilities based on the list of references of the papers. However, for a novice researcher it is still very hard to determine which papers are really significant among those available. It should be very nice for a beginner to receive suggestions directly from experts of the field. An expert knows which papers are more influential, which papers were published at top conferences and in prestigious journals, what is the state-of-the-art and so on. The expert takes into account a lot of implicit factors when determines if a paper is relevant or not. But, unfortunately, we cannot assign a personal expert for each area of interests to every novice researcher. This calls for systems able to facilitate scientific publication search.

Among the proposed solutions for the scientific publication search, personalization systems [4, 8] focus on the need of a certain user, maintaining his/her profile and notifying him/her periodically or on-demand about papers related to the user's interests. The main goal of these systems is to filter information flow delivering to the user only items related to his/her interests. Differently, recommendation systems [1, 12] usually exploit the knowledge obtained from different users of the system to generate useful recommendations. Many recommendation systems [11, 12] use collaborative filtering algorithms [14] to compare similarity between users in order to recommend to the target user previously unseen items contained in the profiles of similar users. To some extent, currently developed systems mitigate search of the relevant literature for the person that is a novice in the field. However, they have several shortcomings. For instance, there exists a lack of expert knowledge, when all the users are novices. Pro-active recommendations sometimes turn into a heavy flow of potentially interesting items. Finally, most of the systems require explicit feedback, which is not always convenient and often discouraged by users.

In this paper we present a multi-agent system that facili-

tates scientific publications search. The architecture of the system is distributed and includes several multi-agent platforms. Each platform represents an organizational community of users with similar research interests. Alternatively, virtual communities of cloned experts can be created to facilitate the search of the papers relevant to a specific topic. As said above, we cannot assign an expert to each novice, but we can assign to a novice user an agent which has access to the expert's knowledge, represented in other agents. Users interact with their personal agents which cooperate and produce a transfer of knowledge about relevant publications from experts to beginners. Agents in our system are based on Systems for Implicit Culture Support [3] that learn behavioral patterns by observing how publications are used, namely which papers users cite, download, etc. Agents use learnt patterns and exchange feedback to produce personal recommendations more effectively. The system also provides an opportunity to share opinions and information contained in single bibliographies. It is possible to have an external information source, e.g. CiteSeer, to obtain citation graph and to download desired documents.

Although personal internet assistant agents have been developed for more then ten years [6, 10], the idea of using agents to support search of scientific publications is relatively new (see e.g. [8, 12]). Unlike most currently developed systems we use implicit feedback, avoiding extra work of the users. The framework used for the recommendation creation is general, being able to recommend not only papers, but expert in the research area as well. The main contribution of this work is the architecture of the MAS and its application to publications search. The use of Implicit Culture as a component of the system is justified by a number of applications.

The paper has the following structure. Section 2 motivates our approach by example. In Section 3 we describe the general architecture of the system, whereas experimental results are presented in Section 4. We discuss related work in Section 5 and conclude in Section 6.

2. MOTIVATION

In this section we illustrate the possible use of the system and show the role of agents in the system.

2.1 Motivating Example

Ali is a first year PhD student who studies Data Mining. His tutor Nick has recommended him to have a look on the paper "Privacy-preserving data mining" by Agrawal and Srikant. Ali submits the title and the authors of the paper to the system and his personal agent asks personal agents of his colleagues whether they have the paper in their bibliographies. In our case, the other PhD students, who are working in the same research group with Ali, are Javed and Julia. The platform containing the personal agents of these students is depicted in the upper left part of Figure 1. Julia is the PhD student of the second year and she already has the paper requested by Ali in her bibliography. Her personal agent provides this information in response to the query of Ali's personal agent. Ali receives complete information about the paper, including its location on the Internet, or in the local intranet, and comments from Julia, if, of course, she wants to give them. Now Ali can download the paper and then cite it in his paper. If the information about the target paper has not been found on the platform,

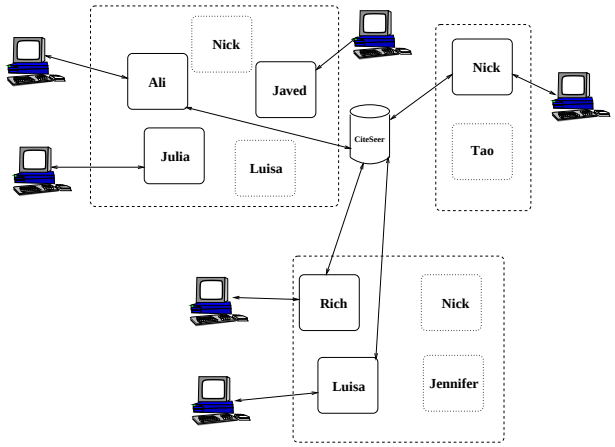


Figure 1: An example of several platforms with three ad hoc communities. Cloned expert agents are depicted with dotted squares. Julia, Ali and Javed also form an organizational community.

it is possible to contact other platforms or to query different information sources, e.g. CiteSeer.

To describe virtual community creation we introduce Luisa who is Ali's co-tutor, and works on Security. Ali wants to find relevant papers on the intersection of these two topics (Data Mining and Security), but his tutors are very busy. However, they agreed that Ali clones their personal agents. Ali's personal agent creates a virtual community cloning the personal agents of Nick and Luisa. Now, the corresponding expert agents run on the platform with Ali's personal agent. Each agent on the platform contains the information regarding behavior of its user, in particular Luisa's and Nick's citing, reading and browsing behavior. In our case, to find relevant papers on the intersection of Data Mining and Security, Ali's personal agent queries expert agents (Nick's and Luisa's) and shows to Ali the results ranking the papers recommended by both experts higher. The idea of combining Nick's and Luisa's results is to find papers that are relevant for them both. Moreover, being cloned and placed on another platform, the agents of Nick and Luisa adapt themselves to the local community, namely changing their recommendations towards the local common interests, or "culture".

In Figure 1 three ad hoc communities are depicted. For instance, Luisa and Rich work in the same research laboratory and consider research of Nick and Jennifer to be relevant to their current interests. Therefore cloned agents of Nick and Jennifer are running on the platform. Nick works with Tao and has his expert agent on the platform.

2.2 The Role of Agents

In our architecture, agents do the "dirty" work for their users. They perform bibliography maintenance, answer queries asked by other agents, and provide links where to download the desired papers. Among all this, they also represent the means of knowledge transfer from more experienced users to beginners.

Autonomy and mobility are capabilities that we exploit when we clone agents. Running in a new environment (the ad hoc community), the cloned agent uses the past experience of its user and its original community to support and

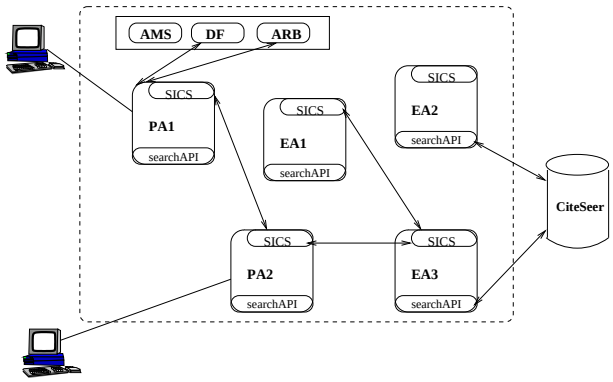


Figure 2: The architecture of a single system node. *Users* of the system node query their *personal agents* on the platform. Personal agents contact one another to find papers on a specified topic. They also use a search API to contact different information providers, e.g. CiteSeer. A SICS module inside the personal agent filters and re-ranks collected information. There are also *expert agents* running on the platform, which are the clones of the personal agents of the experts on the topic.

interact with the new community. The main idea here is that all the agents learn the "culture" of the community — i.e., knowledge about typical behavioral patterns of the members of the community — that can be used to select the relevant literature.

3. SYSTEM INTERNAL ARCHITECTURE

The system consists of several multi-agent platforms, or *system nodes*, which are able to communicate with one another. The system nodes are multi-agent systems implemented with Java Agent DEvelopment framework (JADE). A single platform runs on a dedicated computer and comprises one or several personal agents and several expert agents. The overall architecture of such a platform is depicted in Figure 2. It represents a community of users with similar research interests, e.g. personnel of a research lab or PhD students working in the same research field. Each user is assisted by a personal agent that helps to find relevant publications. The number of personal agents on the platform corresponds to the number of users of the system node. An expert agent is a clone of the personal agent of an expert user. Its task is to share its experience with other agents supporting them in their search. The platform also contains several auxiliary JADE agents. We describe system actors in more details in the next subsection.

The presence of several platforms where users are grouped by research interests helps us to handle scalability issues. Moreover, personal and expert agents running on the same platform and assisting users with similar interests are able to adapt to the specific needs of the community and to produce more valuable suggestions.

3.1 System Participants

- A *user* accesses the system using an html/php interface on the client side. It submits a query to the system, which can be either a *paper request* or an *information request*. In the case of the paper request, the user specifies the author(s) and the title of the paper. For the information request the user has to specify the topic he/she is interested in.
- A *personal agent* is a software agent running on the server side. The task of the personal agent is to assist the user in searching scientific publications. In case of a paper request it searches for the desired paper on the platform or queries an external information source using searchAPI. If the user submits an information request about a specific topic, then the personal agent starts the interaction with other personal and expert agents on the platform. The SICS module of the agent processes past actions, e.g. reading and citing, of its user and, partially (using feedback exchange), the actions of other users in order to find papers relevant to the given topic. The next subsection describes the SICS module in more detail.
- An *expert agent* is the clone of an expert's personal agent. It can represent, for instance, an author that is considered to be one of the top researchers in a certain area. In the best case, it represents a person and has information about how such a person usually browse the web searching for papers and which papers are eventually downloaded and read. In the worst (but anyway, quite good) case, the agent has the knowledge only about the use of citations by its user. This knowledge is extracted from the list of publications of the author. The papers that the expert agent suggests are considered to be of extreme importance and to form the state-of-the-art on the topic. It is also possible for the cloned agent to update its knowledge base, contacting the original personal agent.
- By an *external information source* we mean any database containing scientific publications and offering some kind of API to access them. For the experiments we used data from CiteSeer, but we could have chosen Google Scholar or any university or public library index.
- A *wrapper* is an agent which deals with those information sources that have no API. Typically, each information source requires a tailored wrapper dealing only with this particular source. The presence of wrapper(s) on the platform is optional.
- An *Agent Management System (AMS)* exerts a supervisory control over the platform. It provides the registration, the search, the deletion of agents and other services. It is an internal JADE agent running on every platform.
- A *Directory Facilitator (DF)* holds the list of services that are provided by an agent and provides a set of agents that offer a specific service. It provides agents with the IDs of other personal/expert agents. In our case both personal and expert agents have the service "PaperRecommender" that defines the agent's ability to answer queries about scientific articles.
- An *Agent Resource Broker (ARB)* provides a link between the platform and other platforms. Using this link, the agents can propagate requests of their users to different platforms. Contacting different system nodes

Table 1: The actions that can be observed by the system

action	object	attributes
participate	conference	paper, session_name, topic
publish	journal	paper, special_issue_title, topic
cite	paper	paper, topic
query		topic
download	paper	topic
view_details	paper	topic
reject	paper	topic

that correspond to different research groups it is possible to collect responses that represent several points of view.

3.2 Internal Agent Structure

Since expert agents are cloned personal agents, they have exactly the same structure. The agents have certain resources, e.g. database of past user actions or bibliography, and capabilities, in particular the searchAPI and the SICS module. A *bibliography* is collection of *bibitems*, each representing a single publication. It is stored as .bib file and can be used in LaTeX. A *searchAPI* is an API that agent uses to query an information source. Basically, it is a library that contains a set of functions to query the information source. It is dedicated to the specific source and is usually provided by the source owner.

3.2.1 SICS

SICS stands for System for Implicit Culture Support. It is responsible for recommendation creation process. It consists of the three modules: the *observer* that records users' actions, the *inductive module* that applies data mining techniques in order to induce a theory about these actions, and the *composer* that produces answers to the query by using the information saved by the observer and produced by the inductive module. The observer can partially collect information about actions of other users. It is possible due to feedback exchange which we describe in the next subsection. We refer the reader interested in more details of the SICS module to the paper of Blanzieri et al. [3].

Every *action* in our terminology has an *agent*, which executes the action, e.g. the user executing the action; an *object* which participates in the action, e.g. paper; and a set of *attributes*, which are the features that can be useful for the analysis of the actions, e.g. session name for the action of participating in a conference. In our application, the SICS module analyzes the actions presented in Table 1. An *agent* is always an author; therefore we omit it in the table. Every action is recorded being executed with respect to a certain topic. To determine topics related to an article, we can use the list of keywords given in the article. Furthermore, the keywords that were used to retrieve the article from an information source could be also considered as a list of topics, to which document is relevant.

We explain the information contained in the table in detail:

- *Participate.* An author can participate in a conference, publishing a paper in the conference proceedings. In order to observe this action it is just necessary to know the information about where the paper is published. We use a name of the conference session as an additional indicator of the topic of the paper.
- *Publish.* A paper can be published in a journal, related to a certain research field. Again, we observe this action if we have the information about where the paper is published. In case of publication in a special issue, the name of the special issue is an additional indicator of the topic of the paper
- *Cite.* This action indicates the papers cited by an author in his/her paper. This information is extracted from citation graph.
- *Query.* This action corresponds to an information request. Submitting the information request, a user specifies the topic.
- *Download.* This action can be performed when an author looks through the list of papers provided by his/her personal agent in response to a query.
- *View.* It indicates that an author clicked on the "view details" link corresponding to one of the papers provided by his/her personal agent.
- *Reject.* It comes with those papers that were recommended by a personal agent, but did not attract any attention (like view or download actions) from the user.

As it is possible to notice, some of the actions are observed directly in the system interface (query, download, view, reject) while others are inferred from the available information (participate, publish, cite). Both lists can be extended.

We would like to stress that we use keywords, or topics to separate papers of authors who have several research interests. This solves the problem mentioned by McNee et al. [11] of the irrelevant recommendations received by users who have only one topic in common with an author that has a wide range of research interests.

There is also a problem of interpretation of reject action. For example, reject from a novice user is more informative, since it can mean that the paper is rejected because it is irrelevant. An expert user can reject the paper just because he/she saw it many times already. We use the past history to analyze these details.

3.2.2 Implicit Culture

The goal of the SICS in general is to infer a cultural theory about users' actions and to encourage a novice to behave according to this theory. The cultural theory captures the users' knowledge about the environment, which is implicitly hidden in the actions. Here we use the actions listed in Table 1 and the formalism presented in [2] to give an example of the theory:

$$\forall a \in Authors, p \in Papers, t \in Topics : \\ \text{query}(a,t) \rightarrow \text{download}(a,p,t). \quad (1)$$

This theory expresses that "all authors who search papers on the topic t download the paper p".

Theory is induced by the inductive module of the SICS. It is extracted from past observations on user behavior using data mining techniques and can be updated after a certain period of time. The composer module produces recommendations taking into account the cultural theory and user preferences, encouraging the user to behave consistently

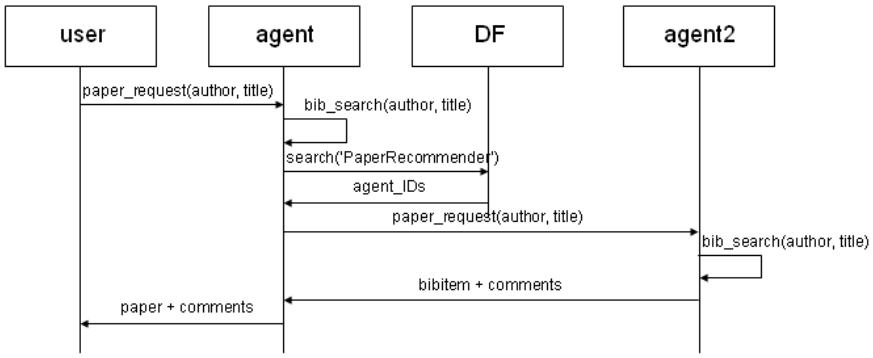


Figure 3: Sequence diagram in response to a paper request.

with the theory. For example, using (1), the composer module can suggest to have a look on the paper p.

We would like to achieve consistence with the theory because it is natural to assume that the way the experienced researchers select relevant papers is much more close to the optimal one than the novice's actions. Therefore, if the novice follows the recommendations, his/her behavior will converge to one nearer to optimal. The relationship characterized by this knowledge transfer is called *"Implicit Culture"*.

3.3 Search in the System

There are two types of user requests: a *paper request* and an *information request*.

3.3.1 Paper Request

When a user submits a paper request, he/she knows the title and the list of the authors. This is reasonable, because he/she could have found this information in another article, or on a conference website. The personal agent of the user checks if user's bibliography contains this item. In case it is present in the bibliography, the system locates the paper and shows it to the user. Otherwise, personal agent uses its previous experience to fill in a list of agents to ask. It is the task of the composer module to propose agents who can answer the query about the target paper. To find suitable agents, the composer module adopts the theory produced by the inductive module and the history of observations. In case the agent cannot fill the list itself, it contacts Directory Facilitator (DF) which returns IDs of agents that offer service "PaperRecommender" on the platform. The agent contacts every agent in the list, propagating user request. The contacted agents respond with bibitem that contains information and comments of their users. Using the bibitem, the agent locates the paper and shows it and the comments to the user. The sequence diagram corresponding to system actions during the search is given in Figure 3.

However, the requested paper can be absent in the bibliographies of other users. In this case, the personal agent contacts different data repositories, like CiteSeer which can be queried from the system. Once the paper is found, the personal agent offers the user either to read the abstract or to download the paper.

3.3.2 Information Request

If a user submits an information request, he/she has to specify the topic he/she is interested in. The personal agent checks if agents of the experts in this topic are present on the platform. If they are on the platform, the agent propagates the user's information request to them. The goal of the virtual community of expert agents is to recommend items that are considered relevant to the topic and to include them in so-called "reading list". To fill it with the items relevant to the topic the agents analyze citations, search and reading behaviors of the community members. Then, the personal agent shows the obtained results to the user. The results of the information request can be supplemented with the results obtained from CiteSeer or from another community.

If there are no topic experts on the platform, the personal agent contacts other agents on the platform, or external platforms in order to find experts in the topic. Note, that here the SICS module of queried agent simulates some kind of reputation scheme by discovering which agent can be an expert. To determine the level of user's expertise it is possible to analyze how often his/her paper is cited. This information can be obtained using CiteSeer. After the list of top experts in the field is filled, the personal agent clones experts' personal agents thus creating an ad hoc community running on the user PC and answering user's queries.

The user browses through a list of results being able to execute a set of basic actions that can be applied to a single item: it is possible to view the details (e.g. abstract) of a paper; a paper can be downloaded or rejected. Information about the actions that user executes (view, download, reject) is recorded by the observer module of the SICS and is also sent to the agents which participated in the search. They use this *feedback* to produce more effective recommendations next time. The personal agent of the user uses this information in further searches too. The sequence diagram of system actions during an information search is depicted in Figure 4.

3.3.3 Feedback Exchange

The goal of feedback exchange is to propagate information about users' actions to the other agents. When an agent contacts other agents in order to receive suggestions, it keeps the list of agents from which it received the results. When the user of the agent-requester accesses one of these results (e.g. view, download, reject) the agent communicates this information to the agent-suggestor. Thus, the agents partially observe actions of other users.

For example, in Figure 4, *agent2* sends results to *agent* in response to the information request. *Agent* shows results

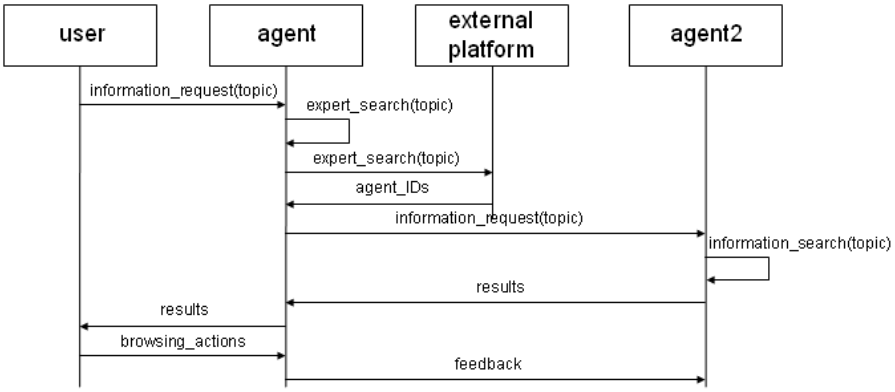


Figure 4: Sequence diagram in response to an information request.

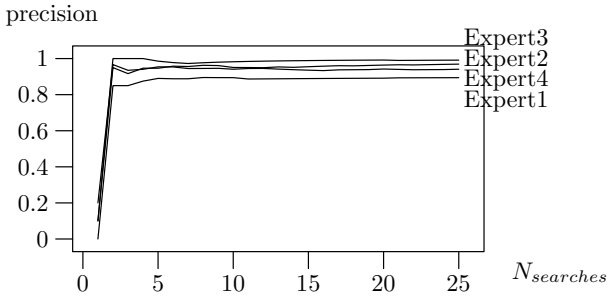


Figure 5: Average precision of 10 simulations with different number of searches.

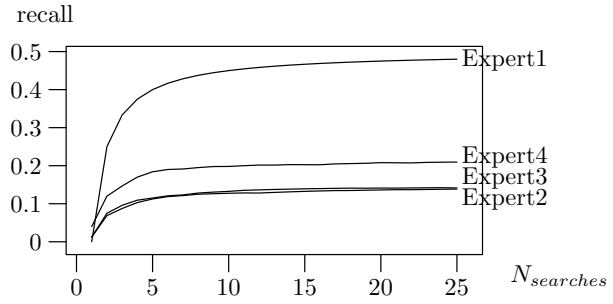


Figure 6: Average recall of 10 simulations with different number of searches.

to its user and records his/her browsing actions. The information about these actions is sent to *agent2* which is able to store them in a database of observations and use in further interactions with *agent* for producing more qualitative suggestions.

4. EVALUATION

In this section we present the experimental results obtained using the platform. We also define the measures we used to estimate the quality of suggestions produced by the SICS.

The aim of the experiment is to check the hypothesis that agents with the SICS are able to produce recommendations in accordance with the "community culture" and adapting

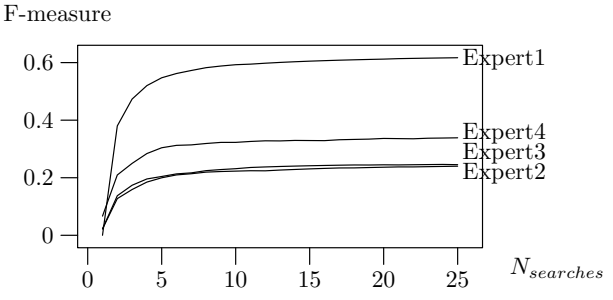


Figure 7: Average F-measure of 10 simulations with different number of searches.

their suggestions to the needs of a single community. We build a community of several experts, and we build their preferences. We expect that personal agent will be able to adapt suggestions to expert preferences.

In our experiment we add in each expert agent a piece of code that imitates behavior of the real user. The main function of this replacement is to generate pseudo-user response to the recommendations. The responses are generated according to a user profile. The user profile determines click-through ratio of the acceptance of the possible results. The dataset contains m results each corresponding to a particular paper. The dataset is used during simulations instead of contacting CiteSeer. The profile determines the probabilities $p(j)$ of choosing the j-th paper, $j \in \{1, \ldots, m\}$ while searching for a specified topic. We assume that the user accepts one and only one paper during search, so $\sum_{j=1}^{m} p(j) = 1$. In our experiment, the number of papers m is equal to 10. However, we would like to stress that the architecture of SICS does not depend on a particular user model, it is rather general being able to incorporate a number of different techniques.

To produce suggestions personal agents in the experiments did not infer any theory, using just comparison of the past actions therefore making algorithm equivalent to collaborative filtering. However, in general, our architecture allows using more complex techniques, like association rules, or clustering.

We use the following measures in order to evaluate the quality of suggestions:

Table 2: Profiles of the experts

	Common rank of the paper									
Expert name	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Expert1	0	0	0	0	0	0	0	0.5	0	0.5
Expert2	0.23	0.18	0.14	0.14	0.14	0.07	0.05	0	0.05	0
Expert3	0.19	0.14	0.14	0.14	0.1	0.14	0.05	0	0.1	0
Expert4	0.29	0.29	0.14	0	0.14	0	0.14	0	0	0

- We call a paper **relevant** to a topic if the probability of its acceptance, as specified in the user profile, is non-zero.
- **Precision** is the ratio of the number of suggested relevant papers to the total number of suggested papers, relevant and irrelevant.
- **Recall** is the ratio of the number of proposed relevant papers to the total number of relevant papers.
- **F-measure** is a trade-off between precision and recall. It is calculated as follows:

$$\text{F-measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}.$$

In order to build a small community of expert agents we generate four different profiles, each representing one expert agent. As experts we select the most cited authors, namely authors of the most cited papers that CiteSeer returns for the keyword "Data Mining". To determine papers the expert consider to be relevant, we take the first five of the most cited papers by each expert and look at the papers they cite. The top ten papers they cite all together are considered to be relevant to the selected group of authors. Then, we create a personal profile for every expert. The profile contains the probability that the expert considers the paper from the top ten selected to be relevant. These probabilities are calculated from the expert's citations, as follows:

$$p(j) = \frac{c_j}{\sum_{i=1}^{m} c_i}, \quad j = 1 \ldots m.$$

Here c_j is the number of times an expert cites the j-th paper in his/her five most cited papers that we select for the experiment. Of course, some of the selected papers are not relevant to a particular expert, being relevant to a group as a whole. The profiles are given in Table 2. The information that we need to determine experts and the information about citations is taken from CiteSeer using a specially developed wrapper.

We repeat searches for the keyword "Data Mining" 25 times, measuring precision, recall and F-measure of the recommendations after every search. We model user acceptance behavior as follows: given a keyword, accepted result is generated randomly according to the distribution that is specified in the profile. Other papers obtained from the SICS and agents are marked as rejected. At the end of 25-searches' session the observation data are deleted. We repeat the search sessions several times in order to control the effect of the order of paper acceptance.

The results contain precision, recall and F-measure of the papers recommended by the SICS module and by the agents queried by expert's personal agent. Lines in Figure 5 represent precision of the recommendations that are produced by the four personal agents for the experts, which names are shown on the right part of the figure. In Figure 6 and Figure 7 we have analogous curves for recall and F-measure, correspondingly.

From these figures we can note that the SICS modules of the personal agents are able to track users' interest, adapting suggestions to the needs of the community. The precision keeps on a very high level (due to the high number of relevant papers in our experiments) while recall and, as a result, F-measure tend to increase over time, reaching adequate values just after five searches.

5. RELATED WORK

In a previous paper [1], we applied Implicit Culture concepts to the development of a multi-agent recommendation system for web search. Although this paper is about web search too, the context of application is completely different. The two architectures are similar, but there are some significant differences. Firstly, here we consider a possibility of ad hoc community creation for information filtering, while in the previous work we were more focused on an organizational community. Secondly, we consider a rather wide set of actions here, while in the previous work it was reduced to *request, accept,* and *reject.* Finally, in the case of web search a lot of repetitions, namely searches for the same keyword several times, occur. In the case of scientific papers, the actions tend to transform from several actions of search type to the actions of citation type, which can be repeated many times for good papers or can be performed only once or even never for bad papers.

A CiteSeer extension aimed at personalization of the search is presented in [4]. This paper describes the mechanism of user profiles creation and maintenance. The information in the profiles indicates user interests and can be used to generate a list of potentially interesting papers. The main difference between our architecture and the one presented in the paper is that we propose collaboration of the users that should lead to the more qualitative search results for all the collaborators. Unlike their approach, we do not require any explicit feedback from the user, collecting information by means of observations and implicit inferences.

There are a number of papers presenting different approaches to facilitating scientific literature search. Bradshaw et al. [5] describe Rosetta, a system that indexes papers based on the way they have been described when cited in other articles. Using this information they find the items that correspond to the different variations of the user query, grouping them by topics. GroupLens group in [11] explores the use of collaborative filtering to recommend research papers. Torres et al. [15] combine content-based and collaborative filtering approaches. They have also shown that different algorithms are more suitable for recommending different kinds of papers. Middleton et al. [12] propose an ontological approach to build a recommendation system for scientific articles. The research interests are organized in an ontology

according to which papers are classified. The papers corresponding to the classes of the ontology that represents user's interests are recommended to the user. The system records relevance feedback to compute users' interests and to recommend papers using collaborative filtering.

The system proposed in this paper has an architecture that permits to deal with several information sources. We have used the notion of Implicit Culture that is more general than collaborative filtering [2]. Moreover, the use of agents in our approach provides several advantages, such as possibility of sharing bibliographies and creation of virtual communities. Finally, with our framework it is possible to recommend not only papers, but also the experts.

6. CONCLUSION AND FUTURE WORK

In this paper, we have presented a multi-agent system that facilitates scientific publication search and helps in finding relevant papers on a specified topic. Agents applying Implicit Culture concepts observe users' behaviors and infer knowledge about users' actions. The system finds relevant papers analyzing citation graph and action history of the user. Personal agents produce via interaction a transfer of knowledge about relevant publications from experts to beginners. The experimental results have proved that the SICS is able to adapt suggestions to the user preferences. However, the results are still preliminary and as part of our future work we plan to complete the evaluation.

As a future work we would like to extend the inductive module. For instance, association rules can be used to find papers usually cited together with other papers. Clustering techniques can be applied to distinguish between several topics of the same author. Clusters should contain the papers that deal with similar topics. We also plan to use CiteSeer API [13] for conducting further experiments. We are also planning to do real-user experiments as future work.

7. ACKNOWLEDGEMENTS

We would like to thank the participants of the poster session at 2K* 2005 symposium for their valuable comments. This work is funded by research projects EU SERENITY "System Engineering for Security and Dependability", COFIN "Artificial Intelligence Techniques for the Retrieval of High Quality Information on the Web" and by Fondo Progetti PAT, MOSTRO "Modeling Security and Trust Relationships within Organizations" and QUIEW (Quality-based indexing of the Web), art. 9, Legge Provinciale 3/2000, DGP n. 1587 dd. 09/07/04.

The authors would like to thank the anonymous reviewers for their helpful comments.

8. REFERENCES

[1] A. Birukov, E. Blanzieri, and P. Giorgini. Implicit: An agent-based recommendation system for web search. In *Proceedings of the 4th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 618–624. ACM Press, 2005.

[2] E. Blanzieri and P. Giorgini. From collaborative filtering to implicit culture: a general agent-based framework. In *Proceedings of the Workshop on Agents and Recommender Systems*, Barcellona, 2000.

[3] E. Blanzieri, P. Giorgini, P. Massa, and S. Recla. Implicit culture for multi-agent interaction support. In *CooplS '01: Proceedings of the 9th International Conference on Cooperative Information Systems*, pages 27–39, London, UK, 2001. Springer-Verlag.

[4] K. D. Bollacker, S. Lawrence, and C. L. Giles. Discovering relevant scientific literature on the web. *IEEE Intelligent Systems*, 15(2):42–47, 2000.

[5] S. Bradshaw, A. Scheinkman, and K. Hammond. Guiding people to information: providing an interface to a digital library using reference as a basis for indexing. In *IUI '00: Proceedings of the 5th international conference on Intelligent user interfaces*, pages 37–43, New York, NY, USA, 2000. ACM Press.

[6] L. Chen and K. Sycara. Webmate: a personal agent for browsing and searching. In *AGENTS '98: Proceedings of the second international conference on Autonomous agents*, pages 132–139, New York, NY, USA, 1998. ACM Press.

[7] Google scholar: http://scholar.google.com/.

[8] W. C. Janssen and K. Popat. Uplib: a universal personal digital library system. In *DocEng '03: Proceedings of the 2003 ACM symposium on Document engineering*, pages 234–242, New York, NY, USA, 2003. ACM Press.

[9] S. Lawrence, C. L. Giles, and K. Bollacker. Digital libraries and autonomous citation indexing. *Computer*, 32(6):67–71, 1999.

[10] H. Lieberman. Letizia: An agent that assists web browsing. In C. S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 924–929, Montreal, Quebec, Canada, 1995. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[11] S. M. McNee, I. Albert, D. Cosley, P. Gopalkrishnan, S. K. Lam, A. M. Rashid, J. A. Konstan, and J. Riedl. On the recommending of citations for research papers. In *CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 116–125, New York, NY, USA, 2002. ACM Press.

[12] S. E. Middleton, N. R. Shadbolt, and D. C. D. Roure. Ontological user profiling in recommender systems. *ACM Trans. Inf. Syst.*, 22(1):54–88, 2004.

[13] Y. Petinot, C. L. Giles, V. Bhatnagar, P. B. Teregowda, H. Han, and I. Councill. Citeseer-api: towards seamless resource location and interlinking for digital libraries. In *CIKM '04: Proceedings of the thirteenth ACM conference on Information and knowledge management*, pages 553–561, New York, NY, USA, 2004. ACM Press.

[14] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, Chapel Hill, North Carolina, United States, 1994. ACM Press.

[15] R. Torres, S. M. McNee, M. Abel, J. A. Konstan, and J. Riedl. Enhancing digital libraries with techlens+. In *JCDL '04: Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 228–236, New York, NY, USA, 2004. ACM Press.

[16] J. White. Acm opens portal to computing literature. *Communications of the ACM*, 44(7):14–16, 2001.

Security in Multiagent Systems by Policy Randomization

Praveen Paruchuri, Milind Tambe, Fernando Ordóñez, Sarit Kraus*
University of Southern California, Los Angeles, CA 90089, {paruchur, tambe, fordon}@usc.edu
* Bar-Ilan University, Ramat-Gan 52900, Israel, sarit@cs.biu.ac.il

ABSTRACT

Security in multiagent systems is commonly defined as the ability of the system to deal with intentional threats from other agents. This paper focuses on domains where such intentional threats are caused by unseen adversaries whose actions or payoffs are unknown. In such domains, action randomization can effectively deteriorate an adversary's capability to predict and exploit an agent/agent team's actions. Unfortunately, little attention has been paid to intentional randomization of agents' policies in single-agent or decentralized (PO)MDPs without significantly sacrificing rewards or breaking down coordination. This paper provides two key contributions to remedy this situation. First, it provides three novel algorithms, one based on a non-linear program and two based on linear programs (LP), to randomize single-agent policies, while attaining a certain level of expected reward. Second, it provides Rolling Down Randomization (*RDR*), a new algorithm that efficiently generates randomized policies for decentralized POMDPs via the single-agent LP method.

Categories and Subject Descriptors

I.2.8 [Artificial Intelligence]: Distributed Artificial Intelligence - Multi Agent Systems

General Terms

Security, Design, Theory

Keywords

Safety and Security in Agent Systems, Teamwork, MDP, POMDP, Decentralized POMDP

1. INTRODUCTION

In many adversarial domains, it is crucial for agent and agent teams based on single-agent or decentralized (PO)MDPs to randomize policies in order to avoid action predictability. Such policy randomization is crucial for security in domains where we cannot explicitly model our adversary's actions and capabilities or its payoffs, but the adversary observes our agents' actions and exploits any action predictability in some unknown fashion. Consider agents that schedule security inspections, maintenance or refueling at seaports or airports. Adversaries may be unobserved terrorists with unknown capabilities and actions, who can learn the schedule from observations. If the schedule is deterministic, then these adversaries may exploit schedule predictability to intrude or attack and cause tremendous unanticipated sabotage. Alternatively, consider a team of UAVs (Unmanned Air Vehicles)[2] monitoring a region undergoing a humanitarian crisis. Adversaries may be humans intent on causing some significant unanticipated harm — e.g. disrupting food convoys, harming refugees or shooting down the UAVs — the adversary's capabilities, actions or payoffs are unknown and difficult to model explicitly. However, the adversaries can observe the UAVs and exploit any predictability in UAV surveillance, e.g. engage in unknown harmful actions by avoiding the UAVs' route.

While we cannot explicitly model the adversary's actions, capabilities or payoffs, in order to ensure security of the agent/agent-team we make two worst case assumptions about the adversary. (We show later that a weaker adversary, i.e. one who fails to satisfy these assumptions, will in general only lead to enhanced security.) The first assumption is that the adversary can estimate the agent's state or belief state. In fully observable domains, the adversary estimates the agent's state to be the current world state which both can observe fully. If the domain is partially observable, we assume that the adversary estimates the agent's belief states, because: (i) the adversary eavesdrops or spies on the agent's sensors such as sonar or radar (e.g., UAV/robot domains); or (ii) the adversary estimates the most likely observations based on its model of the agent's sensors; or (iii) the adversary is co-located and equipped with similar sensors. The second assumption is that the adversary knows the agents' policy, which it may do by learning over repeated observations or obtaining this policy via espionage or other exploitation.

Thus, we assume that the adversary may have enough information to predict the agents' actions with certainty if the agents followed a deterministic policy. Hence, our work maximizes policy randomization to thwart the adversary's prediction of the agent's actions based on the agent's state and minimize adversary's ability to cause harm . Unfortunately, while randomized policies are created as a side effect [1] and turn out to be optimal in some stochastic games [10], little attention has been paid to intentionally maximizing randomization of agents' policies even for single agents. Obviously, simply randomizing an MDP/POMDP policy can degrade an agent's expected rewards, and thus we face a randomization-reward tradeoff problem: how to randomize policies with only a limited loss in expected rewards. Indeed, gaining an explicit understanding of the randomization-reward tradeoff requires new techniques for policy generation rather than the traditional single-objective maximization techniques. However, generating policies that provide appropriate randomization-reward tradeoffs is difficult, a difficulty that is exacerbated in agent teams based on decentralized POMDPs,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

as randomization may create miscoordination.

This paper provides two key contributions to remedy this situation. First, we provide novel techniques that enable policy randomization in single agents, while attaining a certain expected reward threshold. We measure randomization via an entropy-based metric (although our techniques are not dependent on that metric). In particular, we illustrate that simply maximizing entropy-based metrics introduces a non-linear program that does not guarantee polynomial run-time. Hence, we introduce our CRLP (Convex combination for Randomization) and BRLP (Binary search for Randomization) linear programming (LP) techniques that randomize policies in polynomial time with different tradeoffs as explained later. The second part of the paper provides a new algorithm, RDR (Rolling Down Randomization), for generating randomized policies for decentralized POMDPs, given a threshold on the expected team reward loss. RDR starts with a joint deterministic policy for decentralized POMDPs, then iterates, randomizing policies for agents turn-by-turn, keeping policies of all other agents fixed. A key insight in RDR is that given fixed randomized policies for other agents, we can generate a randomized policy via CRLP or BRLP, i.e., our efficient single-agent methods.

The motivation for use of entropy-based metrics to randomize our agents' policies stems from information theory. It is well known that the expected number of probes (e.g., observations) needed to identify the outcome of a distribution is bounded below by the entropy of that distribution [15]. Thus, by increasing policy entropy, we force the opponent to execute more probes to identify the outcome of our known policy, making it more difficult for the opponent to anticipate our agent's actions and cause harm. In particular, in our (PO)MDP setting, the conflict between the agents and the adversary can be interpreted as a game, in which the agents generate a randomized policy above a given expected reward threshold; the adversary knows the agent's policy and the adversary's action is to guess the exact action of the agent/agent-team by probing. For example, in the UAV setting, given our agent's randomized policy, the adversary generates probes to determine the direction our UAV is headed from a given state. Thus, in the absence of specific knowledge of the adversary, we can be sure to increase the average number of probes the adversary uses by increasing the lower bound given by the entropy of the policy distribution at every state.

2. RANDOMIZATION: SINGLE AGENTS

Before considering agent teams, we focus on randomizing single agent MDP policies, e.g. a single MDP-based UAV agent is monitoring a troubled region, where the UAV gets rewards for surveying various areas of the region, but as discussed above, security requires it to randomize its monitoring strategies to avoid predictability.

An MDP is denoted as a tuple: $\langle S, A, P, R\rangle$. S is a set of world states $\{s_1, \ldots, s_m\}$, A the set of actions $\{a_1, \ldots, a_k\}$, P the set of tuples $p(s, a, j)$ representing the transition function and R the set of tuples $r(s, a)$ denoting the immediate reward. If $x(s, a)$ represents the number of times the MDP visits state s and takes action a and α_j represents the number of times that the MDP starts in each state $j \in S$, then the optimal policy, maximizing expected reward, is derived via the following linear program [4]:

$$\begin{aligned} \max \quad & \sum_{s\in S}\sum_{a\in A} r(s,a)x(s,a) \\ \text{s.t.} \quad & \sum_{a\in A} x(j,a) - \sum_{s\in S}\sum_{a\in A} p(s,a,j)x(s,a) = \alpha_j \qquad \forall j \in S \\ & x(s,a) \geq 0 \qquad \forall s \in S, a \in A \end{aligned} \tag{1}$$

If x^* is the optimal solution to (1), optimal policy π^* is given by (2) below, where $\pi^*(s, a)$ is the probability of taking action a in state s. It turns out that π^* is deterministic and uniformly optimal regardless of the initial distribution $\{\alpha_j\}_{j\in S}$ [4] i.e., $\pi^*(s, a)$ has a value of either 0 or 1. However, such deterministic policies are undesirable in domains like our UAV example.

$$\pi^*(s,a) = \frac{x^*(s,a)}{\sum_{\hat{a}\in A} x^*(s,\hat{a})} \; . \tag{2}$$

2.1 Randomness of a policy

We borrow from information theory the concept of entropy of a set of probability distributions to quantify the randomness, or information content, in a policy of the MDP. For a discrete probability distribution $p_1, \ldots, p_n$ the only function, up to a multiplicative constant, that captures the randomness is the entropy, given by the formula $H = -\sum_{i=1}^{n} p_i \log p_i$ [15]. We introduce a *weighted entropy* function to quantify the randomness in a policy π of an MDP and express it in terms of the underlying frequency x. Note from the definition of a policy π in (2) that for each state s the policy defines a probability distribution over actions. The weighted entropy is defined by adding the entropy for the distributions at every state weighted by the likelihood the MDP visits that state, namely

$$\begin{aligned} H_W(x) &= -\sum_{s\in S} \frac{\sum_{\hat{a}\in A} x(s,\hat{a})}{\sum_{j\in S}\alpha_j} \sum_{a\in A} \pi(s,a)\log\pi(s,a) \\ &= -\frac{1}{\sum_{j\in S}\alpha_j} \sum_{s\in S}\sum_{a\in A} x(s,a)\log\left(\frac{x(s,a)}{\sum_{\hat{a}\in A} x(s,\hat{a})}\right) \; . \end{aligned}$$

We note that the randomization approach we propose works for alternative functions of the randomness yielding similar results. For example we can define an *additive entropy* taking a simple sum of the individual state entropies as follows:

$$\begin{aligned} H_A(x) &= -\sum_{s\in S}\sum_{a\in A} \pi(s,a)\log\pi(s,a) \\ &= -\sum_{s\in S}\sum_{a\in A} \frac{x(s,a)}{\sum_{\hat{a}\in A} x(s,\hat{a})} \log\left(\frac{x(s,a)}{\sum_{\hat{a}\in A} x(s,\hat{a})}\right) \; , \end{aligned}$$

We now present three algorithms to obtain random solutions that maintain an expected reward of at least $E_{\min}$ (a certain fraction of the maximal expected reward E^* obtained solving (1)). These algorithms result in policies that, in our UAV-type domains, enable an agent to get a sufficiently high expected reward, e.g. surveying enough area, using randomized flying routes to avoid predictability.

2.2 Maximal entropy solution

We can obtain policies with maximal entropy but a threshold expected reward by replacing the objective of Problem (1) with the definition of the weighted entropy $H_W(x)$. The reduction in expected reward can be controlled by enforcing that feasible solutions achieve at least a certain expected reward $E_{\min}$. The following problem maximizes the weighted entropy while maintaining the expected reward above $E_{\min}$:

$$\begin{aligned} \max \quad & -\frac{1}{\sum_{j\in S}\alpha_j} \sum_{s\in S}\sum_{a\in A} x(s,a)\log\left(\frac{x(s,a)}{\sum_{\hat{a}\in A} x(s,\hat{a})}\right) \\ \text{s.t.} \quad & \sum_{a\in A} x(j,a) - \sum_{s\in S}\sum_{a\in A} p(s,a,j)x(s,a) = \alpha_j \qquad \forall j \in S \\ & \sum_{s\in S}\sum_{a\in A} r(s,a)x(s,a) \geq E_{\min} \\ & x(s,a) \geq 0 \qquad \forall s \in S, a \in A \end{aligned} \tag{3}$$

$E_{\min}$ is an input domain parameter (e.g. UAV mission specification). Alternatively, if E^* denotes the maximum expected reward from (1), then by varying the expected reward threshold $E_{\min} \in [0,E^*]$ we can explore the tradeoff between the achievable expected reward and entropy, and then select the appropriate $E_{\min}$. Note that for $E_{\min} = 0$ the above problem finds the maximum weighted entropy policy, and for $E_{\min} = E^*$, Problem (3) returns the maximum expected reward policy with largest entropy. Solving Problem (3) is our first algorithm to obtain a randomized policy that achieves at least $E_{\min}$ expected reward (Algorithm 1).

Algorithm 1 MAX-ENTROPY($E_{\min}$)

1: Solve Problem (3) with $E_{\min}$, let $x_{E_{\min}}$ be optimal solution
2: **return** $x_{E_{\min}}$ (maximal entropy, expected reward $\geq E_{\min}$)

Unfortunately entropy-based functions like $H_W(x)$ are neither convex nor concave in x, hence there are no complexity guarantees in solving Problem (3), even for local optima [16]. This negative complexity motivates the polynomial methods presented next.

2.3 Efficient single agent randomization

The idea behind these polynomial algorithms is to efficiently solve problems that obtain policies with a high expected reward while maintaining some level of randomness. (A very high level of randomness implies a uniform probability distribution over the set of actions out of a state, whereas a low level would mean deterministic action being taken from a state). We then obtain a solution that meets a given minimal expected reward value by adjusting the level of randomness in the policy. The algorithms that we introduce in this section consider two inputs: a minimal expected reward value $E_{\min}$ and a randomized solution $\bar{x}$ (or policy $\bar{\pi}$). The input $\bar{x}$ can be any solution with high entropy and is used to enforce some level of randomness on the high expected reward output, through linear constraints. For example, one such high entropy input for MDP-based problems is the uniform policy, where $\bar{\pi}(s,a) = 1/|A|$. We enforce the amount of randomness in the high expected reward solution that is output through a parameter $\beta \in [0,1]$. For a given β and a high entropy solution $\bar{x}$, we output a maximum expected reward solution with a certain level of randomness by solving (4).

$$\begin{aligned} \max \quad & \sum_{s\in S}\sum_{a\in A} r(s,a)x(s,a) \\ \text{s.t.} \quad & \sum_{a\in A} x(j,a) - \sum_{s\in S}\sum_{a\in A} p(s,a,j)x(s,a) = \alpha_j \qquad \forall j \in S \\ & x(s,a) \geq \beta\bar{x}(s,a) \qquad \forall s \in S, a \in A\ . \end{aligned} \tag{4}$$

which can be referred to in matrix shorthand as

$$\begin{aligned} \max \quad & r^T x \\ \text{s.t.} \quad & Ax = \alpha \\ & x \geq \beta\bar{x}\ . \end{aligned}$$

As the parameter β is increased, the randomness requirements of the solution become stricter and hence the solution to (4) would have smaller expected reward and higher entropy. For $\beta = 0$ the above problem reduces to (1) returning the maximum expected reward solution E^*; and for $\beta = 1$ the problem obtains the maximal expected reward (denoted $\overline{E}$) out of all solutions with as much randomness as $\bar{x}$. If E^* is finite, then Problem (4) returns $\bar{x}$ for $\beta = 1$ and $\overline{E} = \sum_{s\in S}\sum_{a\in A} r(s,a)\bar{x}(s,a)$.

Our second algorithm to obtain an efficient solution with a expected reward requirement of $E_{\min}$ is based on the following result which shows that the solution to (4) is a convex combination of the deterministic and highly random input solutions.

THEOREM 1. *Consider a solution $\bar{x}$, which satisfies $A\bar{x} = \alpha$ and $\bar{x} \geq 0$. Let x^* be the solution to (1) and $\beta \in [0,1]$. If x_β is the solution to (4) then $x_\beta = (1-\beta)x^* + \beta\bar{x}$.*

proof: We reformulate problem (4) in terms of the slack $z = x - \beta\bar{x}$ of the solution x over $\beta\bar{x}$ leading to the following problem :

$$\begin{aligned} \beta r^T\bar{x} + \quad \max \quad & r^T z \\ \text{s.t.} \quad & Az = (1-\beta)\alpha \\ & z \geq 0\ , \end{aligned}$$

The above problem is equivalent to (4), where we used the fact that $A\bar{x} = \alpha$. Let z^* be the solution to this problem, which shows that $x_\beta = z^* + \beta\bar{x}$. Dividing the linear equation $Az = (1-\beta)\alpha$, by $(1-\beta)$ and substituting $u = z/(1-\beta)$ we recover the deterministic problem (1) in terms of u, with u^* as the optimal deterministic solution. Renaming variable u to x, we obtain $\frac{1}{1-\beta}z^* = x^*$, which concludes the proof. □

Since $x_\beta = (1-\beta)x^* + \beta\bar{x}$, we can directly find a randomized solution which obtains a target expected reward of $E_{\min}$. Due to the linearity in relationship between x_β and β, a linear relationship exists between the expected reward obtained by x_β (i.e $r^T x_\beta$) and β. In fact setting $\beta = \frac{r^T x^* - E_{\min}}{r^T x^* - r^T \bar{x}}$ makes $r^T x_\beta = E_{\min}$. We now present below algorithm CRLP based on the observations made about β and x_β.

Algorithm 2 CRLP($E_{\min}, \bar{x}$)

1: Solve Problem (1), let x^* be the optimal solution
2: Set $\beta = \frac{r^T x^* - E_{\min}}{r^T x^* - r^T \bar{x}}$
3: Set $x_\beta = (1-\beta)x^* + \beta\bar{x}$
4: **return** x_β (expected reward $= E_{\min}$, entropy based on $\beta\bar{x}$)

Algorithm CRLP is based on a linear program and thus obtains, in polynomial time, solutions to problem(4) with expected reward values $E_{\min} \in [\overline{E}, E^*]$. Note that Algorithm CRLP might unnecessarily constrain the solution set as Problem(4) implies that at least $\beta\sum_{a\in A}\bar{x}(s,a)$ flow has to reach each state s. This restriction may negatively impact the entropy it attains, as experimentally verified in Section 5. This concern is addressed by a reformulation of Problem (4) replacing the flow constraints by policy constraints at each stage. For a given $\beta \in [0,1]$ and a solution $\bar{\pi}$ (policy calculated from $\bar{x}$), this replacement leads to the following linear program

$$\begin{aligned} \max \quad & \sum_{s\in S}\sum_{a\in A} r(s,a)x(s,a) \\ \text{s.t.} \quad & \sum_{a\in A} x(j,a) - \sum_{s\in S}\sum_{a\in A} p(s,a,j)x(s,a) = \alpha_j, \quad \forall j \in S \\ & x(s,a) \geq \beta\bar{\pi}(s,a)\sum_{b\in A} x(s,b), \qquad \forall s \in S, a \in A\ . \end{aligned} \tag{5}$$

For $\beta = 0$ this problem reduces to (1) returning E^*, for $\beta = 1$ it returns a maximal expected reward solution with the same policy as $\bar{\pi}$. This means that for β at values 0 and 1, problems (4) and (5) obtain the same solution if policy $\bar{\pi}$ is the policy obtained from the flow function $\bar{x}$. However, in the intermediate range of 0 to 1 for β, the policy obtained by problems (4) and (5) are different even if $\bar{\pi}$ is obtained from $\bar{x}$. Thus, theorem 1 holds for problem (4) but not for (5). Table 1, obtained experimentally, validates our claim by showing the maximum expected rewards and entropies obtained (entropies in parentheses) from problems (4) and (5) for various settings of β, e.g. for β = 0.4, problem (4) provides a maximum expected reward of 26.29 and entropy of 5.44, while problem (5) provides a maximum expected reward of 25.57 and entropy of 6.82.

Table 1 shows that for the same value of β in Problems (4) and

(5) we get different maximum expected rewards and entropies implying that the optimal policies for both problems are different, hence Theorem 1 does not hold for (5). Indeed, while the expected reward of Problem (4) is higher for this example, its entropy is lower than Problem (5). Hence to investigate another randomization-reward tradeoff point, we introduce our third algorithm BRLP, which uses problem (5) to perform a binary search to attain a policy with expected reward $E_{\min} \in [\overline{E}, E^*]$, adjusting the parameter β.

Beta	.2	.4	.6	.8
$Problem(4)$	29.14 (3.10)	26.29 (5.44)	23.43 (7.48)	20.25 (9.87)
$Problem(5)$	28.57 (4.24)	25.57 (6.82)	22.84 (8.69)	20.57 (9.88)

Table 1: Maximum expected rewards(entropies) for various β

Algorithm 3 $\mathrm{BRLP}(E_{\min}, \bar{x})$

1: Set $\beta_l = 0$, $\beta_u = 1$, and $\beta = 1/2$.
2: Obtain $\bar{\pi}$ from $\bar{x}$
3: Solve Problem (5), let x_β and $E(\beta)$ be the optimal solution and expected reward value returned
4: **while** $|E(\beta) - E_{\min}| > \epsilon$ **do**
5: **if** $E(\beta) > E_{\min}$ **then**
6: Set $\beta_l = \beta$
7: **else**
8: Set $\beta_u = \beta$
9: $\beta = \frac{\beta_u + \beta_l}{2}$
10: Solve Problem (5), let x_β and $E(\beta)$ be the optimal solution and expected reward value returned
11: **return** x_β (expected reward $= E_{\min} \pm \epsilon$, entropy related to $\beta\bar{x}$)

Given input $\bar{x}$, algorithm BRLP runs in polynomial time, since at each iteration it solves an LP and for tolerance of ϵ, it takes at most $O\left(\frac{E(0)-E(1)}{\epsilon}\right)$ iterations to converge (E(0) and E(1) expected rewards correspond to 0 and 1 values of β).

2.4 Incorporating models of the adversary

Throughout this paper, we set $\bar{x}$ based on uniform randomization $\bar{\pi} = 1/|A|$. By manipulating $\bar{x}$, we can accommodate the knowledge of the behavior of the adversary. For instance, if the agent knows that a specific state s cannot be targeted by the adversary, then $\bar{x}$ for that state can have all values 0, implying that no entropy constraint is necessary. In such cases, $\bar{x}$ will not be a complete solution for the MDP but rather concentrate on the sets of states and actions that are under risk of attack. For $\bar{x}$ that do not solve the MDP Theorem 1 does not hold and therefore Algorithm CRLP is not valid. In this case, a high-entropy solution that meets a target expected reward can still be obtained via Algorithm BRLP.

Before turning to agent teams next, we quickly discuss applying these algorithms in single agent POMDPs. For single-agent finite-horizon POMDPs with known starting belief states[12], we convert the POMDP to (finite horizon) *belief MDP*, allowing BRLP/CRLP to be applied; returning a randomized policy. However, addressing unknown starting belief states is an issue for future work.

3. FROM SINGLE AGENT TO AGENT TEAMS

The logical first step in moving to agent teams would be to start with multiagent MDPs, where a centralized planner provides policies to multiple agents in a fully observable setting. However, such multiagent MDPs are equivalent to single agent MDPs [14]. Indeed, assuming a shared, observable random device — justified, given a fully observable setting — our BRLP/CRLP algorithms can already generate randomized policies for such multiagent MDPs. Hence, this section focuses on the more general decentralized POMDPs, that are not equivalent to single agent POMDPs [3].

3.1 MTDP: A Decentralized POMDP model

We use notation from MTDP (Multiagent Team Decision Problem) [14] for our decentralized POMDP model; other models are equivalent [3]. Given a team of n agents, an MTDP is defined as a tuple: $\langle S, A, P, \Omega, O, R\rangle$. S is a finite set of world states $\{s_1, \ldots, s_m\}$. $A = \times_{1\le i\le n} A_i$, where $A_1, \ldots, A_n$, are the sets of action for agents 1 to n. A joint action is represented as $\langle a_1, \ldots, a_n\rangle$. $P(s_i, \langle a_1, \ldots, a_n\rangle, s_f)$, the transition function, represents the probability that the current state is s_f, if the previous state is s_i and the previous joint action is $\langle a_1, \ldots, a_n\rangle$. $\Omega = \times_{1\le i\le n}\Omega_i$ is the set of joint observations where Ω_i is the set of observations for agents i. $O(s, \langle a_1, \ldots, a_n\rangle, \omega)$, the observation function, represents the probability of joint observation $\omega \in \Omega$, if the current state is s and the previous joint action is $\langle a_1, \ldots, a_n\rangle$. We assume that observations of each agent are independent of each other's observations, i.e. $O(s, \langle a_1, \ldots, a_n\rangle, \omega) = O_1(s, \langle a_1, \ldots, a_n\rangle, \omega_1) \cdot \ldots \cdot O_n(s, \langle a_1, \ldots, a_n\rangle, \omega_n)$. The agents receive a single, immediate joint reward $R(s, \langle a_1, \ldots, a_n\rangle)$. For deterministic policies, each agent i chooses its actions based on its policy, Π_i, which maps its observation history to actions. Thus, at time t, agent i will perform action $\Pi_i(\vec{\omega}_i^t)$ where $\vec{\omega}_i^t = \omega_i^1, ..., \omega_i^t$. $\Pi = \langle \Pi_1, ..., \Pi_n\rangle$ refers to the joint policy of the team of agents. In this model, execution is distributed but planning is centralized; and agents don't know each other's observations and actions at run time.

Unlike previous work, in our work, policies are randomized and hence agents obtain a probability distribution over a set of actions rather than a single action. Furthermore, this probability distribution is indexed by a sequence of action-observation tuples rather than just observations, since observations do not map to unique actions. Thus in MTDP, a randomized policy maps $\mathbf{\Psi_i^t}$ to a probability distribution over actions, where $\mathbf{\Psi_i^t} = \langle \psi_i^1, \ldots, \psi_i^t\rangle$ and $\psi_i^t = \langle a_i^{t-1}, \omega_i^t\rangle$. Thus, at time t, agent i will perform an action selected randomly based on the probability distribution returned by $\Pi_i(\mathbf{\Psi_i^t})$. Furthermore we denote the probability of an individual action under policy Π_i given $\mathbf{\Psi_i^t}$ as $P^{\Pi_i}(a_i^t|\mathbf{\Psi_i^t})$.

3.2 Illustrative UAV team Domain

To demonstrate key concepts in our algorithms, we introduce a simple UAV team domain that is analogous to the illustrative multi-agent tiger domain [11] except for an adversary – indeed, to enable replicable experiments, rewards, transition and observation probabilities from [11] are used. Consider a region in a humanitarian crisis, where two UAVs execute daily patrols to monitor safe food convoys. However, these food convoys may be disrupted by landmines placed in their route. The convoys pass over two regions: Left and Right. For simplicity, we assume that only one such landmine may be placed at any point in time, and it may be placed in any of the two regions with equal probability. The UAVs must destroy the landmine to get a high positive reward whereas trying to destroy a region without a landmine disrupts transportation and creates a high negative reward; but the UAV team is unaware of which region has the landmine. The UAVs can perform three actions *Shoot-left, Sense* and *Shoot-right* but they cannot communicate with each other. We assume that both UAVs are observed with equal probability by some unknown adversary with unknown capabilities, who wishes to cause sabotage. Our worst case assumption about the adversary is as stated in Section 1: (a) the adversary has access to UAV policies due to learning or espionage (b) the adversary eavesdrop or estimates the UAV observations. Thus, if the policy is not randomized, the adversary may exploit UAV action predictability in some unknown way such as jamming UAV sensors, shooting down UAVs or attacking the food convoys, etc. Since little is known about the adversary's ability to cause sabo-

tage, the UAV team must maximize the adversary's uncertainty via policy randomization, while ensuring an above-threshold reward.

When an individual UAV takes action *Sense*, it leaves the state unchanged, but provides a noisy observation OR or OL, to indicate whether the landmine is to the left or right. The *Shoot-left* and *Shoot-right* actions are used to destroy the landmine, but the landmine is destroyed only if both UAVs simultaneously take either *Shoot-left* or *Shoot-right* actions. Unfortunately, if agents miscoordinate and one takes *Shoot-left* and the other *Shoot-right* they incur high negative reward as the landmine is not destroyed but the food-convoy route is damaged. Once the shoot action occurs, the problem is restarted (the UAVs face a landmine the next day).

4. RANDOMIZATION: AGENT TEAMS

Let p_i be the probability of adversary targeting agent i, and $H_W(i)$ be the weighted entropy for agent i's policy. We design an algorithm that maximizes the *multiagent weighted entropy*, given by $\sum_{i=1}^{n} p_i * H_W(i)$, in MTDPs while maintaining the team's expected joint reward above a threshold. Unfortunately, generating optimal policies for decentralized POMDPs is of higher complexity (NEXP-complete) than single agent MDPs and POMDPs [3], i.e., MTDP presents a fundamentally different class where we cannot directly use the single agent randomization techniques.

Hence, to exploit efficiency of algorithms like BRLP or CRLP, we convert the MTDP into a single agent POMDP, but with a method that changes the state space considered. To this end, our new iterative algorithm called RDR (Rolling Down Randomization) iterates through finding the best randomized policy for one agent while fixing the policies for all other agents — we show that such iteration of fixing the randomized policies of all but one agent in the MTDP leads to a single agent problem being solved at each step. Thus, each iteration can be solved via BRLP or CRLP. For a two agent case, we fix the policy of agent i and generate best randomized policy for agent j and then iterate with agent j's policy fixed.

Overall RDR starts with an initial joint deterministic policy calculated in the algorithm as a starting point. Assuming this fixed initial policy as providing peak expected reward, the algorithm then rolls down the reward, randomizing policies turn-by-turn for each agent. Rolling down from such an initial policy allows control of the amount of expected reward loss from the given peak, in service of gaining entropy. *The key contribution of the algorithm is in the rolling down procedure that gains entropy (randomization)*, and this procedure is independent of how the initial policy for peak reward is determined. The initial policy may be computed via algorithms such as [6] that determine a global optimal joint policy (but at a high cost) or from random restarts of algorithms that compute a locally optimal policy [11, 5], that may provide high quality policies at lower cost. The amount of expected reward to be rolled down is input to RDR. RDR then achieves the rolldown in $1/d$ steps where d is an input parameter.

The turn-by-turn nature of RDR suggests some similarities to JESP [11], which also works by fixing the policy of one agent and computing the best-response policy of the second and iterating. However, there are significant differences between RDR and JESP, as outlined below: (i) JESP uses conventional value iteration based techniques whereas RDR creates randomized policies via LP formulations. (ii) RDR defines a new extended state and hence the belief-update, transition and reward functions undergo a major transformation. (iii) The d parameter is newly introduced in RDR to control number of rolldown steps. (iv) RDR climbs down from a given optimal solution rather than JESP's hill-climbing up solution.

RDR Details: For expository purposes, we use a two agent domain, but we can easily generalize to n agents. We fix the policy of one agent (say agent 2), which enables us to create a single agent POMDP if agent 1 uses an extended state, i.e. at each time t, agent 1 uses an extended state $e_1^t = \langle s^t, \mathbf{\Psi_2^t} \rangle$. Here, $\mathbf{\Psi_2^t}$ is as introduced in the previous section. By using e_1^t as agent 1's state at time t, given the fixed policy of agent 2, we can define a single-agent POMDP for agent 1 with transition and observation function as follows.

$$\begin{aligned} P'(e_1^t, a_1^t, e_1^{t+1}) &= P(\langle s^{t+1}, \mathbf{\Psi_2^{t+1}} \rangle | \langle s^t, \mathbf{\Psi_2^t} \rangle, a_1^t) \\ &= P(\omega_2^{t+1} | s^{t+1}, a_2^t, \mathbf{\Psi_2^t}, a_1^t) \\ &\quad \cdot P(s^{t+1} | s^t, a_2^t, \mathbf{\Psi_2^t}, a_1^t) \cdot P(a_2^t | s^t, \mathbf{\Psi_2^t}, a_1^t) \\ &= P(s^t, \langle a_2^t, a_1^t \rangle, s^{t+1}) \cdot O_2(s^{t+1}, \langle a_2^t, a_1^t \rangle, \omega_2^{t+1}) \\ &\quad \cdot P^{\Pi_2}(a_2^t | \mathbf{\Psi_2^t}) \end{aligned} \tag{6}$$

$$\begin{aligned} O'(e_1^{t+1}, a_1^t, \omega_1^{t+1}) &= \Pr(\omega_1^{t+1} | e_1^{t+1}, a_1^t) \\ &= O_1(s^{t+1}, \langle a_2^t, a_1^t \rangle, \omega_1^{t+1}) \end{aligned} \tag{7}$$

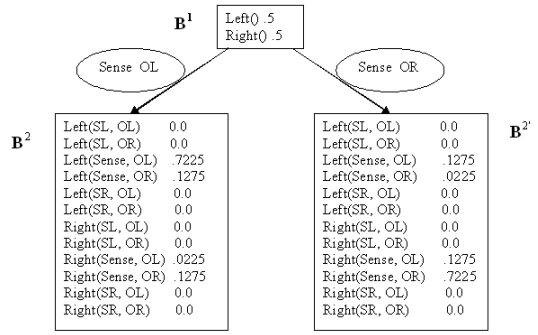


Figure 1: RDR applied to UAV team domain

Thus, we can create a belief state for agent i in the context of j's fixed policy by maintaining a distribution over $e_i^t = \langle s^t, \mathbf{\Psi_j^t} \rangle$. Figure 1 shows three belief states for agent 1 in the UAV domain. For instance B^2 shows probability distributions over e_1^2. In $e_1^2 = (\text{Left}\langle Sense, OL \rangle)$, Left implies landmine to the left is the current state, Sense is the agent 2's action at time 1, OL (Observe Left) is agent 2's observation at time 2. The belief update rule derived from the transition and observation functions is given in (8), where denominator is the transition probability when action a_1 from belief state B_1^t results in ω_1^{t+1} being observed. Immediate rewards for the belief states are assigned using (9).

$$\begin{aligned} B_1^{t+1}(e_1^{t+1}) = \sum_{s^t} & B_1^t(e_1^t) \cdot P(s^t, (a_1^t, a_2^t), s^{t+1}) \cdot P^{\Pi_2}(a_2^t | \mathbf{\Psi_2^t}) \\ & \cdot O_2(s^{t+1}, (a_1^t, a_2^t), \omega_2^{t+1}) \cdot O_1(s^{t+1}, (a_1^t, a_2^t), \omega_1^{t+1}) \\ & / P(\omega_1^{t+1} | B_1^t, a_1) \end{aligned} \tag{8}$$

$$\Re(a_1^t, B_1^t) = \sum_{e_1^t} B_1^t(e_1^t) \cdot \sum_{a_2^t} R(s^t, (a_1^t, a_2^t)) \cdot P^{\Pi_2}(a_2^t | \mathbf{\Psi_2^t}) \tag{9}$$

Thus, RDR's policy generation implicitly coordinates the two agents, without communication or a correlational device. Randomized actions of one agent are planned taking into account the impact of randomized actions of its teammate on the joint reward. Algorithm 4 (at the end of paper) presents the pseudo-code for RDR for two agents, and shows how we can use beliefs over extended states e_i^t to construct LPs that maximize entropy while maintaining a certain expected reward threshold. The inputs to this algorithm are the parameters d, *percentdec* and $\bar{x}$. The parameter d specifies the number of iterations taken to solve the problem. It also decides the amount of reward that can be sacrificed at each step of the algorithm for improving entropy. Parameter *percentdec* specifies the

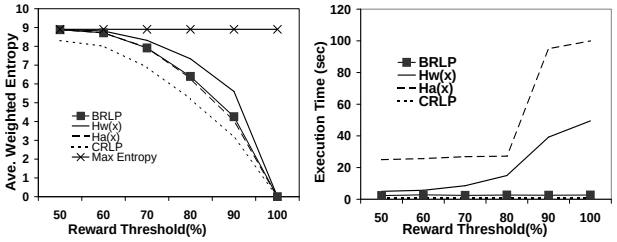
percentage of expected reward the agent team forgoes for improving entropy. As with Algorithm 2, parameter $\bar{x}$ provides an input solution with high randomness; and it is obtained using a uniform policy as discussed in Section 2.3. Step 1 of the algorithm uses the Compute joint policy() function which returns a optimal joint deterministic policy from which the individual policies and the expected joint reward are extracted. Obtaining this initial policy is not RDR's main thrust, and could be a global optimal policy obtained via [6] or a very high quality policy from random restarts of local algorithms such as [11, 5]. The input variable *percentdec* varies between 0 to 1 denoting the percentage of the expected reward that can be sacrificed by RDR. The step size as calculated in the algorithm, denotes the amount of reward sacrificed during each iteration. RDR then iterates 1/d times(step 3).

The function GenerateMDP generates all reachable belief states (lines 2 through 6) from a given starting belief state *B* and hence a belief MDP is generated. The number of such belief states is $O(|A_1||\Omega_1|)^{T-1}$ where T is the time horizon. The extended states in each B increases by a factor of $|A_2||\Omega_2|$ with increasing T. Thus the time to calculate B(e) for all extended states e, for all belief states B in agent 1's belief MDP is $O(|S|^2(|A_1||A_2||\Omega_1||\Omega_2|)^{T-1})$. Lines 7 through 12 compute the reward for each belief state. The total computations to calculate the reward is $O(|S||A_1||A_2|(|A_1||A_2||\Omega_1||\Omega_2|)^{T-1})$. The belief MDP generated is denoted by the tuple $\langle B, A, trans, R\rangle$. We reformulate the MDP obtained to problem 4 and use our polynomial BRLP procedure to solve it, using $\bar{x}$ as input. Thus, algorithm RDR is exponentially faster than an exhaustive search of a policy space, and comparable to algorithms that generate locally optimal policies [11].

5. EXPERIMENTAL RESULTS

We present three sets of experimental results. Our first set of experiments examine the tradeoffs in run-time, expected reward and entropy for single-agent problems. Figures 2a and 2b show the results for these experiments based on generation of MDP policies. The results show averages over 10 MDPs where each MDP represents a flight of a UAV, with state space of 28-40 states. The states represent the regions monitored by the UAVs. The transition function assumes that a UAV action can make a transition from a region to one of four other regions, where the transition probabilities were selected at random. The rewards for each MDP were also generated using random number generators. These experiments compare the performance of our four methods of randomization for single-agent policies. In the figures, *CRLP* refers to algorithm 2; *BRLP* refers to algorithm 3; whereas $H_W(x)$ and $H_A(x)$ refer to Algorithm 1 with these objective functions. Figure 2a examines the tradeoff between entropy and expected reward thresholds. It shows the average weighted entropy on the y-axis and reward threshold percent on the x-axis. The average maximally obtainable entropy for these MDPs is 8.89 (shown by line on the top) and three of our four methods (except CRLP) attain it at about 50% threshold, i.e. an agent can attain maximum entropy if it is satisfied with 50% of the maximum expected reward. However, if no reward can be sacrificed (100% threshold) the policy returned is deterministic.

Figure 2b shows the run-times, plotting the execution time in seconds on the y-axis, and expected reward threshold percent on the x-axis. These numbers represent averages over the same 10 MDPs as in Figure 2a. Algorithm CRLP is the fastest and its runtime is very small and remains constant over the whole range of threshold rewards as seen from the plot. Algorithm BRLP also has a fairly constant runtime and is slightly slower than CRLP. Both CRLP and BRLP are based on linear programs and hence their small and fairly constant runtimes. Algorithm 1, for both $H_A(x)$ and $H_W(x)$ objectives, exhibits an increase in the runtime as the expected reward threshold increases. This trend that can be attributed to the fact that maximizing a non-concave objective while simultaneously attaining feasibility becomes more difficult as the feasible region shrinks.



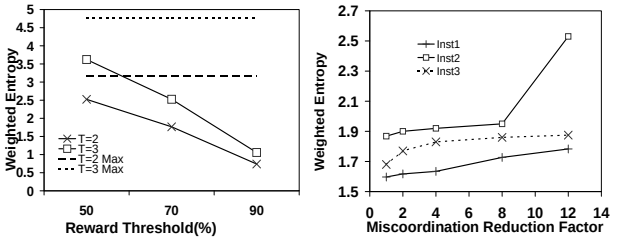
(a) (b)

Figure 2: Comparison of Single Agent Algorithms

We conclude the following from Figure 2: (i) CRLP is the fastest but provides the lowest entropy. (ii) BRLP is significantly faster than Algorithm 1, providing 7-fold speedup on average over the 10 MDPs over the entire range of thresholds. (iii) Algorithm 1 with $H_W(x)$ provides highest entropy among our methods, but the average gain in entropy is only 10% over BRLP. (iv) CRLP provides a 4-fold speedup on an average over BRLP but with a significant entropy loss of about 18%. In fact, CRLP is unable to reach the maximal possible entropy for the threshold range considered in the plot. Thus, BRLP appears to provide the most favorable tradeoff of run-time to entropy for the domain considered, and we use this method for the multiagent case. However, for time critical domains CRLP might be the algorithm of choice and therefore both BRLP and CRLP provide useful tradeoff points.

Reward Threshold	1	.5	.25	.125
90%	.67(.59)	1.73(.74)	3.47(.75)	7.07(.75)
50%	.67(1.53)	1.47(2.52)	3.4(2.62)	7.47(2.66)

Table 2: RDR: Avg. run-time in sec and (Entropy), $T = 2$



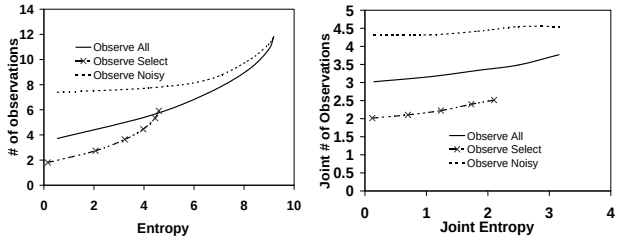
(a) (b)

Figure 3: Results for RDR

Our second set of experiments examine the tradeoffs in run-time, expected joint reward and entropy for the multiagent case. Table 2 shows the runtime results and entropy (in parenthesis) averaged over 10 instances of the UAV team problem based on the original transition and observation functions from [11] and its variations. d, the input parameter controlling the number of rolldown steps of algorithm 4, varies from 1 to 0.125 for two values of percent reward threshold (90% and 50%) and time horizon T=2. We conclude that as d decreases, the run-time increases, but the entropy remains fairly constant for $d \leq .5$. For example, for reward threshold of 50%, for d = 0.5, the runtime is 1.47 secs, but the run-time increases more than 5-fold to 7.47 when d = 0.125; however, entropy only changes from 2.52 to 2.66 with this change in d.

Thus in our next set of graphs, we present results for d = .5, as it provides the most favorable tradeoff, if other parameters remain fixed. Figure 3a plots RDR expected reward threshold percent on the x-axis and weighted entropy on the y-axis averaged over the same 10 UAV-team instances. Thus, if the team needs to obtain 90% of maximum expected joint reward with a time horizon $T = 3$, it gets a weighted entropy of 1.06 only as opposed to 3.62 if it obtains 50% of the expected reward for the same d and T. Similar to the single-agent case, the maximum possible entropy for the multiagent case is also shown by a horizontal line at the top of the graph. Figure 3b studies the effect of changing miscoordination cost on RDR's ability to improve entropy. As explained in Section 3.2, the UAV team incurs a high cost of miscoordination, e.g. if one UAV shoots left and the other shoots right. We now define miscoordination reduction factor (MRF) as the ratio between the original miscoordination cost and a new miscoordination cost. Thus, high MRF implies a new low miscoordination cost, e.g. an MRF of 4 means that the miscoordination cost is cut 4-fold. We plot this MRF on x-axis and entropy on y-axis, with expected joint reward threshold fixed at 70% and the time horizon T at 2. We created 5 reward variations for each of our 10 UAV team instances we used for 3a; only 3 instances are shown, to reduce graph clutter(others are similar). For *instance 2*, the original miscoordination cost provided an entropy of 1.87, but as this cost is scaled down by a factor of 12, the entropy increases to 2.53.

Based on these experiments, we conclude that: (i) Greater tolerance of expected reward loss allows higher entropy; but reaching the maximum entropy is more difficult in multiagent teams — for the reward loss of 50%, in the single agent case, we are able to reach maximum entropy, but we are unable to reach maximum entropy in the multiagent case. (ii) Lower miscoordination costs allow higher entropy for the same expected joint reward thresholds. (iii) Varying d produces only a slight change in entropy; thus we can use d as high as 0.5 to cut down runtimes. (iv) RDR is time efficient because of the underlying polynomial time BRLP algorithm.



(a) (b)

Figure 4: Improved security via randomization

Our third set of experiments examine the tradeoffs in entropy of the agent/agent-team and the total number of observations (probes) the enemy needs to determine the agent/agent-team actions at each state. The primary aim of this experiment is to show that maximizing policy entropy indeed makes it more difficult for the adversary to determine/predict our agents' actions, and thus more difficult for the adversary to cause harm, which was our main goal at the beginning of this paper. Figures 4a and 4b plot the number of observations of enemy as function of entropy of the agent/agent-team. In particular for the experiment we performed, the adversary runs yes-no probes to determine the agent's action at each state, i.e. probes that return an answer yes if the agent is taking the particular action at that state in which case the probing is stopped, and a no otherwise. The average number of yes-no probes at a state is the total number of observations needed by the adversary to determine the correct action taken by the agent in that state. The more deterministic the policy is, the fewer the probes the adversary needs to run; if the policy is completely deterministic, the adversary need not run any probes as it knows the action. Therefore, the aim of the agent/agent-team is to maximize the policy entropy, so that the expected number of probes asked by the adversary is maximized.

In contrast, the adversary minimizes the expected number of probes required to determine the agents' actions. Hence, for any given state s, the adversary uses the Huffman procedure to optimize the number of probes [8], and hence the total number of probes over the entire MDP state space can be expressed as follows. Let $S = \{s_1, s_2, ..., s_m\}$ be the set of MDP states and $A = \{a_1, a_2, ..., a_n\}$ be the action set at each state. Let $p(s, a) = \{p_1, ..., p_n\}$ be the probabilities of taking the action set $\{a'_1, ..., a'_n\}$, $a'_i \in A$ at state s sorted in decreasing order of probability. The number of yes-no probes at state s is denoted by $\zeta_s = p_1 * 1 + ... + p_{n-1} * (n-1) + p_n * (n-1)$. If the weight of the states (see notion of weight introduced in section 2) is $W = \{w_1, w_2, ..., w_m\}$, then the number of observations over the set of states is denoted *Observe-all* $= \sum_{s=1...m} \{w_s * \zeta_s\}$. Setting some weights to zero implies that the adversary was not concerned with those states, and the number of observations in this situation is denoted *Observe-select*. While the number of observations in both Observe-all and Observe-select are obtained assuming the adversary obtains an accurate policy of the agent or agent team, in real situations, an adversary may obtain a noisy policy, and the adversary's number of observations in such a case is denoted Observe-noisy.

Figure 4a demonstrates the effectiveness of entropy maximization using the BRLP method against an adversary using yes-no probes procedure as his probing method for the single agent case. The plot shows the number of observations on y-axis and entropy on the x-axis averaged over the 10 MDPs we used for our single-agent experiment. The plot has 3 lines corresponding to the three adversary procedures namely *Observe-all*, *Observe-select* and *Observe-noisy*. *Observe-all* and *Observe-select* have been plotted to study the effect of entropy on the number of probes the adversary needs. For example, for *Observe-all*, when entropy is 8, the average number of probes needed by the adversary is 9. The purpose of the *Observe-noisy* plot is to show that the number of probes that the adversary requires can only remain same or increase when using a noisy policy, as opposed to using the correct agent policy. The noise in our experiments is that two actions at each state of the MDP have incorrect probabilities. Each data point in the *Observe-noisy* case represents an average of 50 noisy policies, averaging over 5 noisy policies for each reward threshold over each of the 10 MDPs.

Figure 4b plots a similar graph as 4a for the multiagent case, averaged over the 10 UAV-team instances with two UAVs. The plot has three lines namely *Observe-all*, *Observe-select* and *Observe-noisy* with the same definitions as in the single agent case but in a distributed POMDP setting. However, in plot 4b the y-axis represents joint number of yes-no probes and the x-axis represents joint entropy. Both these parameters are calculated as weighted sums of the individual parameters for each UAV, assuming that the adversary assigns equal weight to both the UAVs.

We conclude the following from plots 4a and 4b: (i) The number of observations(yes-no probes) increases as policy entropy increases, whether the adversary monitors the entire state space (*observe-all*) or just a part of it (*observe-select*). (ii) If the adversary obtains a noisy policy (*observe-noisy*), it needs a larger number of observations when compared to the adversary obtaining an accurate policy. (iii) As entropy increases, the agents' policy tends to become more uniform and hence the effect of noise on the number of yes-no probes reduces. In the extreme case where the policy is totally uniform the *Observe-all* and *Observe-noisy* both have same number of

probes. This can be observed at the maximal entropy point in plot 4a. The maximal entropy point is not reached in plot 4b as shown in the results for RDR. From the above we conclude that maximizing entropy has indeed made it more difficult for the adversary to determine our agents' actions and cause harm.

6. SUMMARY AND RELATED WORK

This paper focuses on security in multiagent systems where intentional threats are caused by unseen adversaries, whose actions and capabilities are unknown, but the adversaries can exploit any predictability in our agents' policies. Policy randomization for single-agent and decentralized (PO)MDPs, with some guaranteed expected rewards, are critical in such domains. To this end, this paper provides two key contributions: (i) provides novel algorithms, in particular the polynomial-time CRLP and BRLP algorithms, to randomize single-agent MDP and POMDP policies, while attaining a certain level of expected reward; (ii) provides RDR, a new algorithm to generate randomized policies for decentralized POMDPs. RDR can be built on BRLP or CRLP, and thus is able to efficiently provide randomized policies. Finally, while our techniques are applied for analyzing randomization-reward tradeoffs, they could potentially be applied more generally to analyze different tradeoffs between competing objectives in single/decentralized (PO)MDP. More details about this work can be accessed at http://www-scf.usc.edu/ paruchur/japan.ppt.

Randomization as a goal has received little attention in the literature, and is seen as a means or side-effect in attaining other objectives, e.g., in resource-constrained MDPs [1] or memoryless POMDP policy generators [9] (for breaking loops). In [13] coordination of multiple agents executing randomized policies in a distributed MDP setting is discussed, but there randomization occurs as a side-effect of resource constraints; furthermore, agents communicate to resolve the resulting team miscoordination. In contrast, our work explicitly emphasizes maximizing entropy in policies, and attains implicit coordination (no communication) over randomized policies in decentralized POMDPs. Significant attention has been paid to learning in stochastic games, where agents must learn dominant strategies against explicitly modeled adversaries [10, 7]. Such dominant strategies may lead to randomization, but randomization itself is not the goal. Our work in contrast does not require an explicit model of the adversary and, in this worst case setting, hinders any adversary's actions by increasing the policy's weighted entropy through efficient algorithms, such as BRLP.

Acknowledgements : This research is supported by NSF grants #0208580, #0222914 & ISF #8008. It is also supported by the United States Department of Homeland Security through Center for Risk and Economic Analysis of Terrorism Events (CREATE). Sarit Kraus is also affiliated with UMIACS.

7. REFERENCES

[1] E. Altman. *Constrained Markov Decision Process*. Chapman and Hall, 1999.

[2] R. Beard and T. McLain. Multiple uav cooperative search under collision avoidance and limited range communication constraints. In *IEEE CDC*, 2003.

[3] D. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of MDPs. In *UAI*, 2000.

[4] D. Dolgov and E. Durfee. Constructing optimal policies for agents with constrained architectures. Technical report, Univ of Michigan, CSE-TR-476-03, 2003.

[5] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *AAMAS*, 2004.

[6] E. Hansen, D. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, 2004.

[7] J. Hu and P. Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. In *ICML*, 1998.

[8] D. A. Huffman. A method for the construction of minimum redundancy codes. In *Proc. IRE 40*, 1952.

[9] T. Jaakkola, S. Singh, and M. Jordan. Reinforcement learning algorithm for partially observable markov decision problems. *Advances in NIPS*, 7, 1994.

[10] M. Littman. Markov games as a framework for multi-agent reinforcement learning. In *ML*, 1994.

[11] R. Nair, D. Pynadath, M. Yokoo, M. Tambe, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*, 2003.

[12] S. Paquet, L. Tobin, and B. Chaib-draa. An online POMDP algorithm for complex multiagent environments. In *AAMAS*, 2005.

[13] P. Paruchuri, M. Tambe, F. Ordonez, and S. Kraus. Towards a formalization of teamwork with resource constraints. In *AAMAS*, 2004.

[14] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *JAIR*, 16:389–423, 2002.

[15] C. Shannon. A mathematical theory of communication. *The Bell Labs Technical Journal*, pages 379–457,623,656, 1948.

[16] S. Vavasis. Nonlinear optimization: Complexity issues. In *University Press, New York*, 1991.

Algorithm 4 RDR$(d, percentdec, \bar{x})$

$\pi_1, \pi_2, Optimalreward \leftarrow Compute\_joint\_policy()$
$stepsize \leftarrow percentdec \cdot Optimalreward \cdot d$
for $i \leftarrow 1$ **to** $1/d$ **do**
  $MDP \leftarrow GenerateMDP(b, \Pi_{(i+1)\text{Mod}_2}, T)$
  $Entropy, \Pi_{i\text{Mod}_2} \leftarrow BRLP(Optimalrew - stepsize * i, \bar{x})$

GenerateMDP(b, π_2, T) :
reachable$(0) \leftarrow \{b\}$
for $t \leftarrow 1$ **to** T **do**
  for all $B^{t-1} \in$ *reachable*$(t-1)$ **do**
    for all $a_1 \in A_1, \omega_1 \in \Omega_1$ **do**
      $trans, reachable(t) \overset{\cup}{\leftarrow}$ UPDATE(B^{t-1}, a_1, ω_1)
for $t \leftarrow T$ **downto** 1 **do**
  for all $B^t \in$ *reachable*(t) **do**
    for all $a_1 \in A_1$ **do**
      for all $s^t \in S, \psi_2^t \leftarrow \langle a_2^{t-1}, \omega_2^t \rangle$ **do**
        for all a_2^t given $\mathbf{\Psi_2^t}$ **do** {Equation 9}
          $\Re_t^{a_1}(B^t) \overset{+}{\leftarrow} B^t(s^t, \mathbf{\Psi_2^t}) \cdot R\left(s^t, \langle a_1^t, a_2^t \rangle\right) . P(a_2^t | \mathbf{\Psi_2^t})$
return $\langle B, A, trans, \Re \rangle$

UPDATE(B^t, a_1, ω_1) :
for all $s^{t+1} \in S, \psi_2^t \leftarrow \langle a_2^{t-1}, \omega_2^t \rangle$ **do**
  for all $s^t \in S$ **do** {Equation 8}
    $B^{t+1}(s^{t+1}, \mathbf{\Psi_2^{t+1}}) \overset{+}{\leftarrow} B^t(s^t, \mathbf{\Psi_2^t}) \cdot P(s^t, (a_1^t, a_2^t), s^{t+1}) \cdot O_1(s^{t+1}, (a_1^t, a_2^t), \omega_1^{t+1}) \cdot O_2(s^{t+1}, (a_1^t, a_2^t), \omega_2^{t+1}) \cdot P(a_2^t | \mathbf{\Psi_2^t})$
  $trans \leftarrow normalize(B^{t+1}(s^{t+1}, \mathbf{\Psi_2^{t+1}}))$
return $trans, B^{t+1}$

Real-Time Adaptive A*

Sven Koenig
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781
skoenig@usc.edu

Maxim Likhachev
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213-3891
maxim+@cs.cmu.edu

ABSTRACT

Characters in real-time computer games need to move smoothly and thus need to search in real time. In this paper, we describe a simple but powerful way of speeding up repeated A* searches with the same goal states, namely by updating the heuristics between A* searches. We then use this technique to develop a novel real-time heuristic search method, called Real-Time Adaptive A*, which is able to choose its local search spaces in a fine-grained way. It updates the values of all states in its local search spaces and can do so very quickly. Our experimental results for characters in real-time computer games that need to move to given goal coordinates in unknown terrain demonstrate that this property allows Real-Time Adaptive A* to follow trajectories of smaller cost for given time limits per search episode than a recently proposed real-time heuristic search method [5] that is more difficult to implement.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Graph and Tree Search Strategies*

General Terms

Algorithms

Keywords

A*; Agent Planning; D* Lite; Games; Heuristic Search; Incremental Search; Perception, Action and Planning in Agents; Planning with the Freespace Assumption; Real-Time Decision Making

1. INTRODUCTION

Agents need to use different search methods than the off-line search methods often studied in artificial intelligence. Characters in real-time computer games, for example, need to move smoothly and thus need to search in real time. Real-time heuristic search methods find only the beginning of a trajectory from the current state of an agent to a goal state [9, 4]. They restrict the search to a small part of the state space that can be reached from the current state with a small number of action executions (local search space). The agent determines the local search space, searches it, decides how to move within it, and executes one or more actions along the resulting trajectory. The agent then repeats this process until it reaches a goal state. Real-time heuristic search methods thus do not plan all the way to a goal state which often results in smaller total search times but larger trajectory costs. Most importantly, real-time heuristic search methods can satisfy hard real-time requirements in large state spaces since the sizes of their local search spaces (= their lookaheads) are independent of the sizes of the state spaces and can thus remain small. To focus the search and prevent cycling, they associate heuristics with the states and update them between searches, which accounts for a large chunk of the search time per search episode. In this paper, we describe Real-Time Adaptive A*, a novel real-time heuristic search method. It is a contract anytime method [12] that is able to choose its local search spaces in a very fine-grained way, updates the heuristics of all states in the local search space and is able to do so very quickly. Our experimental results for goal-directed navigation tasks in unknown terrain demonstrate that Real-Time Adaptive A* follows trajectories of smaller cost for given time limits per search episode than a recently proposed real-time heuristic search method [5] because it updates the heuristics more quickly, which allows it to use larger local search spaces and overcompensate for its slightly less informed heuristics. At the same time, Real-Time Adaptive A* is easier to implement.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profi t or commercial advantage and that copies bear this notice and the full citation on the fi rst page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specifi c permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. MAIN IDEA

Our main idea is simple but powerful. Assume that one has to perform several A* searches with consistent heuristics in the same state space and with the same goal states but possibly different start states. Adaptive A* makes the heuristics more informed after each A* search in order to speed up future A* searches. We now explain the main idea behind Adaptive A*.

A* [3] is an algorithm for finding cost-minimal paths in state spaces (graphs). For every state s, the user supplies a heuristic $h[s]$ that estimates the goal distance of the state (= the cost of a cost-minimal path from the state to a goal state). The heuristics need to be consistent [10]. For every state s encountered during the search, A* maintains two values: the smallest cost $g[s]$ of any discovered path from the start state s_{curr} to state s (which is initially infinity), and an estimate $f[s] := g[s] + h[s]$ of the distance from the start state s_{curr} via state s to a goal state. A* then operates as follows: It maintains a priority queue, called open list, which initially contains only the start state s_{curr}. A* removes a state s with a smallest f-value from the priority queue. If state s is a goal state, it terminates. Otherwise, it expands the state, meaning that it updates the g-value of each successor state of the state and then inserts those successor states into the open list whose g-value decreased. It then repeats

the process. After termination, the g-value of every expanded state s is equal to the distance from the start state s_{curr} to state s.

We now explain how one can make the heuristics more informed after each A* search in order to speed up future A* searches. Assume that s is a state that was expanded during such an A* search. We can obtain an admissible (= non-overestimating) estimate of its goal distance $gd[s]$ as follows: The distance from the start state s_{curr} to any goal state via state s is equal to the distance from the start state s_{curr} to state s plus the goal distance $gd[s]$ of state s. It clearly can not be smaller than the goal distance $gd[s_{\text{curr}}]$ of the start state s_{curr}. Thus, the goal distance $gd[s]$ of state s is no smaller than the goal distance $gd[s_{\text{curr}}]$ of the start state s_{curr} (= the f-value $f[\bar{s}]$ of the goal state $\bar{s}$ that was about to be expanded when the A* search terminates) minus the distance from the start state s_{curr} to state s (= the g-value $g[s]$ of state s when the A* search terminates).

$$\begin{aligned} g[s] + gd[s] &\geq gd[s_{\text{curr}}] \\ gd[s] &\geq gd[s_{\text{curr}}] - g[s] \\ gd[s] &\geq f[\bar{s}] - g[s] \end{aligned}$$

Consequently, $f[\bar{s}] - g[s]$ provides an admissible estimate of the goal distance $gd[s]$ of state s and can be calculated quickly. More informed heuristics can be obtained by calculating and assigning this difference to every state that was expanded during the A* search and thus is in the closed list when the A* search terminates. (The states in the open list are not updated since the distance from the start state to these states can be smaller than their g-values when the A* search terminates.) We evaluated this idea experimentally in [7]. We now use it to develop a novel real-time heuristic search method, called Real-Time Adaptive A* (RTAA*), which reduces to the case discussed above if its lookahead is infinity.

3. NOTATION

We use the following notation to describe search tasks: S denotes the finite set of states. $s_{\text{curr}} \in S$ denotes the start state of the search, and *GOAL* $\subset S$ denotes the set of goal states. $A(s)$ denotes the finite set of actions that can be executed in state $s \in S$. $c[s,a] > \epsilon$ (for a given constant $\epsilon > 0$) denotes the cost of executing action $a \in A(s)$ in state $s \in S$, whereas *succ*$(s,a) \in S$ denotes the resulting successor state. The action costs c can increase during the search but we require the goal distances of all states to remain bounded from above by some constant.

4. REAL-TIME ADAPTIVE A*

Figure 1 shows the pseudo code of RTAA*. The legend explains the constants, variables, and functions that we will refer to in the following. The variables annotated with [USER] have to be initialized before RTAA* is called. s_{curr} needs to be set to the start state of the agent, c to the initial action costs, and h to the initial heuristics, which need to be consistent for the initial action costs, that is, need to satisfy $h[s] = 0$ for all goal states s and $h[s] \leq h[\textit{succ}(s,a)] + c[s,a]$ for all non-goal states s and actions a that can be executed in them [10]. Variables annotated with [A*] are updated during the call to astar() {03} (= line 3 in the pseudo code), which performs a (forward) A* search guided by the current heuristics from the current state of the agent toward the goal states until a goal state is about to be expanded or *lookahead* > 0 states have been expanded. After the A* search, we require $\bar{s}$ to be the state that was about to be expanded when the A* search terminated. We denote this state with $\bar{s}$ consistently throughout this paper. We require that $\bar{s} =$ *FAILURE* if the A* search terminated due to an empty open list, in which case there is no finite-cost trajectory from the current state to any goal state and RTAA* thus returns failure {05}. We require *CLOSED* to contain the states expanded during the A* search and the g-value $g[s]$ to be defined for all generated states s, including all expanded states. We define the f-values $f[s] := g[s] + h[s]$ for these states s. The expanded states s form the local search space, and RTAA* updates their heuristics by setting $h[s] := f[\bar{s}] - g[s] = g[\bar{s}] + h[\bar{s}] - g[s]$ {06-07}. (The heuristics of the other states are not changed.) RTAA* then executes actions along the trajectory found by the A* search until state $\bar{s}$ is reached (or, equivalently, a state is reached that was not expanded or, also equivalently, the local search space is left), *movements* > 0 actions have been executed, or the cost of an action on the trajectory increases {09-16}. It then repeats the process until it reaches a goal state, in which case it returns success {17}.

constants and functions

S	set of states of the search task, a set of states
GOAL	set of goal states, a set of states
$A(\,)$	sets of actions, a set of actions for every state
succ$(\,)$	successor function, a state for every state-action pair

variables

lookahead	number of states to expand at most, an integer larger than zero
movements	number of actions to execute at most, an integer larger than zero
s_{curr}	current state of the agent, a state [USER]
c	current action costs, a float for every state-action pair [USER]
h	current (consistent) heuristics, a float for every state [USER]
g	g-values, a float for every state [A*]
CLOSED	closed list of A* (= all expanded states), a set of states [A*]
$\bar{s}$	state that A* was about to expand when it terminated, a state [A*]

```
procedure realtime_adaptive_astar():
{01} while (s_curr ∉ GOAL) do
{02}   lookahead := any desired integer greater than zero;
{03}   astar();
{04}   if s̄ = FAILURE then
{05}     return FAILURE;
{06}   for all s ∈ CLOSED do
{07}     h[s] := g[s̄] + h[s̄] − g[s];
{08}   movements := any desired integer greater than zero;
{09}   while (s_curr ≠ s̄ AND movements > 0) do
{10}     a := the action in A(s_curr) on the cost-minimal trajectory from s_curr to s̄;
{11}     s_curr := succ(s_curr, a);
{12}     movements := movements − 1;
{13}     for any desired number of times (including zero) do
{14}       increase any desired c[s, a] where s ∈ S and a ∈ A(s);
{15}     if any increased c[s, a] is on the cost-minimal trajectory from s_curr to s̄ then
{16}       break;
{17} return SUCCESS;
```

Figure 1: Real-Time Adaptive A*

The values of *lookahead* and *movements* determine the behavior of RTAA*. For example, RTAA* performs a single A* search from the start state to a goal state and then moves the agent along the trajectory found by the A* search to the goal state if it always chooses infinity for *lookahead* and *movements* and no action costs increase.

We now prove several important properties of RTAA* that hold no matter how it chooses its values of *lookahead* and *movements*. We make use of the following known properties of A* searches with consistent heuristics: First, they expand every state at most once. Second, the g-values of every expanded state and state $\bar{s}$ are equal to the distance from the start state to state s and state $\bar{s}$, respectively. Thus, one knows cost-minimal trajectory from the start state to all expanded states and state $\bar{s}$. Third, the f-values of the series of expanded states over time are monotonically nondecreasing. Thus, $f[s] \leq f[\bar{s}]$ for all expanded states s and $f[\bar{s}] \leq f[s]$ for all generated states s that remained unexpanded.

THEOREM 1. *The heuristics of the same state are monotonically nondecreasing over time and thus indeed become more informed over time.*

Proof: Assume that the heuristic of state s is updated on line

{07}. Then, state s was expanded and it thus holds that $f[s] \leq f[\bar{s}]$. Consequently, $h[s] = f[s] - g[s] \leq f[\bar{s}] - g[s] = g[\bar{s}] + h[\bar{s}] - g[s]$ and the update cannot decrease the heuristic of state s since it changes the heuristic from $h[s]$ to $g[\bar{s}] + h[\bar{s}] - g[s]$. ■

THEOREM 2. *The heuristics remain consistent.*

Proof: We prove this property by induction on the number of A* searches. The initial heuristics are provided by the user and consistent. It thus holds that $h[s] = 0$ for all goal states s. This continues to hold since goal states are not expanded and their heuristics thus not updated. (Even if RTAA* updated the heuristic of state $\bar{s}$, it would leave the h-value of that state unchanged since $f[\bar{s}] - g[\bar{s}] = g[\bar{s}] + h[\bar{s}] - g[\bar{s}] = h[\bar{s}]$. Thus, the heuristics of goal states would remain zero even in that case.) It also holds that $h[s] \leq h[succ(s, a)] + c[s, a]$ for all non-goal states s and actions a that can be executed in them. Assume that some action costs increase on lines {13-14}. Let c denote the action costs before all increases and c' denote the action costs after all increases. Then, $h[s] \leq h[succ(s, a)] + c[s, a] \leq h[succ(s, a)] + c'[s, a]$ and the heuristics thus remain consistent. Now assume that the heuristics are updated on lines {06-07}. Let h denote the heuristics before all updates and h' denote the heuristics after all updates. We distinguish three cases:

- First, both s and $succ(s, a)$ were expanded, which implies that $h'[s] = g[\bar{s}] + h[\bar{s}] - g[s]$ and $h'[succ(s, a)] = g[\bar{s}] + h[\bar{s}] - g[succ(s, a)]$. Also, $g[succ(s, a)] \leq g[s] + c[s, a]$ since the A* search discovers a trajectory from the current state via state s to state $succ(s, a)$ of cost $g[s]+c[s, a]$ during the expansion of state s. Thus, $h'[s] = g[\bar{s}] + h[\bar{s}] - g[s] \leq g[\bar{s}]+h[\bar{s}]-g[succ(s, a)]+c[s, a] = h'[succ(s, a)]+c[s, a]$.
- Second, s was expanded but $succ(s, a)$ was not, which implies that $h'[s] = g[\bar{s}] + h[\bar{s}] - g[s]$ and $h'[succ(s, a)] = h[succ(s, a)]$. Also, $g[succ(s, a)] \leq g[s] + c[s, a]$ for the same reason as in the first case, and $f[\bar{s}] \leq f[succ(s, a)]$ since state $succ(s, a)$ was generated but not expanded. Thus, $h'[s] = g[\bar{s}] + h[\bar{s}] - g[s] = f[\bar{s}] - g[s] \leq f[succ(s, a)] - g[s] = g[succ(s, a)]+h[succ(s, a)]-g[s] = g[succ(s, a)]+h'[succ(s, a)] - g[s] \leq g[succ(s, a)] + h'[succ(s, a)] - g[succ(s, a)] + c[s, a] = h'[succ(s, a)] + c[s, a]$.
- Third, s was not expanded, which implies that $h'[s] = h[s]$. Also, $h[succ(s, a)] \leq h'[succ(s, a)]$ since the heuristics of the same state are monotonically nondecreasing over time. Thus, $h'[s] = h[s] \leq h[succ(s, a)] + c[s, a] \leq h'[succ(s, a)] + c[s, a]$.

Thus, $h'[s] \leq h'[succ(s, a)] + c[s, a]$ in all three cases and the heuristics thus remain consistent. ■

THEOREM 3. *The agent reaches a goal state.*

Proof: Assume that the heuristics are updated on lines {06-07}. Let h denote the heuristics of RTAA* before all updates and h' denote the heuristics after all updates. The heuristics of the same state are monotonically nondecreasing over time, according to Theorem 1. Assume that the agent moves from its current state s to some state s' (with $s \neq s'$) along a cost-minimal trajectory from state s to state $\bar{s}$. It holds that $h'[s] = f[\bar{s}] - g[s] = f[\bar{s}]$ since state s is the start of the search and its g-value is thus zero and since it was expanded and its heuristic was thus updated. Furthermore, it holds that $h'[s'] = f[\bar{s}] - g[s']$ since either state s' was expanded and its heuristic was thus updated or $s' = \bar{s}$ and then $h'[s'] = h'[\bar{s}] = h[\bar{s}] = f[\bar{s}] - g[\bar{s}] = f[\bar{s}] - g[s']$. Thus, after the agent moved from state s to state s' and its current state thus changed from state s to state s', the heuristic of the current state decreased by $h'[s] - h'[s'] = f[\bar{s}] - (f[\bar{s}] - g[s']) = g[s']$ and the sum of the heuristics of all states but its current state thus increased by $g[s']$, which is bounded from below by a positive constant since $s \neq s'$ and we assumed that all action costs are bounded from below by a positive constant. Thus, the sum of the heuristics of all states but the current state of the agent increases over time beyond any bound if the agent does not reach a goal state. At the same time, the heuristics remain consistent, according to Theorem 2 (since consistent heuristics are admissible), and are thus no larger than the goal distances which we assumed to be bounded from above, which is a contradiction. Thus, the agent is guaranteed to reach a goal state. ■

THEOREM 4. *If the agent is reset into the start state whenever it reaches a goal state then the number of times that it does not follow a cost-minimal trajectory from the start state to a goal state is bounded from above by a constant if the cost increases are bounded from below by a positive constant.*

Proof: Assume for now that the cost increases leave the goal distances of all states unchanged. Under this assumption, it is easy to see that the agent follows a cost-minimal trajectory from the start state to a goal state if it follows a trajectory from the start state to a goal state where the h-value of every state is equal to its goal distance. If the agent does not follow such a trajectory, then it transitions at least once from a state s whose h-value is not equal to its goal distance to a state s' whose h-value is equal to its goal distance since it reaches a goal state according to Theorem 3 and the h-value of the goal state is zero since the heuristics remain consistent according to Theorem 2. We now prove that the h-value of state s is then set to its goal distance. When the agent executes some action $a \in A(s)$ in state s and transitions to state s', then state s is a parent of state s' in the A* search tree produced during the last call of astar() and it thus holds that (1) state s was expanded during the last call of astar(), (2) either state s' was also expanded during the last call of astar() or $s' = \bar{s}$, (3) $g[s'] = g[s] + c[s, a]$. Let h denote the heuristics before all updates and h' denote the heuristics after all updates. Then, $h'[s] = f[\bar{s}] - g[s]$ and $h'[s'] = f[\bar{s}] - g[s'] = gd[s']$. The last equality holds because we assumed that the h-value of state s' was equal to its goal distance and thus can no longer change since it could only increase according to Theorem 1 but would then make the heuristics inadmissible and thus inconsistent, which is impossible according to Theorem 2. Consequently, $h'[s] = f[\bar{s}] - g[s] = gd[s'] + g[s'] - g[s] = gd[s'] + c[s, a] \geq gd[s]$, proving that $h'[s] = gd[s]$ since a larger h-value would make the heuristics inadmissible and thus inconsistent, which is impossible according to Theorem 2. Thus, the h-value of state s is indeed set to its goal distance. After the h-value of state s is set to its goal distance, the h-value can no longer change since it could only increase according to Theorem 1 but would then make the heuristics inadmissible and thus inconsistent, which is impossible according to Theorem 2. Since the number of states is finite, it can happen only a bounded number of times that the h-value of a state is set to its goal distance. Thus, the number of times that the agent does not follow a cost-minimal trajectory from the start state to a goal state is bounded. The theorem then follows since the number of times that a cost increase does not leave the goal distances of all states unchanged is bounded since we assumed that the cost increases are

bounded from below by a positive constant but the goal distances are bounded from above. After each such change, the number of times that the agent does not follow a cost-minimal trajectory from the start state to a goal state is bounded. ■

5. RELATIONSHIP TO LRTA*

RTAA* is similar to a version of Learning Real-Time A* recently proposed in [5], an extension of the original Learning Real-Time A* algorithm [9] to larger lookaheads. For simplicity, we refer to this particular version of Learning Real-Time A* as LRTA*. RTAA* and LRTA* differ only in how they update the heuristics after an A* search. LRTA* replaces the heuristic of each expanded state with the sum of the distance from the state to a generated but unexpanded state s and the heuristic of state s, minimized over all generated but unexpanded states s. (The heuristics of the other states are not changed.) Let $\bar{h}'$ denote the heuristics after all updates. Then, the heuristics of LRTA* after the updates satisfy the following system of equations for all expanded states s:

$$\bar{h}'[s] = \min_{a \in A(s)} (c[s,a] + \bar{h}'[succ(s,a)])$$

The properties of LRTA* are similar to the ones of RTAA*. For example, its heuristics for the same state are monotonically nondecreasing over time and remain consistent, and the agent reaches a goal state. We now prove that LRTA* and RTAA* behave exactly the same if their lookahead is one and they break ties in the same way. They can behave differently for larger lookaheads, and we give an informal argument why the heuristics of LRTA* tend to be more informed than the ones of RTAA* with the same lookaheads. On the other hand, it takes LRTA* more time to update the heuristics and it is more difficult to implement, for the following reason: LRTA* performs one search to determine the local search space and a second search to determine how to update the heuristics of the states in the local search space since it is unable to use the results of the first search for this purpose, as explained in [5]. Thus, there is a trade-off between the total search time and the cost of the resulting trajectory, and we need to compare both search methods experimentally to understand this trade-off better.

THEOREM 5. *RTAA* with lookahead one behaves exactly like LRTA* with the same lookahead if they break ties in the same way.*

Proof: We show the property by induction on the number of A* searches. The heuristics of both search methods are initialized with the heuristics provided by the user and are thus identical before the first A* search. Now consider any A* search. The A* searches of both search methods are identical if they break ties in the same way. Let $\bar{s}$ be the state that was about to be expanded when their A* searches terminated. Let h denote the heuristics of RTAA* before all updates and h' denote the heuristics after all updates. Similarly, let $\bar{h}$ denote the heuristics of LRTA* before all updates and $\bar{h}'$ denote the heuristics after all updates. Assume that $h[s] = \bar{h}[s]$ for all states s. We show that $h'[s] = \bar{h}'[s]$ for all states s. Both search methods expand only the current state s of the agent and thus update only the heuristic of this one state. Since $s \neq \bar{s}$, it holds that $h'[s] = g[\bar{s}] + h[\bar{s}] - g[s] = g[\bar{s}] + h[\bar{s}]$ and $\bar{h}'[s] = \min_{a \in A(s)}(c[s,a] + \bar{h}'[succ(s,a)]) = \min_{a \in A(s)}(g[succ(s,a)] + \bar{h}'[succ(s,a)]) = \min_{a \in A(s)}(g[succ(s,a)] + \bar{h}[succ(s,a)]) = g[\bar{s}] + \bar{h}[\bar{s}] = g[\bar{s}] + h[\bar{s}]$. Thus, both search methods set the heuristic of the current state to the same value and then move to state $\bar{s}$. Notice that *lookahead* $= 1$ implies without loss of generality that *movements* $= 1$. Consequently, they behave exactly the same. ■

We now give an informal argument why the heuristics of LRTA* with lookaheads larger than one tend to be more informed than the ones of RTAA* with the same lookahead (if both real-time heuristic search methods use the same value of *movements*). This is not a proof but gives some insight into the behavior of the two search methods. Assume that both search methods are in the same state and break ties in the same way. Let h denote the heuristics of RTAA* before all updates and h' denote the heuristics after all updates. Similarly, let $\bar{h}$ denote the heuristics of LRTA* before all updates and $\bar{h}'$ denote the heuristics after all updates. Assume that $h[s] = \bar{h}[s]$ for all states s. We now prove that $h'[s] \leq \bar{h}'[s]$ for all states s. The A* searches of both search methods are identical if they break ties in the same way. Thus, they expand the same states and thus also update the heuristics of the same states. We now show that the heuristics h' cannot be consistent if $h'[s] > \bar{h}'[s]$ for at least one state s. Assume that $h'[s] > \bar{h}'[s]$ for at least one state s. Pick a state s with the smallest $\bar{h}'[s]$ for which $h'[s] > \bar{h}'[s]$ and pick an action a with $a = \arg\min_{a \in A(s)}(c[s,a] + \bar{h}'[succ(s,a)]$. State s must have been expanded since $h[s] = \bar{h}[s]$ but $h'[s] > \bar{h}'[s]$. Then, it holds that $\bar{h}'[s] = c[s,a] + \bar{h}'[succ(s,a)]$. Since $\bar{h}'[s] = c[s,a] + \bar{h}'[succ(s,a)] > \bar{h}'[succ(s,a)]$ and state s is a state with the smallest $\bar{h}'[s]$ for which $h'[s] > \bar{h}'[s]$, it must be the case that $h'[succ(s,a)] \leq \bar{h}'[succ(s,a)]$. Put together, it holds that $h'[s] > \bar{h}'[s] = c[s,a] + \bar{h}'[succ(s,a)] \geq c[s,a] + h'[succ(s,a)]$. This means that the heuristics h' are inconsistent but we have earlier proved already that they remain consistent, which is a contradiction. Consequently, it holds that $h'[s] \leq \bar{h}'[s]$ for all states s. Notice that this proof does not imply that the heuristics of LRTA* always dominate the ones of RTAA* since the search methods can move the agent to different states and then update the heuristics of different states, but it suggests that the heuristics of LRTA* with lookaheads larger than one tend to be more informed than the ones of RTAA* with the same lookaheads and thus that the trajectories of LRTA* tend to be of smaller cost than the trajectories of RTAA* with the same lookaheads (if both real-time heuristic search methods use the same value of *movements*).

In the remainder of the paper, we assume that the agents always choose the same constant for *lookahead*, which is an external parameter, and always use infinity for *movements*, both for LRTA* and RTAA*.

6. APPLICATION

Real-time heuristic search methods are often used as alternative to traditional search methods for solving offline search tasks [9]. We, however, apply RTAA* to goal-directed navigation in unknown terrain, a search task that requires agents to execute actions in real time. Characters in real-time computer games, for example, often do not know the terrain in advance but automatically observe it within a certain range around them and then remember it for future use. Figure 2 shows an example. To make these agents easy to control, the users can click on some position in known or unknown terrain and the agents then move autonomously to this position. If the agents observe during execution that their current trajectory is blocked, then they need to search for another plan. The searches need to be fast since the agents need to move smoothly even if the processor is slow, the other game components use most of the available processor cycles and there are a large number of agents that all have to search repeatedly. Thus, there is a time limit per A* search, which suggests that real-time heuristic search methods are a good fit for our navigation tasks. To apply them, we discretize the terrain into cells that are either blocked or unblocked, a common practice in the context of real-time computer games [1]. The agents initially do not know which cells are blocked but use a navigation strat-



Figure 2: Total Annihilation

egy from robotics [8]: They assume that cells are unblocked unless they have the cells already observed to be blocked (freespace assumption). They always know which (unblocked) cells they are in, observe the blockage status of their four neighboring cells, raise the action costs of actions that enter the newly observed blocked cells, if any, from one to infinity, and then move to any one of the unblocked neighboring cells with cost one. We therefore use the Manhattan distances as consistent heuristic estimates of the goal distances. The task of the agents is to move to the given goal cell, which we assume to be possible.

7. ILLUSTRATION

Figure 3 shows a simple goal-directed navigation task in unknown terrain that we use to illustrate the behavior of RTAA*. Black squares are blocked. All cells have their initial heuristic in the lower left corner. We first compare RTAA* with lookahead infinity to LRTA* with the same lookahead and forward A* searches. All search methods start a new search episode (= run another search) when the cost of an action on their current trajectory increases and break ties between cells with the same f-values in favor of cells with larger g-values and remaining ties in the following order, from highest to lowest priority: right, down, left and up. We do this since we believe that systematic rather than random tie-breaking helps the readers to understand the behavior of the different search methods better since all search methods then follow the same trajectories. Figures 4, 5, and 6 show the agent as a small black circle. The arrows show the planned trajectories from the current cell of the agent to the goal cell, which is in the lower right corner. Cells that the agent has already observed to be blocked are black. All other cells have their heuristic in the lower left corner. Generated cells have their g-value in the upper left corner and their f-value in the upper right corner. Expanded cells are grey and, for RTAA* and LRTA*, have their updated heuristics in the lower right corner, which makes it easy for the readers to compare them to the heuristics before the update in the lower left corner. Notice that forward A* searches, RTAA* with lookahead infinity and LRTA* with the same lookahead follow the same trajectories if they break ties in the same way. They differ only in the number of cell expansions, which is larger for forward A* searches (23) than for RTAA* (20) and larger for RTAA* (20) than for LRTA* (19). The first property is due to RTAA* and LRTA* updating the heuristics while forward A* searches do not. Thus, forward A*

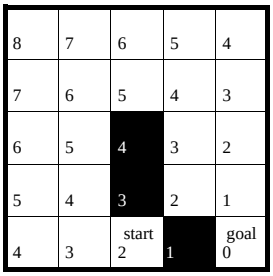


Figure 3: Example

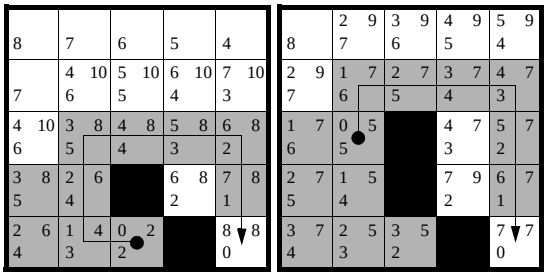


Figure 4: Forward A* Searches

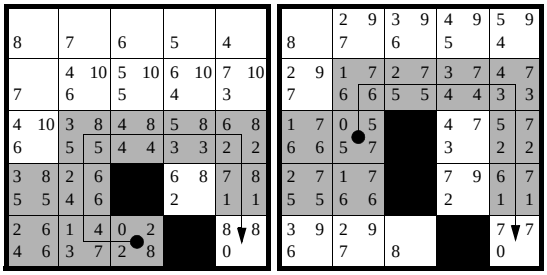


Figure 5: RTAA* with *lookahead* $= \infty$

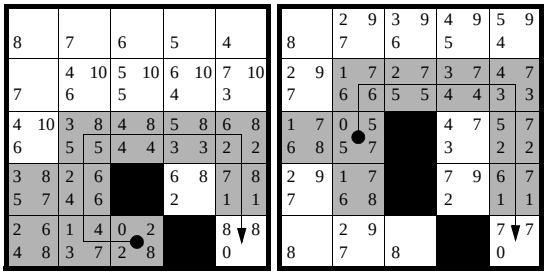


Figure 6: LRTA* with *lookahead* $= \infty$

searches fall prey to the local minimum in the heuristic value surface and thus expand the three leftmost cells in the lowest row a second time, while RTAA* and LRTA* avoid these cell expansions. The second property is due to some updated heuristics of LRTA* being larger than the ones of RTAA*. Notice, however, that most updated heuristics are identical, although this is not guaranteed in general. We also compare RTAA* with lookahead four to RTAA* with lookahead infinity. Figures 5 and 7 show that decreasing the lookahead of RTAA* increases the trajectory cost (from 10 to 12) but decreases the number of cell expansions (from 20 to 17) because smaller lookaheads imply that less information is used dur-

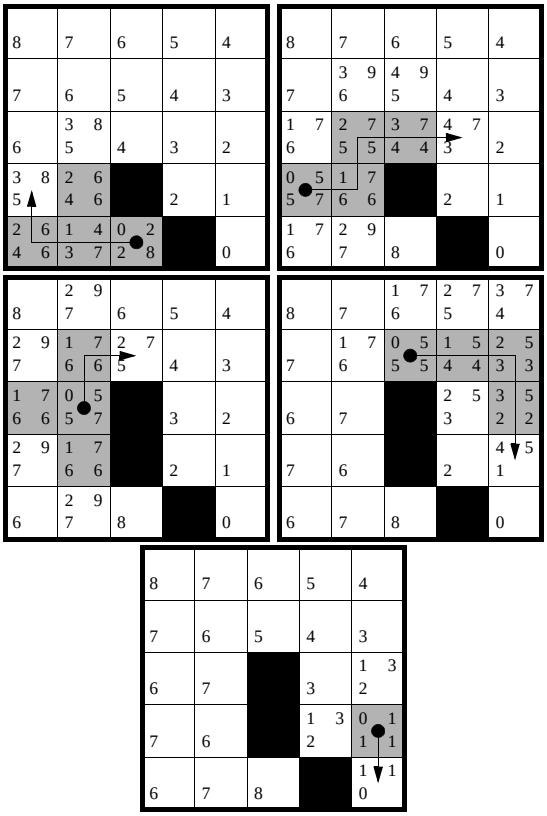


Figure 7: RTAA* with *lookahead* $= 4$

ing each search episode. (Notice that the last search episode of RTAA* with lookahead four expands only one cell since the goal cell is about to be expanded next.) We now perform systematic experiments to see whether they confirm the trends shown in our examples.

8. EXPERIMENTAL RESULTS

We now run experiments to compare RTAA* to forward A* searches and LRTA*, as in the previous section. All three search methods search forward. We also compare RTAA* to two search methods that search backward, namely backward A* searches and D* Lite [6] (an incremental version of backward A* searches that is similar to but simpler than D* [11]), which is possible because there is only one goal cell.[1] All search methods use binary heaps as priority queues and now break ties between cells with the same f-values in favor of cells with larger g-values (which is known to be a good tie-breaking strategy) and remaining ties randomly.

[1]Notice that the total number of cell expansions and total search time of forward A* searches are significantly smaller than the ones of backward A* searches. The big impact of the search direction can be explained as follows: The agent observes blocked cells close to itself. Thus, the blocked cells are close to the start of the search in the early phases of forward A* searches, but close to the goal of the search in the early phases of backward A* searches. The closer the blocked cells are to the start of the search, the more cells there are for which the Manhattan distances are perfectly informed and thus the faster A* searches are that break ties between cells with the same f-values in favor of cells with larger g-values since they expand only states along a cost-minimal trajectory from cells for which the Manhattan distances are perfectly informed to the goal of the search.

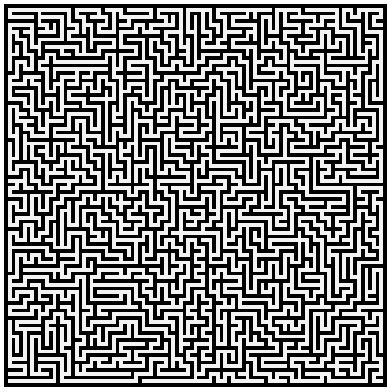


Figure 8: Test Terrain: Mazes

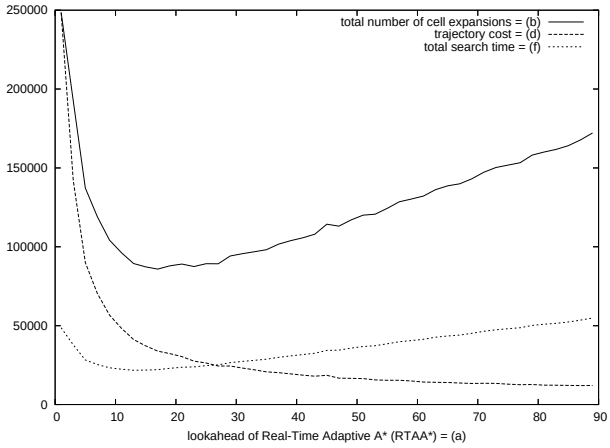


Figure 9: Performance of Real-Time Adaptive A*

We perform experiments on a SUN PC with a 2.4 GHz AMD Opteron Processor 150 in randomly generated four-connected mazes of size 151×151 that are solvable. Their corridor structure is generated with depth-first search. The start and goal cells are chosen randomly. Figure 8 shows an example (of smaller size than used in the experiments). We use the Manhattan distances as heuristics. The performance measures in Table 1 are averaged over the same 2500 grids. We show the standard deviation of the mean for three key performance measures in square brackets to demonstrate the statistical significance of our results, namely the total number of cell expansions and the total search time on one hand and the resulting trajectory cost on the other hand. We first verify the properties suggested in the previous sections:

- RTAA* with lookahead infinity, LRTA* with the same lookahead, D* Lite and forward and backward A* searches follow the same trajectories if they break ties in the same way and their trajectory costs are indeed approximately equal. (The slight differences are due to remaining ties being broken randomly.) The total number of cell expansions and the total search time are indeed larger for forward A* searches than RTAA*, and larger for RTAA* than LRTA*, as suggested in "Illustration."

- Decreasing the lookahead of RTAA* indeed increases the trajectory cost but initially decreases the total number of cell expansions and the total search time, as suggested in "Il-

Table 1: Experiments in Mazes

(a) = lookahead (= bound on the number of cell expansions per search episode), (b) = total number of cell expansions (until the agent reaches the goal cell) [in square brackets: standard deviation of the mean], (c) = total number of search episodes (until the agent reaches the goal cell), (d) = trajectory cost (= total number of action executions until the agent reaches the goal cell) [in square brackets: standard deviation of the mean], (e) = number of action executions per search episode, (f) = total search time (until the agent reaches the goal cell) in microseconds [in square brackets: standard deviation of the mean], (g) = search time per search episode in microseconds, (h) = search time per action execution in microseconds, (i) = increase of heuristics per search episode and expanded cell (= per update) = $h'[s] - h[s]$ averaged over all search episodes and expanded cells s

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
	Real-Time Adaptive A* (RTAA*)							
1	248537.79 [5454.11]	248537.79	248537.79 [5454.11]	1.00	48692.17 [947.08]	0.20	0.20	1.00
9	104228.97 [2196.69]	11583.50	56707.84 [1248.70]	4.90	23290.36 [360.77]	2.01	0.41	2.40
17	85866.45 [1701.83]	5061.92	33852.77 [774.35]	6.69	22100.20 [301.99]	4.37	0.65	3.10
25	89257.66 [1593.02]	3594.51	26338.21 [590.00]	7.33	24662.64 [310.42]	6.86	0.94	3.30
33	96839.84 [1675.46]	2975.65	22021.81 [521.77]	7.40	27994.45 [350.79]	9.41	1.27	3.28
41	105702.99 [1699.20]	2639.59	18628.66 [435.06]	7.06	31647.50 [382.30]	11.99	1.70	3.08
49	117035.65 [1806.59]	2473.55	16638.27 [390.09]	6.73	35757.69 [428.17]	14.46	2.15	2.90
57	128560.04 [1939.38]	2365.94	15366.63 [361.95]	6.49	39809.02 [477.72]	16.83	2.59	2.79
65	138640.02 [2019.98]	2270.38	14003.74 [314.38]	6.17	43472.99 [517.36]	19.15	3.10	2.63
73	150254.51 [2176.68]	2224.29	13399.01 [309.72]	6.02	47410.44 [567.88]	21.31	3.54	2.57
81	160087.23 [2269.20]	2172.94	12283.65 [270.03]	5.65	50932.60 [608.94]	23.44	4.15	2.39
89	172166.56 [2436.73]	2162.75	12078.40 [261.16]	5.58	54874.88 [663.96]	25.37	4.54	2.37
∞	642823.02 [20021.00]	1815.92	5731.20 [81.72]	3.16	226351.59 [7310.53]	124.65	39.49	4.22
	Learning Real-Time A* (LRTA*)							
1	248537.79 [5454.11]	248537.79	248537.79 [5454.11]	1.00	67252.19 [1354.67]	0.27	0.27	1.00
9	87613.37 [1865.31]	9737.38	47290.61 [1065.07]	4.86	27286.14 [437.09]	2.80	0.58	2.93
17	79312.59 [1540.44]	4676.76	30470.32 [698.08]	6.52	29230.15 [409.96]	6.25	0.96	3.61
25	82850.86 [1495.61]	3338.86	23270.38 [551.75]	6.97	34159.80 [450.49]	10.23	1.47	3.74
33	92907.75 [1548.37]	2858.19	20015.55 [472.86]	7.00	40900.68 [516.47]	14.31	2.04	3.71
41	102787.86 [1619.33]	2570.83	17274.12 [403.65]	6.72	47559.60 [587.96]	18.50	2.75	3.54
49	113139.63 [1716.88]	2396.66	15398.47 [360.45]	6.42	54324.06 [665.02]	22.67	3.53	3.38
57	125013.41 [1829.10]	2307.68	14285.14 [328.39]	6.19	61590.97 [744.33]	26.69	4.31	3.25
65	133863.67 [1956.49]	2201.60	13048.50 [300.69]	5.93	67482.95 [829.44]	30.65	5.17	3.12
73	146549.69 [2080.76]	2181.76	12457.92 [277.60]	5.71	74868.92 [909.31]	34.32	6.01	3.02
81	157475.45 [2209.65]	2150.04	11924.96 [262.61]	5.55	81469.32 [989.84]	37.89	6.83	2.95
89	166040.29 [2355.33]	2102.91	11324.72 [246.94]	5.39	86883.98 [1077.54]	41.32	7.67	2.88
∞	348072.76 [7021.57]	1791.19	5611.09 [80.43]	3.13	203645.42 [3782.37]	113.69	36.29	8.20
	D* Lite							
–	47458.83 [581.03]	1776.24	5637.46 [77.03]	3.17	37291.83 [378.20]	20.99	6.62	–
	Forward A* Search							
–	1857468.48 [68324.90]	1732.07	5354.26 [76.91]	3.09	544065.45 [21565.61]	314.11	101.61	–
	Backward A* Search							
–	5245087.82 [93697.15]	1795.72	5535.05 [77.09]	3.08	1698163.04 [31051.10]	945.67	306.80	–

lustration." Increasing the trajectory cost increases the total number of search episodes. If the lookahead is already small and continues to decrease, then eventually the speed with which the total number of search episodes increases is larger than the speed with which the lookahead and the time per search episode decreases, so that the number of cell expansions and the total search time increase again, as the graphs in Figure 9 show. (The graph also shows that the total number of cell expansions and the total search time are proportional, as expected.)

- RTAA* with lookaheads larger than one and smaller than infinity indeed increases the heuristics less than LRTA* with the same lookaheads per update, as suggested in "Relationship to LRTA*." Consequently, its trajectory costs and total number of cell expansions are larger than the ones of LRTA* with the same lookaheads. However, it updates the heuristics much faster than LRTA* with the same lookaheads, resulting in smaller total search times.

- RTAA* with lookahead one and LRTA* with the same lookahead follow the same trajectories and update the same states if they break ties in the same way and their trajectory costs and total number of cell expansions are indeed approximately equal, as suggested in "Relationship to LRTA*."

One advantage of RTAA* is that its total planning time with a carefully chosen lookahead is smaller than that of all other search methods, although its trajectory costs then are not the smallest ones. This is important for applications where planning is slow but actions can be executed fast. However, the advantage of RTAA* over the other search methods for our particular application is a different one: Remember that there is a time limit per search episode so that the characters move smoothly. If this time limit is larger than 20.99 microseconds, then one should use D* Lite for the search because the resulting trajectory costs are smaller than the ones of all other search methods whose search time per search episode is no larger than 20.99. (The table shows that the trajectory costs of forward A* search are smaller but, as argued before, this difference is due to noise.) However, if the time limit is smaller than 20.99 microseconds, then one has to use a real-time heuristics search method. In this case, one should use RTAA* rather than LRTA*. Assume, for example, that the time limit is 20.00 microseconds. Then one can use either RTAA* with a lookahead of 67, resulting in a trajectory cost of 13657.31, or LRTA* with a lookahead of 43, resulting in a trajectory cost of 16814.49. Thus, the trajectory cost of LRTA* is about 23 percent higher than the one of RTAA*, which means that RTAA* improves the state of the art in real-time heuristic search. A similar argument holds if the search time is amortized over the number of action executions, in which case there is a time limit per action execution. If this time limit is larger than 6.62 microseconds, then one should use D* Lite for the search because the resulting trajectory costs are smaller than the ones of all other search methods whose search time per action execution is no larger than 6.62. However, if the time limit is smaller than 6.62 microseconds, then one has to use a real-time heuristics search method. In this case, one should use RTAA* rather than LRTA*. Assume, for example,

that the time limit is 4.00 microseconds. Then one can use either RTAA* with a lookahead of 79, resulting in a trajectory cost of 12699.99, or LRTA* with a lookahead of 53, resulting in a trajectory cost of 14427.80. Thus, the trajectory cost of LRTA* is about 13 percent higher than the one of RTAA*, which again means that RTAA* improves the state of the art in real-time heuristic search.

9. CONCLUSIONS

In this paper, we developed Real-Time Adaptive A* (RTAA*). This real-time heuristic search method is able to choose its local search spaces in a fine-grained way. It updates the values of all states in its local search spaces and can do so very quickly. Our experimental results for goal-directed navigation tasks in unknown terrain demonstrated that this property allows RTAA* to move to the goal with smaller total search times than a variety of tested alternative search methods. Furthermore, we showed that RTAA* follows trajectories of smaller cost for given time limits per search episode than a recently proposed real-time heuristic search method because it updates the heuristics more quickly, which allows it to use larger local search spaces and overcompensate for its slightly less informed heuristics. It is future work to extend RTAA* to inconsistent heuristics, compare it to real-time heuristic search methods besides LRTA* and combine the idea behind RTAA* with other ideas of enhancing real-time heuristic search methods, for example, the ones described in [2] and [4]. In particular, it would be interesting to study a hierarchical version of RTAA*.

Acknowledgments

This research has been partly supported by an NSF award to Sven Koenig under contract IIS-0350584. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, companies or the U.S. government.

10. REFERENCES

[1] M. Bjornsson, M. Enzenberger, R. Holte, J. Schaeffer, and P. Yap. Comparison of different abstractions for pathfinding on maps. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1511–1512, 2003.

[2] V. Bulitko and G. Lee. Learning in real-time search: A unifying framework. *Journal of Artificial Intelligence Research*, page (in press), 2005.

[3] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 2:100–107, 1968.

[4] T. Ishida. *Real-Time Search for Learning Autonomous Agents*. Kluwer Academic Publishers, 1997.

[5] S. Koenig. A comparison of fast search methods for real-time situated agents. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems*, pages 864–871, 2004.

[6] S. Koenig and M. Likhachev. D* Lite. In *Proceedings of the National Conference on Artificial Intelligence*, pages 476–483, 2002.

[7] S. Koenig and M. Likhachev. Adaptive A* [poster abstract]. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1311–1312, 2005.

[8] S. Koenig, C. Tovey, and Y. Smirnov. Performance bounds for planning in unknown terrain. *Artificial Intelligence*, 147:253–279, 2003.

[9] R. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 1990.

[10] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1985.

[11] A. Stentz. The focussed D* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1652–1659, 1995.

[12] S. Zilberstein. *Operational Rationality through Compilation of Anytime Algorithms*. PhD thesis, Computer Science Department, University of California at Berkeley, Berkeley (California), 1993.

Winning back the CUP for distributed POMDPs: Planning over continuous belief spaces

Pradeep Varakantham, Ranjit Nair*, Milind Tambe, Makoto Yokoo**
University of Southern California, Los Angeles, CA 90089, {varakant, tambe}@usc.edu
* Automation and Control Solutions, Honeywell Laboratories, Minneapolis, MN 55418, ranjit.nair@honeywell.com
** Dept. of Intelligent Systems, Kyushu University, Fukuoka, 812-8581 Japan, yokoo@is.kyushu-u.ac.jp

ABSTRACT

Distributed Partially Observable Markov Decision Problems (Distributed POMDPs) are evolving as a popular approach for modeling multiagent systems, and many different algorithms have been proposed to obtain locally or globally optimal policies. Unfortunately, most of these algorithms have either been explicitly designed or experimentally evaluated assuming knowledge of a starting belief point, an assumption that often does not hold in complex, uncertain domains. Instead, in such domains, it is important for agents to explicitly plan over continuous belief spaces. This paper provides a novel algorithm to explicitly compute finite horizon policies over continuous belief spaces, without restricting the space of policies. By marrying an efficient single-agent POMDP solver with a heuristic distributed POMDP policy-generation algorithm, locally optimal joint policies are obtained, each of which dominates within a different part of the belief region. We provide heuristics that significantly improve the efficiency of the resulting algorithm and provide detailed experimental results. To the best of our knowledge, these are the first run-time results for analytically generating policies over continuous belief spaces in distributed POMDPs.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence - Multi Agent Systems

General Terms

Algorithms, Theory

Keywords

Multi-agent systems, Continuous initial beliefs, Partially Observable Markov Decision Process (POMDP), Distributed POMDP

1. INTRODUCTION

Distributed Partially Observable Markov Decision Problems (Distributed POMDPs) are evolving as a popular approach for modeling multiagent teamwork [13, 18, 14, 6, 3]. Distributed POMDPs provide a particularly useful model for domains ranging from sensor nets to disaster rescue, where agents need to work in a team in the presence of domain costs and uncertainties. Since optimal policy computation for these models has been proven to be NEXP-Complete [3], researchers have pursued two approaches in building algorithms for distributed POMDPs. The first approach restricts the domain, limiting agents' interactions (transition independence) [2, 14] or approximating observability of the local state [6]. The second approach, in-line with the thrust of this work, does not restrict the domains, but finds local optimal solutions [11, 7, 16].

Unfortunately, these approaches often assume knowledge of the starting belief state when generating policies[16, 11, 14, 10]. However, in many domains, including sensor nets and disaster rescue, such an assumption is a significant restriction, since it requires that starting belief state be known in advance. Alternatively, policy computation must be done on-line, which, given significant runtime for generating policies for distributed POMDPs, would prevent agents from generating a high-quality response in real-time.

Thus, there is a need in distributed POMDPs to generate policies for continuous belief spaces rather than just an initial single belief point, so policies may be computed off-line or without the need for a known starting belief state. This would allow agents to respond in real-time in time-critical domains. Unfortunately, while some of the approaches for locally optimal policy generation in distributed POMDPs can be applied or directly extended to apply to policy generation over continuous belief spaces[16, 7], these are based on finite-state controllers, which restrict policy representations; a restriction that is even more significant when planning over continuous belief spaces. Alternatively, a globally optimal policy generation algorithm has been suggested for continuous spaces[8], but this approach must surmount the substantial computational barrier given the NEXP-complete complexity result. In fact, no experimental evaluations are currently available for run-time efficiency when generating policies over continuous belief spaces with either of these approaches.

To remedy this situation, we present a novel algorithm for locally optimal policy generation for continuous starting belief region that does not restrict the space of policies considered. In particular, we build on the "Joint Equilibrium-based Search for Policies" (JESP) algorithm[11] which finds locally optimal policies from an unrestricted set of possible policies, with a finite planning horizon. However, JESP currently assumes a known starting belief state. To enable planning over continuous belief spaces, we combine JESP with algorithms that plan optimal policies over continuous belief spaces for single agent POMDPs [1, 5, 15]. In particular, whereas the original JESP performed iterative best-response computations from a single starting belief state, the combined algorithm exploits the single-agent POMDP techniques to perform best-response computations over continuous regions of the belief space.

The paper thus presents a new algorithm, CS-JESP (Continuous Space JESP), that allows us to generate a piece-wise linear and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

convex value function over continuous belief spaces for the optimal policy of one agent in the distributed POMDP, given fixed policies of other agents — the familiar cup-like shape of this value function is what we reference in the title of this paper [9]. The cup-shape implies that when dealing with a continuous starting belief space, agents usually have more than one policy, each of which dominates in a different region of the belief space. This region-wise dominance highlights the three important challenges addressed in CS-JESP. First, CS-JESP requires computation of best response policies for one agent, given that different policies dominate over different regions of the belief space for the second agent. To efficiently compute best response policies per belief region, it is critical to employ techniques that prune out unreachable future belief states. To that end, we illustrate application of the belief bound techniques[15] for improved efficiency. Second, owing to these best response calculations for different belief regions, often the policies for contiguous belief regions can be identical. To address this inefficiency, we implement a merging method that combines such adjacent regions with equivalent policies. Third, to improve the performance of the algorithm, we implement region-based convergence, i.e. once policies have converged for a region, these are not considered for subsequent best response computations.

2. BACKGROUND

As mentioned earlier, this paper presents a solution that combines value iteration algorithms of single agent POMDPs and JESP. In this section, we present a brief overview of these techniques.

2.1 Single Agent POMDPs and DB-GIP

A single-agent POMDP can be represented using the tuple $\left\langle \hat{\mathcal{S}}, \hat{\mathcal{A}}, \hat{\mathcal{P}}, \hat{\Omega}, \hat{\mathcal{O}}, \hat{\mathcal{R}} \right\rangle$, where $\hat{\mathcal{S}}$ is a finite set of states; $\hat{\mathcal{A}}$ is a finite set of actions; $\hat{\Omega}$ is a finite set of observations; $\hat{\mathcal{P}}(s, a, s')$ provides the probability of transitioning from state s to s' when taking action a; $\hat{\mathcal{O}}(s', a, o)$ is probability of observing o after taking an action a and reaching s'; $\hat{\mathcal{R}}(s, a)$ is the reward function. We use, $\hat{\mathcal{B}}$ to denote the feasible belief space for a POMDP. A belief point $b \in \hat{\mathcal{B}}$, is a probability distribution over the set of states $\hat{\mathcal{S}}$. A value function over a belief state is defined as:
$V(b) = \max_{a \in \hat{\mathcal{A}}} \left\{ \hat{\mathcal{R}}(b, a) + \gamma \, \Sigma_{b' \in \hat{\mathcal{B}}} \Pr(b'|b, a) V(b') \right\}$. Currently, the most efficient exact algorithms for POMDPs are value iteration algorithms, specifically GIP [1] and RBIP [5]. These are dynamic programming algorithms, where at each iteration the value function is represented with a minimal set of dominant vectors called the parsimonious set. Given a parsimonious set at time t, $\mathcal{V}_t$, we generate the parsimonious set at time $t-1$, $\mathcal{V}_{t-1}$ as follows (notation similar to that in [1] and [5]):

1. $\left\{ v_{t-1}^{a,o,i}(s) = \hat{\mathcal{R}}(s, a)/|\hat{\Omega}| + \gamma \, \Sigma_{s' \in \hat{\mathcal{S}}} \Pr(o, s'|s, a) v_t^i(s') \right\} = \hat{\mathcal{V}}_{t-1}^{a,o}$ where $v_t^i \in \mathcal{V}_t$.
2. $\mathcal{V}_{t-1}^{a,o} = PRUNE(\hat{\mathcal{V}}_{t-1}^{a,o})$
3. Doing crosssum ($\oplus$) of vectors .
   $\mathcal{V}_{t-1}^{a} = PRUNE(\cdots(PRUNE(\mathcal{V}_{t-1}^{a,o_1} \oplus \mathcal{V}_{t-1}^{a,o_2}) \cdots \oplus \mathcal{V}_{t-1}^{a,o_{|\hat{\Omega}|}})$
4. $\mathcal{V}_{t-1} = PRUNE(\bigcup_{a \in \hat{\mathcal{A}}} \mathcal{V}_{t-1}^{a})$

Each $PRUNE$ call executes a linear program (LP) which is recognized as a computationally expensive phase in the generation of parsimonious sets [1, 5]. As shown in [15], the performance of these dynamic programming algorithms can be improved considerably by exploiting the dynamics of the domain.

DB-GIP: This is an exact (not approximate) technique that builds on top of GIP, to provide significant performance improvements. There are two key ideas in DB-GIP (we focus on DB-GIP as it is shown to be more efficient than DB-RBIP)[15]. The first technique called DB, prunes unreachable belief spaces, given information about the starting belief space, B_0. To that end, it computes belief bounds over states at future time steps given B_0. These belief bounds are obtained by solving the maximization and minimization problems on the equation for belief probability as follows. Given an action a and observation ω, we can express the maximization problem as

$$\max_{b_t \in B_t} b_{t+1}^{a,\omega}(s_{t+1}) \quad \text{s.t.} \quad b_{t+1}^{a,\omega}(s_{t+1}) = c^T b_t / d^T b_t$$

where $c(s_t) = \hat{\mathcal{O}}_t(s_{t+1}, a, \omega)\hat{\mathcal{P}}_t(s_t, a, s_{t+1})$ and
$d(s_t) = \sum_{s_{t+1} \in \hat{\mathcal{S}}_{t+1}} \hat{\mathcal{O}}_t(s_{t+1}, a, \omega)\hat{\mathcal{P}}_t(s_t, a, s_{t+1})$.

Rewriting the problem in terms of the new variables as follows:

$$\min_x \left(-c^T x / d^T x \right) \quad \text{s.t.} \quad \sum_i x_i = 1, \; 0 \le x_i \le b_t^{\max}(s_i) =: \bar{x}_i$$

where $\sum_i b_t^{\max}(s_i) \ge 1$ to ensure existence of a feasible solution.

Expressing this problem as a Lagrangian enables efficient polynomial time solution that provides us a bound on the belief probability over a state at time step t+1, given the belief probabilities over states at time t. By bounding the belief probability over states, we can significantly improve the run-time of the LP, since the LP now performs the PRUNE operations mentioned above over narrower ranges of the belief space. DB leads to orders of magnitude improvement in existing algorithms [15] .

A second idea in DB-GIP is the Dynamic States (DS) technique, that exploits dynamic state spaces. Given an initial set of starting states, this technique initially finds the reachable states at each iteration, by using the transition function. Then, it finds dominant policies at each iteration over all belief probabilities of only these reachable states rather than over all belief probabilities of all the states. This reduction in reachable belief space, can significantly improve the performance of the existing algorithms[15]. DS in essence is a specialization of DB in that DS may be considered to impose a bound of 0 or 1 on the maximum belief probabilities over states, whereas DB may find a bound in between (e.g. 0.63).

2.2 Distributed POMDPs: MTDP

The distributed POMDP model that we base our work on is MTDP [17], however other models [3] could also be used. These distributed POMDP models are more than just two single agent POMDPs working independently. In particular, given a team of n agents, an MTDP [17] is defined as a tuple: $\langle S, A, P, \Omega, O, R \rangle$. S is a finite set of world states $\{s_1, \ldots, s_m\}$. $A = \times_{1 \le i \le n} A_i$, where $A_1, \ldots, A_n$, are the sets of action for agents 1 to n. A joint action is represented as $\langle a_1, \ldots, a_n \rangle$.
$P(s_i, \langle a_1, \ldots, a_n \rangle, s_f)$, the transition function, represents the probability that the current state is s_f, if the previous state is s_i and the previous joint action is $\langle a_1, \ldots, a_n \rangle$. $\Omega = \times_{1 \le i \le n} \Omega_i$ is the set of joint observations where Ω_i is the set of observations for agents i.
$O(s, \langle a_1, \ldots, a_n \rangle, \omega)$, the observation function, represents the probability of joint observation $\omega \in \Omega$, if the current state is s and the previous joint action is $\langle a_1, \ldots, a_n \rangle$. For the purpose of this paper, we assume that observations of each agent is independent of each other's observations. Thus the observation function can be expressed as $O(s, \langle a_1, \ldots, a_n \rangle, \omega) = O_1(s, \langle a_1, \ldots, a_n \rangle, \omega_1) \cdot \ldots \cdot O_n(s, \langle a_1, \ldots, a_n \rangle, \omega_n)$. The agents receive a single immediate joint reward $R(s, \langle a_1, \ldots, a_n \rangle)$ which is shared equally.

Each agent i chooses its actions based on its local *policy*, π_i,

which is a mapping of its observation history to actions. Thus, at time t, agent i will perform action $\pi_i(\vec{\omega}_i^t)$ where $\vec{\omega}_i^t = \omega_i^1, \ldots, \omega_i^t$. $\pi = \langle \pi_1, \ldots, \pi_n \rangle$ refers to the joint policy of the team of agents. In this model, execution is distributed but planning is centralized.

2.3 Illustrative Domain

We consider the standard multiagent tiger problem from [11]. Two agents are in a corridor facing two doors "left" and "right". Behind one door lies a hungry tiger, and behind the other lies a reward. The set of states, S, is {SL, SR}, where SL indicates tiger behind the left door, and SR indicates tiger behind right door. The agents can jointly or individually open either door. In addition, the agents can independently listen for the presence of the tiger. Thus, the set of actions, A1 = A2 = {'OpenLeft', 'OpenRight', 'Listen'}. The transition function, P specifies that the problem is reset whenever an agent opens one of the doors. However, if both agents listen, the state remains unchanged. After every action each agent receives an observation about the new state. The observation functions are identical and will return either TL or TR with different probabilities depending on the joint action taken and the resulting world state. For example, if both agents listen and the tiger is behind the left door (state is SL), each agent independently receives the observation TL with probability 0.85 and TR with probability 0.15. For more details on this domain, refer to [11].

2.4 JESP algorithm for MTDPs

Given the NEXP-complete complexity of generating globally optimal policies for distributed POMDPs[3], locally optimal approaches [16, 4, 11] have emerged as viable solutions. In this paper, we build on the "JESP" (Joint Equilibrium-Based Search for Policies) [11] algorithm, outlined below (Algorithm 1). The key idea is to find the policy that maximizes the joint expected reward for one agent at a time, keeping policies of the other $n-1$ agents fixed. This process is repeated until an equilibrium is reached (local optimum is found). Multiple local optima are not encountered since planning is centralized. Key innovation in JESP is based on the realization that if policies of all other $n-1$ agents are fixed, then the remaining agent faces a normal single-agent POMDP, but with an extended state space. Thus, in line 4, given a known starting belief state, we use dynamic programming over belief states of this newer more complex single-agent POMDP, to compute agent 1's optimal response to fixed policies of the remaining $n-1$ agents.

Algorithm 1 JESP()

```
1: Π' ← randomly selected joint policy, prevVal ← value of Π',
   conv ← 0, Π ← Π'
2: while conv ≠ n do
3:   for i ← 1 to n do
4:     val, Π_i ← OPTIMALBESTRESPONSE(b, Π', T)
5:     if val = prevVal then
6:       conv +← 1
7:     else
8:       Π'_i ← Π_i, prevVal ← val, conv ← 1
9:     if conv = n then break
10: return Π
```

The key is then to define the extended state in JESP. For a two agent case, for each time t, the extended state of agent 1 is defined as a tuple $e_1^t = \langle s^t, \vec{\omega}_2^t \rangle$, where $\vec{\omega}_2^t$ is the observation history of the other agent. By treating e_1^t as the state of agent 1 at time t, we can define the transition function and observation function for the resulting single-agent POMDP for agent 1 as follows:

$$\begin{aligned} P'(e_1^t, a_1^t, e_1^{t+1}) &= \Pr(e_1^{t+1} | e_1^t, a_1^t) \\ &= P(s^t, (a_1^t, \pi_2(\vec{\omega}_2^t)), s^{t+1}) \\ &\quad \cdot O_2(s^{t+1}, (a_1^t, \pi_2(\vec{\omega}_2^t)), \omega_2^{t+1}) \end{aligned} \quad (1)$$

$$\begin{aligned} O'(e_1^{t+1}, a_1^t, \omega_1^{t+1}) &= \Pr(\omega_1^{t+1} | e_1^{t+1}, a_1^t) \\ &= O_1(s^{t+1}, (a_1^t, \pi_2(\vec{\omega}_2^t)), \omega_1^{t+1}) \end{aligned} \quad (2)$$

In other words, when computing agent 1's best-response policy via dynamic programming given the fixed policy of its teammate, we maintain a distribution over the extended states e_1^t, rather than over the world states s^t. Figure 1 shows a trace of the belief state evolution for the multi-agent tiger domain, described in Section 2.3, e.g. e_1^2 of SL(TR) indicates an extended state where the tiger is behind the left door and agent 2 has observed TR. However, as noted above, the main shortcoming of this technique is that it computes a locally optimal policy assuming a fixed starting belief state, and this assumption is embedded in its dynamic programming as shown in line 4 of algorithm 1 — it does not generate policies over continuous belief spaces.

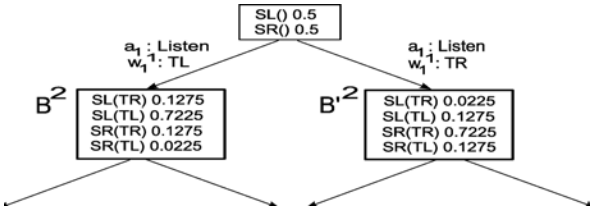


Figure 1: Trace of tiger scenario in JESP

3. CONTINUOUS SPACE JESP (CS-JESP)

One of the key insights in CS-JESP is the synergistic interaction between the JESP algorithm for distributed POMDPs and the DB-GIP technique of single agent POMDPs. We illustrate these interactions with a two-agent example in Section 3.1, and present key ideas in Section 3.2. Further, we describe the algorithm for n agents in Section 3.3 and some theoretical guarantees in Section 3.4.

Unlike previous work, our work focuses on continuous starting belief spaces and thus requires modifications for policy representation. In particular, because different policies may be dominant over different regions in the belief space, we introduce the notion of a *general policy*. A *general policy*, Π_i for an agent i is defined as a mapping from belief regions to policies. Π_i is represented as the set $\{(B_0^1, \pi_i^1), \ldots (B_0^m, \pi_i^m)\}$, where $B_0^1, .., B_0^m$ are belief regions in the starting belief space B_0 and $\pi_i^1, ..\pi_i^m$ are the policies that will be executed starting from those regions. Henceforth we refer π_i^k as *specialized policies*. Thus, given a starting belief point $b_0^k \in B_0^k$, agent i on receiving observations $\omega_i^1, \ldots, \omega_i^t$ will perform the action $\pi_i^k(\vec{\omega}_i^t)$ where $\vec{\omega}_i^t = \omega_i^1, \ldots, \omega_i^t$. $\Pi = \langle \Pi_1, \ldots, \Pi_n \rangle$ refers to the joint general policy of the team of agents.

3.1 Illustrative Example

For ease of explanation, initially the algorithm is explained with two agents, Agent1 and Agent2. However, as we will show in Section 3.3, this algorithm is easily extendable to n agents. Initially, each agent selects a random general policy, Π_i, which will be a singleton set, $\{(B_0, \pi_i)\}$, i.e. a single *specialized policy*,π_i, over the entire starting belief space, B_0. While for exposition purposes this example describes policy computations by individual agents,

in reality in CS-JESP these computations are performed by a centralized policy generator. CS-JESP begins when one agent, say Agent2, fixes its *general policy* Π_2, and other agent, Agent1, finds the best response for Agent2's *general policy*. Fixing Agent2's *specialized policy*, π_2, Agent1 creates a single agent POMDP with an extended state space, as explained in Section 2.4. Agent1 solves this POMDP using DB-GIP technique, explained in Section 2.1, with starting belief space as B_0, and obtains a new general policy Π_1, containing a set $\{(B_0^1, \pi_1^1), ...(B_0^m, \pi_1^m)\}$. Each $B_0^k \in B_0$ is a belief region and is represented by a minimum and maximum value for each of the $|S| - 1$ dimensions that represent the belief space. Now, Agent1 freezes its *general policy*, Π_1, and Agent2 solves a POMDP for each $\pi_1^j \in \Pi_1$, with the starting belief region as B^j. Thus, Agent2 solves m POMDPs, and obtains a new *general policy* Π_2. At this point, bordering regions in Π_2 that have identical policies are merged. This process continues until the solutions converge, and a local optimal is reached, i.e. no agent can improve its value vectors in any belief region.

Figure 2 illustrates the working of the algorithm with the multi-agent tiger scenario (Section 2.3) for time horizon, $T = 2$. Each tree in Figure 2 represents a *specialized policy*. All the trees on the left side of the figure are part of the *general policies* of Agent1, and trees on the right are part of the *general policies* of Agent2. For instance, at the end of iteration 3, both agents contain two specialized policies in their *general policy*. Within each tree (specialized policy), the letter inside each node indicates the action, and edges indicate the observation received. Thus, for the highlighted tree in the top left corner, the root node indicates the Listen(L) action, and upon either observing TL or TR, the *specialized policy* requires the agent to take a Listen(L) action. In this example, belief region over which a *specialized policy* dominates, consists of two numbers, namely the minimum and maximum belief probability of the state SL. These belief regions are indicated below each *specialized policy* in the figure. For instance, for the highlighted tree it is [0,1], but for other trees, regions such as [0.18,0.85] are shown.

The algorithm begins with both agents randomly selecting a *specialized policy* for the entire belief space [0,1]. In iteration 1, Agent2 fixes its *general policy*, and Agent1 comes up with its best response *general policy*. For calculating the best response, the Agent1 solves a POMDP with the starting belief range as [0,1], since Agent2's general policy is defined over this range. After the first iteration, Agent1 contains three *specialized policies* as part of its *general policy*, dominating over ranges [0,0.15], [0.15,0.85], [0.85,1]. In iteration 2, Agent1 fixes its *general policy*, and Agent2 begins its best response calculation with region [0,0.15]. For this range [0,0.15], Agent2 has only one dominant *specialized policy* and same is the case for [0.85,1]. However, for the range [0.15,0.85], Agent2 has two dominant *specialized policies*, one that dominates in the range [0.15,0.5], and the other that dominates in the range [0.5,0.85]. Thus after iteration 2, Agent2 has four *specialized policies* as part of its best response *general policy*. However, regions highlighted (with dotted rectangular boxes) have identical policies and thus after merging we are left with only two *specialized policies*. This algorithm continues with Agent2 fixing its *general policy* at iteration 3. Finally at convergence, each agent contains two *specialized policies* as part of their *general policies*.

3.2 Key Ideas

In this section, we explain in detail the key ideas in the CS-JESP algorithm, namely: (a) JESP and DB-GIP synergy; (b) Calculation of dominanant belief regions for *specialized policies*; (c) Region-based convergence; and (d) Merging of adjacent regions with identical *specialized policies*.

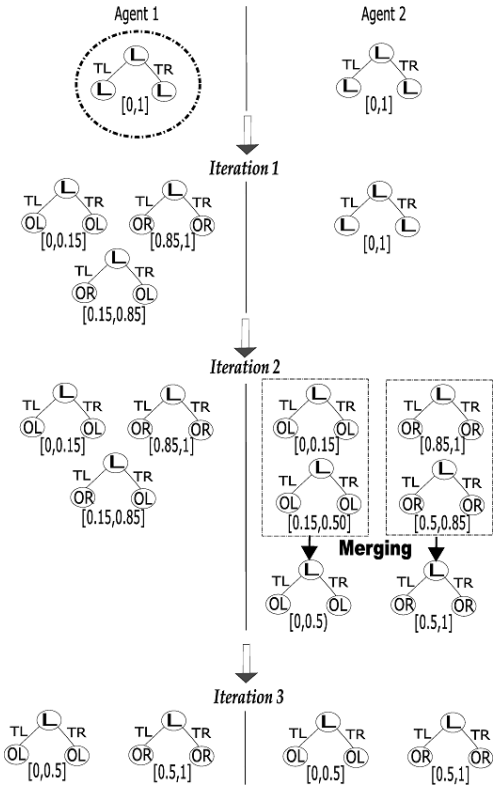


Figure 2: Trace of the algorithm for T=2 in Multi Agent tiger example with a specific starting joint policy

JESP and DB-GIP synergy: Both the DS and DB techniques of DB-GIP can provide significant performance improvements in CS-JESP. First, with respect to DS, JESP's state space is dynamic, where the set of states reachable at time t, e_i^t differ from the set of states at $t + 1$, e_i^{t+1}. DS can exploit this dynamism by computing dominant policies at time t over the belief space generated by the states in e_i^t thus reducing the dimensionality of the state space considered. For instance, in Figure 1, we have two initial states e_1^1= SL or SR, while there are four states e_1^2, e.g.$SL(TL), SL(TR)$ etc. Given a time horizon of T=2, instead of constructing a belief space over (2+4=) 6 dimensions, DS will lead to constructing a belief space over two states at the first time step and four states over the second time step. Such dimensionality reduction leads to significant speedups in CS-JESP. Second, with respect to DB, each agent solves a POMDP over the belief regions in the *general policies* of the other agents. DB is able to exploit forward projections of such starting belief regions to bound the maximum probabilities over states, and thus again restrict the belief space over which dominant policies are planned per belief region, obtaining additional speedups. For instance in Figure 2, at iteration2, Agent2 solves three POMDPs — these POMDPs are defined over extended states given three separate fixed policies of Agent1— one with the starting belief region as [0,0.15], another with [0.15,0.85], and a third with [0.85,1]. Thus, in solving the POMDP starting with the belief range [0,0.15], DB helps prune all the unreachable portions of the belief space given that the starting range is [0,0.15]. In all three POMDPs, the belief region is narrower compared to [0,1].

Region-based convergence: Given continuous initial belief space, we obtain value vectors (vector containing values for all the states)

for all the belief regions in the *general policy*. Thus, convergence is attained when for all agents the value vectors at the current iteration for all the belief regions are equal to those in the previous iteration. For instance, in Figure 2, the convergence is attained in the fourth iteration, with the *general policy* of Agent2 containing the two exact same *specialized policies* from iteration 3. However, once one region has converged — the value vectors for all agents do not change from one iteration to the next for that region — CS-JESP will not test that region further for convergence, but only continue changing policies in regions that have failed to converge.

Merging of adjacent regions with identical *specialized policies*: Merging such regions can be important as the other agent would have to solve fewer number of POMDPs in the next iteration. For instance, in the *general policy* of Agent2 before merging at iteration 2, belief regions [0,0.15] and [0.15,0.5] have identical specialized policies. Similarly, regions [0.5, 0.85] and [0.85,1] have identical specialized policies. Thus Agent2 has only two *specialized policies* after merging (instead of four before merging) and this leads to agent1 solving two instead of four POMDPs at iteration 3.

Merging requires identifying regions adjacent to each other. In the Tiger domain, this is done by doing adjacency check for regions along one dimension. However, finding bordering regions in a $|S|$ dimensional state space requires comparisons along $|S|-1$ dimensional space.

Calculation of dominant belief regions for *specialized policies*: One standard way of representing solutions in single agent POMDPs is through value vectors. In this representation, the best policy for a belief point, b, is computed by testing for a vector that provides the maximum expected value for that belief point.
$\pi_1^* \leftarrow argmax_{\pi \in \{\pi_1\}} v_\pi \cdot b$.

However, in CS-JESP, one agent uses the belief regions of the other agent to calculate the best responses over each of those belief regions. We develop a linear program to address the dominant belief region computation for each policy. Algorithm 2 computes the maximum belief probability of a state, s_j, where a policy or value vector,v dominates all the other policies or value vectors, $\mathcal{V} - v$ in the final policy. Constraint 1 in Algorithm 2, computes points where v dominates all the other vectors in $\mathcal{V}$. Objective function of the algorithm is a maximization over $b(s_j)$, thus finding highest possible belief probability for state s_j amongst all those dominating points. In a similar way, the minimum for s_j can be found by doing a minimize, instead of maximize, in line 1 of the LP. The belief region is calculated by solving these max, min LPs for each state $s_j \in S$. Thus, requiring $2 * |\mathcal{V}| * |S|$ number of LPs to be solved for the computation of an entire belief region.

Algorithm 2 MAXIMUMBELIEF$(s_j, v, \mathcal{V}, B^{min}, B^{max})$

Maximize $b(s_j)$
subject to constraints
1. $b.(v - v') > 0, \forall v' \in \mathcal{V} - v$
2. $\Sigma_{s \in S} b(s) = 1$
3. $B^{min}(s) < b(s) < B^{max}(s), \forall s \in S$

3.3 Algorithm for n agents

In this section, we present the CS-JESP algorithm (Algorithm 3) for n agents. In the initialization stage (lines 1-4), each agent i has only one belief region that corresponds to its entire belief space (Π'_i.*beliefPartition*). Also, each agent has a single randomly selected *specialized policy*, $\Pi'_i.\pi[\langle [0,1], \ldots, [0,1] \rangle]$ (i.e. π is the *specialized policy*), for the entire belief space (line 3). Every *general policy* has "*count*" for each belief region, to track the convergence of policies in that belief region (region-based convergence) — if the *count* reaches n then the region has converged, because no agent will change any further. The flag "*converged*" monitors if joint *general policies* in all the regions have converged.

In each iteration (one execution of lines 6-23) of Algorithm 3, we choose an agent i and find its optimal response to the fixed *general policies* of the remaining agents by calling OPTIMALBESTRESPONSE(). This is repeated until no agent acting alone can improve upon the joint expected reward by changing its own *general policy*.

Although each agent i starts off with the same belief set partition, Π'_i.*beliefPartition*, this will not be true after calling OPTIMALBESTRESPONSE() as seen in Figure 2. The function UPDATEPARTITION() (Algorithm 4) is responsible for creating a new belief set partition for an agent i, depending on the belief regions of the other $n-1$ agents. This new belief set partition is obtained by splitting the overlapping belief regions of the $n-1$ agents, in a way that no two resulting belief regions, which now belong to this partition, overlap. Furthermore, this function computes the Π_i.*count* for all the new regions, from the $count$ values for the regions in Π'_j, where j was the free agent in the last iteration (i.e the agent who computed the best response in the last iteration).

FINDNEWPARTITION() (Algorithm 5) takes two arguments, (i) *partition* and (ii) a belief region, br, and it generates all feasible partitions from the two arguments. To illustrate the working of this function, we provide an example with three states $\{s_1, s_2, s_3\}$. Belief regions in the corresponding belief space can be represented with minimum and maximum belief probabilities for just s_1 and s_2, i.e. $\{(b^{min}[s_1], b^{max}[s_1]), (b^{min}[s_2], b^{max}[s_2])\}$. For example, let *partition* $=\{\langle [0, 0.8], [0.5, 0.9] \rangle\}$ (has only one region) and $br = \{[0.4, 0.9], [0.3, 0.6]\}$. In the first step (line 3), partitions are found for each state, s_i separately. Thus, for the first state, s_1, $[0, 0.8]$ and $[0.4, 0.9]$ yields partitions, $[0, 0.4]$, $[0.4, 0.8]$, $[0.8, 0.9]$. Similarly for the second state, s_2, the partitions found are $[0.3, 0.5]$, $[0.5, 0.6]$,$[0.6, 0.9]$. In the second step (line 4), we compute the cross product of these individual dimension partitions. This gives rise to nine belief regions, viz. $\{[0, 0.4], [0.3, 0.5]\}, \ldots, \{[0.8, 0.9], [0.4, 0.8]\}$, $\{[0.8, 0.9], [0.8, 0.9]\}$. Finally, in the third step (line 5), we prune regions which do not contain any valid points, i.e. $\sum_{s \leq |S|-1} b^{min}[s] > 1$. For instance, the region {[0.8,0.9],[0.4,0.8]} can be pruned, because a belief point in this region has probability of atleast 1.2 $(= 0.8 + 0.4)$.

The function OPTIMALBESTRESPONSE() (Algorithm 6) is then called separately for each belief region in agent i's belief set partition. It returns a new partitioniong of the initial belief space and the optimal policy for each belief region in this partition. CONSTRUCTEXTENDEDPOMDP() constructs a POMDP with extended state space, as explained in Section 2.4, while the function CALCULATEBELIEFREGION() computes the belief regions where each vector v $(\in \mathcal{V})$ dominates.

After computing best responses, CS-JESP() ensures that the number of belief partitions obtained are finite (in lines 14-16) and that Π.*count* is updated correctly for each belief region (in lines 18-21).

It is possible that an agent's best response in adjacent belief regions is the same policy. The function MERGEBELIEFREGIONS() (Algorithm 3.3) is responsible for merging such kind of regions (lines 4-7). Further, once the policies in a belief region have converged, that region is not considered for subsequent merging phases (first part of the condition on line 4).

3.4 Theoretical Results

Algorithm 3 CS-JESP()

1: **for** $i \leftarrow 1$**to** n **do**
2: Π'_i.*beliefPartition* $\leftarrow \{\langle [0,1], \ldots, [0,1]\rangle\}$
3: $\Pi'_i.\pi[\langle [0,1], \ldots, [0,1]\rangle] \leftarrow$ random specialized policy
4: Π'_i.*count*$[\langle (0,1), \ldots, (0,1)\rangle] \leftarrow 0$
5: *converged* $\leftarrow$ false; $i \leftarrow n$;
6: **while** *converged* $=$ false **do**
7: $i \leftarrow (i$ *MOD* $n) + 1$; *converged* $\leftarrow$ true
8: UPDATEPARTITION(i, Π_i, Π')
9: **for all** br **in** Π_i.*beliefPartition* **do**
10: **if** Π_i.*count*$[br] < n$ **then**
11: *converged*$\leftarrow$false
12: $\{\Pi_i,$ *regions*$\} \leftarrow$ OPTIMALBESTRESPONSE(i, Π', br)
13: **for all** br_1 **in** *regions* **do**
14: $\pi \leftarrow \Pi_i.\pi[br_1]$; REMOVE$(\Pi_i, br_1)$
15: **for** *dim* $\leftarrow 1$ **to** $|S| - 1$ **do**
16: br_1[*dim*] $\leftarrow$ ROUNDOFF$(br_1$[*dim*], *precision*$)$
17: **if** VOLUME$(br_1) > 0$ **then**
18: ADD$(\Pi_i$.*beliefPartition*$, br_1, \pi)$
19: **if** $\Pi_i.\pi[br_1] = \Pi'_i.\pi[br]$ **then**
20: Π_i.*count*$[br_1] \leftarrow \Pi'_i$.*count*$[br] + 1$
21: **else**
22: Π_i.*count*$[br_1] \leftarrow 1$
23: MERGEBELIEFREGIONS(Π_i)
24: $\Pi'_i \leftarrow \Pi_i$
25: **return** Π

Algorithm 4 UPDATEPARTITION(i, Π)

1: $\Pi_i \leftarrow \Pi_{(i \text{ MOD } n)+1}$
2: **for all** j **in**$\{1, \ldots, n\} - \{i, (i$ *MOD* $n) + 1\}$ **do**
3: **for all** br_1 **in** Π_j.*beliefPartition* **do**
4: **if** Π_i.*count*$[br_1] < n$ **then**
5: Π_i.*beliefPartition* $\leftarrow$ FINDNEWPARTITION$(\Pi_i$.*beliefPartition*$, br_1)$
6: **if** $i = 1$ **then** $j \leftarrow n$**else** $j \leftarrow i - 1$
7: **for all** br_2 **in** Π_i.*beliefPartition* **do**
8: $br_3 \leftarrow$ OVERLAPPINGREGION$(\Pi_j$.*beliefPartition*$, br_2)$
9: Π_i.*count*$[br_2] \leftarrow \Pi_j$.*count*$[br_3]$
10: **return**

Algorithm 5 FINDNEWPARTITION((*partition*, br))

1: *newPartition* $\leftarrow \emptyset$
2: **for** *dim* $\leftarrow 1$ **to** $|S| - 1$ **do**
3: *1DPartition* $\leftarrow$ SPLITDIMENSION(*dim*, br, *partition*)
4: *newPartition* $\leftarrow$ CROSSPRODUCT(*newPartition*, *1DPartition*)
5: *newPartition* $\leftarrow$ PRUNE(*newPartition*)
6: **return** *newPartition*

Algorithm 6 OPTIMALBESTRESPONSE(i, Π', br)

1: $k \leftarrow 0$
2: *extendedPOMDP* $\leftarrow$ CONSTRUCTEXTENDEDPOMDP(i, Π', br)
3: $\{\mathcal{V}, \pi^{new}\} \leftarrow$ DB-GIP(*extendedPomdp*, br)
4: **for** $j \leftarrow 1$ **to** $\mathcal{V}.size$ **do**
5: $v \leftarrow \mathcal{V}[j]$; $\mathcal{V}' \leftarrow \mathcal{V} - v$
6: *beliefPartition*$[k] \leftarrow$ CALCULATEBELIEFREGION$(v, \mathcal{V}', br)$
7: Π_i.*beliefPartition*$[k] \leftarrow$ *beliefPartition*$[k]$
8: $\Pi_i.\pi$[*beliefPartition*$[k]] \leftarrow \pi^{new}[j]$; $k \stackrel{+}{\leftarrow} 1$
9: **return** $\{\Pi_i,$ *beliefPartition*$\}$

In the following proofs, we use "iteration" to mean one execution of the "while" loop (lines 5-23) of Algorithm 3, n for the number of agents, and "free agent" to denote the ith agent for that iteration.

Algorithm 7 MERGEBELIEFREGIONS(Π_i)

1: **for each** b_1 **in** Π_i.*beliefPartition* **do**
2: **if** Π_i.*count*$(b_1) < n$ **then**
3: **for each** b_2 **in** Π_i.*beliefPartition* **do**
4: **if** Π_i.*count*$(b_2) < n \wedge \Pi_i.\pi[b_1] = \Pi_i.\pi[b_2]$ **then**
5: **if** ISADJACENT(b_1, b_2) **then**
6: $b \leftarrow$MERGEREGIONS(b_1, b_2)
7: ADD$(\Pi_i$.*beliefPartition*$, b, \Pi_i.\pi[b_1])$
8: Π_i.*count*$[b] \leftarrow$ *min*$(\Pi_i$.*count*$[b_1], \Pi_i$.*count*$[b_2])$
9: REMOVE(Π_i, b_1); REMOVE(Π_i, b_2)
10: **return** Π_i

PROPOSITION 1. *In CS-JESP, the joint expected reward for all starting belief points is monotonically increasing with each iteration.*

Proof Sketch. In every iteration, each starting belief point must belong to one of the regions in the belief partition of the free agent. Each such belief region corresponds to one of the value vectors, calculated by a call to OPTIMALBESTRESPONSE(Algorithm 6). Since DB-GIP is optimal, these vectors should either equal or dominate the vectors at the previous iteration, in all belief regions.■

PROPOSITION 2. *CS-JESP will terminate iff the joint policy has converged in all the free agent's belief regions.*

Proof Sketch. By construction, CS-JESP (Algorithm 3) terminates iff *converged* $=$ true, which will happen iff Π_i.*count*$[br] \geq n$, for all belief regions br of the free agent i. Π_i.*count*$[br] \geq n$ iff the joint policy for the region br remains constant for n iterations. In order for the joint *general policy* to remain constant for n iterations, OPTIMALBESTRESPONSE() should return identical *specialized policies* (to those in previous iteration) for all the belief regions , for $n - 1$ free agents. This happens when no one agent can improve the global value by altering its *general policy*, i.e. when local optima is attained. Furthermore, we round off each dimension of a belief region to *precision* decimal spaces, and hence the number of possible belief regions cannot grow indefinitely.■

From Propositions 1 and 2, we can conclude that CS-JESP will always terminate. At termination, the joint policy will be locally optimal as long as none of the belief regions returned by OPTIMALBESTRESPONSE were eliminated by the ROUNDOFF procedure.

4. EXPERIMENTAL RESULTS

This section provides three types of evaluations for CS-JESP using the multiagent tiger domain [11]. The first experiment focuses on run-time evaluations. We provide a comparison of three techniques: (i) CS-JESP+GIP: is the basic version of the combination of the JESP and the value iteration algorithm, GIP of single agent POMDPs. (ii) CS-JESP+DB, is JESP with DB-GIP. (iii)CS-JESP+DBM is CS-JESP+DB with the merging enhancement. Results of this experiment are shown in Figure 3. We experiment with two separate reward structures (presented in [11]). Figure 3(a) and Figure 3(b) focus on reward structure1, while Figure 3(c) and Figure 3(d) focus on reward structure2. In Figure 3(a), x-axis plots varying time horizon while y-axis plots run-time in milliseconds on log-scale[1]. In Figure 3(b), x-axis again plots time horizon, but the y-axis plots run-time in milliseconds (no log-scale is used). Time limit for the problems was set at 7,500,000 ms, after which they

[1]Machine specs for all experiments: Intel Xeon 3.6 GHZ processor, 2GB RAM

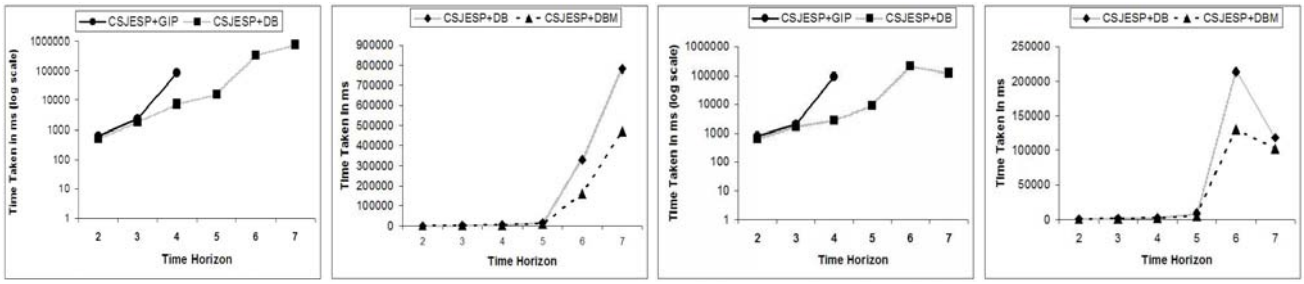


Figure 3: Comparison of (a) CSJESP+GIP, and CSJESP+DB for reward structure 1 (b) CSJESP+DB, and CSJESP+DBM for reward structure 1 (c) CSJESP+GIP, and CSJESP+DB for reward structure 2 (d) CSJESP+DB, and CSJESP+DBM for reward structure 2

were terminated. Figure 3(a) and Figure 3(c) refer to comparisons between CS-JESP+GIP and CS-JESP+DB, while Figure 3(b) and Figure 3(d) refer to comparisons between CS-JESP+DB and CS-JESP+DBM for the two reward structures.

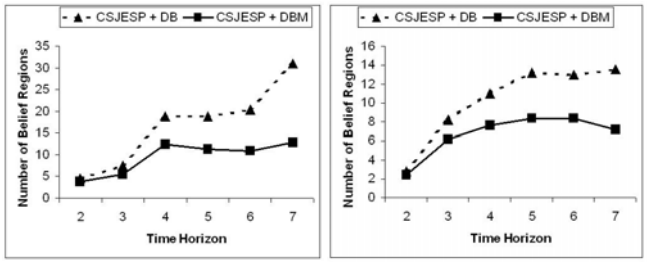


Figure 4: Comparison of the number of belief regions created in CS-JESP+DB and CS-JESP+DBM for reward structures 1 and 2

Figure 3(a) shows that CS-JESP+GIP did not terminate within the specified time limit after T=4. However, CS-JESP+DB converged to the solution even for T = 7, within the specified time limit. Even in cases where CS-JESP+GIP terminates, CS-JESP+DB provides significant speedups. For instance, in Figure 3(a), at T = 4, while CS-JESP+GIP takes in 83717.8 ms, CS-JESP+DB takes only 7345.2 ms leading to a speedup of 11.4 fold. Similar conclusions can be drawn from Figure 3(c). These results illustrate the synergy of JESP and DB-GIP, and the suitability of CS-JESP to take advantage of DB-GIP.

Figure 3(b) shows that CS-JESP+DBM provides further speedups over CS-JESP+DB, as time horizon increases. For instance at T=7 in Figure 3(b), merging in CS-JESP+DBM provided 1.66 fold speedup over CS-JESP+DB. Similar results are obtained with reward structure 2 in Figure 3(d), thus establishing the utility of merging contiguous regions with identical policies. In Figure 3(d) T=7 post merging show a faster execution compared to the T=6 results post merging. This occurs because the number of iterations of CS-JESP required for convergence at T=7 are lower (6) compared with iterations at T=6 (11).

Our second evaluation in Figure 4 focuses on understanding the speedups due to merging in CS-JESP+DBM. The number of belief regions present in the final solution is an indicator of the number of single agent POMDPs getting solved at each iteration. The x-axis in the figures represents the time horizon, while the y-axis is the number of belief regions. Thus in Figure 4, for a time-horizon of 7, using CSJESP+DB led to 31 belief regions, whereas using CSJESP+DBM led to 13 belief regions, a 2.39-fold reduction in the number of belief regions considered. Furthermore, we see that increasing the time horizon leads to increasing reduction in the number of belief regions with CS-JESP+DBM when compared to the number with CS-JESP+DB. Effect of the number of belief regions on the time taken increases with time horizon, because the single agent POMDPs expand in size with the time horizon. This provides the explanation for the timing results for CS-JESP+DBM in Figure 3(b) and Figure 3(d).

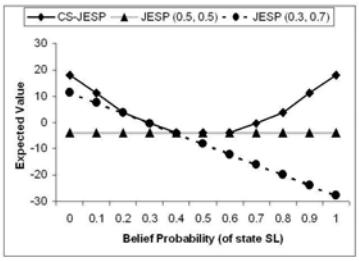


Figure 5: Comparison of the expected values obtained with JESP for specific belief points and CS-JESP

Our third evaluation focused on illustrating that CS-JESP achieves what it set out to do — generating policies over continuous initial belief space as shown in Figure 5. Belief space (in this domain belief probability of SL) is denoted on the x-axis, while the expected value of the policy is depicted on the y-axis. CS-JESP provides a general policy where the expected value is represented by a "CUP"-shape. There are five different policies represented in the cup, each dominant over a single belief region. The figure also indicates that if we were to approximate this entire general policy with a single policy over a single starting belief state, e.g. with JESP, then results may be arbitrarily worse. For instance with JESP (0.3, 0.7), the value at (1, 0) is -27, while the value generated with CS-JESP is 18, a difference of 45. With JESP (0.5, 0.5), the value at (1,0) is -4, where CS-JESP attains a value of 18, a difference of 22.

Of course, we may sample several belief points with JESP and then for a new belief point provide a policy from the nearest sample. Such a proposed heuristic approach naturally leads to our fourth evaluation comparing CS-JESP runtime to an approach that samples the belief space. This evaluation is not meant to be a precise comparison of JESP and CS-JESP, instead the aim is to show that the run-time results for sampled JESP would be comparable to the run times of CS-JESP. In Table 1, we show the run times of JESP and CS-JESP for T = 6, and T=7 for reward structure1. To replicate the policy obtained with CS-JESP, JESP would have to sample atleast as many times as the number of belief regions in the final policy of CS-JESP. For instance, for T=7, the number of samples required for JESP would be thirteen (from Figure 4). Table 1 shows an estimate of such a sampled JESP technique, given runtime re-

sults from [12]. We see that CS-JESP run-times are comparable, yet CS-JESP provides guarantees on these results that are unavailable with sampling.

	CS-JESP	JESP	Sampled Regions	Sampled JESP
T = 6	160336	15000	11	165000
T = 7	470398	73000	13	949000

Table 1: Comparison of run times (in ms) for JESP and CS-JESP

5. SUMMARY AND RELATED WORK

Distributed POMDPs are evolving as a popular approach in modeling multiagent systems, and many different algorithms have been proposed to obtain locally or globally optimal policies. Unfortunately, most of these have either been explicitly designed or experimentally evaluated assuming knowledge of a starting belief, an assumption that often does not hold in complex, uncertain domains. Instead, in such domains, it is important for agents to explicitly plan over continuous belief spaces. This paper provides a novel algorithm, CS-JESP, to explicitly compute finite horizon policies over continuous belief spaces, without restricting the space of policies. By marrying an efficient single-agent POMDP solver with a heuristic distributed POMDP policy-generation algorithm, locally optimal joint policies are obtained, each of which dominates within a different belief region. Key ideas in CS-JESP include: (a) Synergistic interaction between JESP and DB-GIP that enables significant performance improvement in CS-JESP; (b) Calculation of dominanant belief regions for *specialized policies*; (c) Region-based convergence; and (d) Merging of adjacent regions with identical *specialized policies*. To the best of our knowledge, these are the first run-time results for analytically generating policies over continuous belief spaces in distributed POMDPs.

In terms of related work, approaches for solving distributed POMDPs can be broadly classified in two main categories: globally optimal approaches and locally optimal algorithms. Becker *et al.* [2] present an exact globally optimal algorithm – the coverage set algorithm for transition-independent distributed MDPs. However unlike CS-JESP, this algorithm starts from a particular known initial state distribution. Hansen *et al.* [8] present an exact algorithm for partially observable stochastic games (POSGs) based on dynamic programming and iterated elimination of dominant policies. Like our work, they too deal with a continuous belief space and an unknown initial state distribution. This algorithm is important from a theoretical standpoint, but because of the inherent complexity of finding an exact solution for general distributed POMDPs, this algorithm does not scale well.

Among locally optimal approaches, Peshkin *et al.* [16] use gradient descent search to find local optimum finite-controllers with bounded memory. Their algorithm finds locally optimal policies from a limited subset of policies, with an infinite planning horizon. Their work does not consider a continuous belief space and starts from a fixed belief point. We have earlier discussed Nair *et al.* [11]'s JESP algorithm that uses dynamic programming to reach a local optimal. Bernstein *et al.* [7] present a locally optimal bounded policy iteration algorithm for infinite-horizon distributed algorithms. This algorithm has been theoretically shown to work in a continuous belief space from an unknown initial belief distribution. While this is an important contribution, the use of finite-state controllers restricts the policy representation. Also, their experimental results are for a single initial belief. Further, unlike our algorithm they use a *correlation device* in order to ensure coordination among the various agents.

Acknowledgements This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division under Contract No. NBCHD030010, DARPA/IPTO COORDINATORS program and the Air Force Research Laboratory under Contract No. FA875005C0030. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

6. REFERENCES

[1] M. L. Littman A. R. Cassandra and N. L. Zhang. Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. In *UAI*, 1997.

[2] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Transition-independent decentralized Markov decision processes. In *AAMAS*, 2003.

[3] D. S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of MDPs. In *UAI*, 2000.

[4] I. Chadès, B. Scherrer, and F. Charpillet. A heuristic approach for solving decentralized-pomdp: Assessment on the pursuit problem. In *SAC*, 2002.

[5] Z. Feng and S. Zilberstein. Region based incremental pruning for POMDPs. In *UAI*, 2004.

[6] Claudia V. Goldman and Shlomo Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *AAMAS*, 2003.

[7] D. Bernstein; E. Hansen; and S. Zilberstein. Bounded policy iteration for decentralized pomdps. In *IJCAI*, 2005.

[8] Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, 2004.

[9] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *AIJ*, 101(2), 1998.

[10] R.E. Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *AAMAS*, 2004.

[11] R. Nair, D. Pynadath, M. Yokoo, M. Tambe, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*, 2003.

[12] R. Nair, M. Roth, M. Yokoo, and M. Tambe. Communication for improving policy computation in distributed pomdps. In *AAMAS*, 2004.

[13] R. Nair, M. Tambe, and S. Marsella. Role allocation and reallocation in multiagent teams: Towards a practical analysis. In *AAMAS*, 2003.

[14] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *AAAI*, 2005.

[15] M. Tambe P. Varakantham, R. Maheswaran. Exploiting belief bounds: Practical pomdps for personal assistant agents. In *AAMAS*, 2005.

[16] L. Peshkin, N. Meuleau, K.-E. Kim, and L. Kaelbling. Learning to cooperate via policy search. In *UAI*, 2000.

[17] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *JAIR*, 16:389–423, 2002.

[18] P. Xuan, V. Lesser, and S. Zilberstein. Communication decisions in multiagent cooperation. In *Agents*, 2001.

Resource Allocation Among Agents with Preferences Induced by Factored MDPs

Dmitri Dolgov and Edmund Durfee
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109
{ddolgov, durfee}@umich.edu

ABSTRACT

Distributing scarce resources among agents in a way that maximizes the social welfare of the group is a computationally hard problem when the value of a resource bundle is not linearly decomposable. Furthermore, the problem of determining the value of a resource bundle can be a significant computational challenge in itself, such as for an agent operating in a stochastic environment, where the value of a resource bundle is the expected payoff of the optimal policy realizable given these resources. Recent work has shown that the structure in agents' preferences induced by stochastic policy-optimization problems (modeled as MDPs) can be exploited to solve the resource-allocation and the policy-optimization problems simultaneously, leading to drastic (often exponential) improvements in computational efficiency. However, previous work used a flat MDP model that scales very poorly. In this work, we present and empirically evaluate a resource-allocation mechanism that achieves much better scaling by using factored MDP models, thus exploiting both the structure in agents' MDP-induced preferences, as well as the structure within agents' MDPs.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Performance, Design

Keywords

Task and resource allocation in agent systems, (Multi-)agent planning.

1. INTRODUCTION

Resource allocation is recognized as an important problem in many research fields, such as computer science, economics, and operations research. The core question of resource allocation is how to distribute a set of scarce resources among a set of agents (either cooperative or self-interested) in the best possible way, with social welfare (sum of agents' utilities) as one of the most popular optimization criteria. This problem is computationally challenging (NP-complete [19]) when agents' utility functions are defined over *bundles* of resources and are not linearly decomposable. Such computational challenges in combinatorial resource-allocation problems have recently precipitated the move towards resource-allocation mechanisms that model the mechanisms underlying agents' resource preferences (e.g., [15]), as opposed to the more-traditional black-box-utility models.

In particular, we have previously proposed a model where an agent's resource preferences are defined by an underlying Markov decision process, where resources are required to enable actions, and the value of a resource bundle is the expected payoff of the optimal policy that is realizable given these resources [8, 9]. In other words, an agent's MDP is parameterized by the available resources, and the agent's utility for resources is defined as the expected value of the best realizable policy. The model assumes that agents' planning problems are *weakly-coupled* [16], i.e., once the resources are distributed, the agents are independent.

Under this model, a naive resource-allocation approach would require each agent to formulate and solve an exponential number of MDPs (one for each resource bundle) just to define its utility function, and then to solve an NP-hard allocation problem on the exponentially-large input. However, a superior method can be implemented that combines the resource-allocation and the policy-optimization problems and formulates them as a single constrained MDP that can then be reduced to a mixed integer program, leading to an exponential reduction in the number of binary decision variables. This approach exploits the structure in agents' preferences that stems from the regularities of the underlying Markov process, which leads to drastic improvements in scalability of the resource-allocation problem.

There is, however, another source of intractability in the model described above that is due to the MDP models themselves. The traditional *flat* way of representing an MDP requires an explicit enumeration of all possible states of the system, which subjects the model to the curse of dimensionality [2] and leads to an exponential blow-up of the state space in terms of the number of *state features* of the problem. This representational challenge is addressed by *factored* MDPs [3] that model the state space as a cross product of state features, represent the transition function as a dy-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

namic Bayesian network, and assume the reward function can be additively decomposed into functions with compact domains. Unfortunately, a solution to a well-structured factored MDP does not, in general, maintain the structure and compactness of the problem [14, 7], which forces the need for approximate solutions. One popular family of such methods that is particularly well-suited for our goal is approximate linear programming (ALP) [20, 6], for which there exist several very efficient algorithms [17, 12, 5, 10].

The main contribution of this work is that it extends the line of research on resource-allocation methods for agents with MDP-induced preferences [8, 9] to factored MDP models. It builds on the ALP methodologies of [12] to construct an efficient algorithm that combines resource allocation and approximate policy optimization for factored MDPs. This effectively exploits both the structure in agents' preferences that is *due to the underlying MDPs*, as well as the structure *within the MDPs* themselves, leading to exponential improvements in efficiency both in terms of the size of the resource-allocation problem (as the number of resource types increases) and the size of the policy-optimization problem (as the number of state features in an MDP grows).

2. FACTORED MDP AND ALP

A discrete-time, infinite-horizon, discounted MDP (e.g., [18]) can be described as $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{S} = \{s\}$ is the finite set of system states, $\mathcal{A} = \{a\}$ is a finite set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the transition function ($P(\sigma|s, a)$ is the probability of moving into state σ upon executing action a in state s), $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the (bounded) reward function ($R(s, a)$ is the reward for executing action a in state s), and $\gamma \in [0, 1)$ is the discount factor (a unit reward at time τ is aggregated into the total reward as γ^τ).

A solution to such an MDP is a stationary, deterministic policy π. We define a policy as a mapping of states to probability distributions over actions: $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, where $\pi(s, a)$ defines the probability that action a is executed in state s (for deterministic policies, only one $\pi(s, a)$ per state is nonzero). One way of solving an MDP is to compute the optimal value function v, which defines, for every state, the total expected discounted reward of the optimal policy. It is then optimal to simply act greedily with respect to v.

The optimal value function can be obtained, for example, using the following minimization linear program (LP), which is often called the *primal LP* of an MDP (e.g., [18]):

$$\begin{aligned} &\min \sum_s \alpha(s) v(s) \qquad \text{s.t.:} \\ &v(s) \geq R(s, a) + \gamma \sum_\sigma P(\sigma|s, a) v(\sigma), \quad \forall s \in \mathcal{S}, a \in \mathcal{A}, \end{aligned} \tag{1}$$

where α is an arbitrary strictly positive distribution over the state space ($\alpha(s) > 0$). This LP has $|\mathcal{S}|$ variables and $|\mathcal{S}||\mathcal{A}|$ constraints. Equivalently, the problem can be formulated as the *dual LP* with $|\mathcal{S}||\mathcal{A}|$ variables and $|\mathcal{S}|$ constraints:

$$\begin{aligned} &\max \sum_{s,a} R(s, a) x(s, a) \qquad \text{s.t.:} \\ &\sum_a x(\sigma, a) - \gamma \sum_{s,a} x(s, a) P(\sigma|s, a) = \alpha(\sigma), \quad \forall s \in \mathcal{S}, \end{aligned} \tag{2}$$

where $x(s, a) \geq 0$ is called the *occupation measure* ($x(s, a)$ is the discounted number of executions of a in s), and the constraints ensure the conservation of flow through each state.

Given a solution to (2), an optimal policy can be computed as: $\pi(s, a) = x(s, a) / \sum_a x(s, a)$. A policy thus obtained is *uniformly-optimal* (optimal for all initial states) when $\alpha(s)$ is strictly positive. The existence of uniformly-optimal policies does not hold for the resource-parameterized MDP models that we use in this work (detailed in Section 3), so for such problems the initial conditions α will be an important part of problem specification.

2.1 Factored MDPs

The standard MDP model defined above requires an enumeration of all system states and thus scales very poorly. This problem is addressed by a compact MDP representation [3, 4] that defines the state space as the cross-product of state features: $\mathcal{S} = z_1 \times z_2 \ldots z_N$, and uses a factored transition function and an additively-separable reward function.

The transition function is specified as a two-layer dynamic Bayesian network (DBN), with the current state features viewed as the parents of the next time-step features:

$$P(\sigma|s, a) = P(\mathbf{z}(\sigma)|\mathbf{z}(s), a) = \prod_{j=1}^{J_P} p_j(\mathbf{z}_j(\sigma)|a, \mathbf{z}_{p_j}(s)), \tag{3}$$

where $\mathbf{z}(\cdot)$ is the instantiation of all $\mathcal{Z}$ features corresponding to a state, $z_j(\cdot)$ denotes the value of the jth state feature of a state, and $\mathbf{z}_{p_j}(\cdot)$ is the instantiation of the DBN-parents $\mathcal{Z}_{p_j}$ of z_j. Likewise, in the rest of this paper, we will use $\mathcal{Z}_\varphi$ to refer to the set of features in the domain of function φ, and $\mathbf{z}_\varphi$ to refer to an instantiation of these features.

The reward function is compactly defined as

$$R(s, a) = \sum_{j=1}^{J_R} r_j(\mathbf{z}_{r_j}(s), a), \tag{4}$$

where $\mathbf{z}_{r_j}(\cdot)$ is an instantiation of the features $\mathcal{Z}_{r_j} \subseteq \mathcal{Z}$ that are in the domain of the jth local reward function r_j.

Clearly, this factored representation is only beneficial if the local transition functions $p_j(z_j(\sigma)|\mathbf{z}_{p_j}, a)$ and local reward functions $r_j(\mathbf{z}_{r_j}, a)$ have small domains: $|\mathcal{Z}_{p_j}| \ll |\mathcal{Z}|$ and $|\mathcal{Z}_{r_j}| \ll |\mathcal{Z}|$, i.e., each reward and transition component depends on a small subset of all state features $\mathcal{Z}$.

2.2 Primal ALP

Approximate linear programming [20, 6] lowers the dimensionality of the primal LP (1) by restricting the optimization to the space of value functions that are linear combinations of a predefined set of K basis functions h:

$$v(s) = v(\mathbf{z}(s)) = \sum_{k=1}^{K} h_k(\mathbf{z}_{h_k}(s)) w_k, \tag{5}$$

where $h_k(\mathbf{z}_{h_k})$ is the kth basis function defined on a small subset of the state features $\mathcal{Z}_{h_k} \subset \mathcal{Z}$, and w are the new optimization variables. This technique is similar to linear regression, where a function is approximated as a linear combination of a given (in general, non-linear) basis. The difference, however, is that here instead of minimizing a measure of error for the given data points, the goal is to minimize the measure to the unknown optimal value function. For the approximation to be computationally effective, the domain of each basis function has to be small ($|\mathcal{Z}_{h_k}| \ll |\mathcal{Z}|$).

As a notational convenience, we can rewrite the above as $v = Hw$, where H is a $|\mathcal{S}| \times |w|$ matrix composed of basis

functions h_k.[1] Thus, the LP (1) becomes:

$$\min \alpha^T Hw \big| AHw \geq r, \tag{6}$$

where the constraint matrix is $A_{sa,\sigma} = \delta(s,\sigma) - \gamma P(\sigma|s,a)$ (where $\delta_{s\sigma}$ is the Kronecker delta, $\delta_{s\sigma} = 1 \Leftrightarrow s = \sigma$).

For this method to be effective, we need to be able to efficiently compute the objective function $\alpha^T H$ and the constraints AH, which can be done as described in [12]. Consider a factored initial distribution:

$$\alpha(s) = \prod_{j=1}^{J_\alpha} \mu_j(\mathbf{z}_{\mu_j}(s)),$$

where, as usual, the domain of each factor μ_j is taken to be small ($|\mathcal{Z}_{\mu_j}| \ll |\mathcal{Z}|$). The objective function is computed as:

$$\begin{aligned}(\alpha^T H)_k &= \sum_s \alpha(s) H_{sk} = \sum_s \prod_i \mu_j(\mathbf{z}_{\mu_j}(s)) h_k(\mathbf{z}_{h_k}(s)) \\ &= \sum_{\mathbf{z}'} \prod_{j'} \mu_{j'}(\mathbf{z}'_{\mu_{j'}}) h_k(\mathbf{z}'_{h_k}),\end{aligned}$$

where $\mathbf{z}'$ iterates over all features in the union of the domain of h_k and the domains of those $\mu_{j'}$ that have a non-zero intersection with the domain of h_k: $\mathbf{z}' = \{\mathbf{z}_{\mu_j} \cup \mathbf{z}_{h_k} : \mathcal{Z}_{\mu_j} \cap \mathcal{Z}_{h_k} \neq \varnothing\}$, because all μ_i that do not have any variables in common with h_k factor out and their sum is 1 (since it is a sum of a probability distribution over its domain).[2]

The constraints in (6) are also computed efficiently:

$$\begin{aligned}(AH)_{sa,k} &= \sum_\sigma A_{sa,\sigma} h_k(\sigma) = \sum_\sigma (\delta_{s\sigma} - \gamma P(\sigma|s,a)) h_k(\sigma) \\ &= \sum_{\mathbf{z}} \delta(\mathbf{z}(s), \mathbf{z}) h_k(\mathbf{z}) - \gamma \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{z}(s), a) h_k(\mathbf{z}).\end{aligned} \tag{7}$$

The first sum can be computed efficiently, because it is simply $h_k(\mathbf{z}_{h_k}(s))$, since δ is nonzero only at $\mathbf{z}(\sigma) = \mathbf{z}(s)$. The second term can also be computed efficiently, since $P(\mathbf{z}(\sigma)|\mathbf{z}(s), a)$ is a factored probability distribution (just like in the case of α above).

Notice that in the above, the second term in (7) can be precomputed efficiently by performing a backprojection of every basis function h_k:

$$g_k(\mathbf{z}_{g_k}, a) = \sum_{\mathbf{z}} P(\mathbf{z}'|\mathbf{z}(s), a) h_k(\mathbf{z}) \quad \forall k \in [1, K], \tag{8}$$

where the domain of the backprojection g_k is the union of the DBN-parents of all features that are in the domain of the kth basis function: $\mathcal{Z}_{g_k} = \bigcup_{j: z_j \in \mathcal{Z}_{h_k}} \mathcal{Z}_{p_j}$.

The primal ALP described above reduces the number of optimization variables from $|\mathcal{S}|$ to $|w| = K$, and the coefficients of the objective function and every constraint row can be computed efficiently. However, the number of rows in the constraint matrix remains exponential at $|\mathcal{S}||\mathcal{A}|$, so the ALP has to undergo some additional transformation (or approximation) to become feasible. To address this issue, several techniques have been proposed, such as exploiting problem structure [12], sampling [5], and linear approximations [10]. The benefit of the former, compared to the latter two, is that it does not introduce a new source of approximation error, while the downside is that the number of constraints needed to represent the feasible region exactly is exponential in the induced tree width of the associated cluster graph (a graph with a vertex per variable and edges between variables that appear together in the domain of any function).

[1]Note that the exponentially-sized H is never explicitly written out, since each column is a basis function that can be represented compactly.

[2]See [11] or [10] for examples of how the LP coefficients can be computed efficiently.

2.3 Dual ALP

When dealing with constrained MDPs (flat or factored), such as the ones that arise in the resource allocation problem discussed in Section 4, it is more practical to work with the duals of the (A)LPs. Guestrin [11] considers the dual of (6):

$$\max r^T x \Big| H^T A^T x = \alpha. \tag{9}$$

This LP has $|\mathcal{S}||\mathcal{A}|$ variables and $|w| = K$ constraints (approximated flow conservation), where each of the K constraints corresponds to a linear combination of $|\mathcal{Z}_{h_k}|$ real flow constraints, with h_k defining the weights with which they are aggregated into the new approximate constraint. Clearly, (9) is always feasible. Furthermore, as shown in [11], a solution to (9) constitutes a valid density function, summing to $(1-\gamma)^{-1}$, which implies that the LP is bounded.

A problem with the LP (9) is that it has an exponential number of optimization variables. However, as proposed by Guestrin [11], the exact occupation measure x can be represented more compactly by using *marginal occupation measures* (or marginal visitation frequencies) ξ, which define the occupation measure over subsets of the state features. Let us define a marginal occupation measure over each of the following clusters of variables: the domain of every primal basis function $\mathcal{Z}_{h_k}$, the domain of every local reward function $\mathcal{Z}_{r_j}$, and the domains of all backprojections of the basis functions $\mathcal{Z}_{g_k}$. The challenge then becomes in ensuring that the marginal occupation measures ξ are consistent and correspond to a valid global distribution over $\mathcal{Z}$. Fortunately, global consistency (but not necessarily correspondence to a valid flat occupation measure x) can often be achieved via local marginal consistency constraints [11] that ensure that marginal occupation measures ξ_1 and ξ_2 agree on the values of features they share (with non-overlapping features marginalized out):

$$\sum_a \sum_{\widetilde{\mathbf{z}_j}, a} \xi_j(\mathbf{z}_j, a) = \sum_a \sum_{\widetilde{\mathbf{z}_i}, a} \xi_i(\mathbf{z}_i, a), \qquad \forall \mathbf{z}_{ji}, \tag{10}$$

where $\mathcal{Z}_{ji} = \mathcal{Z}_i \bigcup \mathcal{Z}_j$ are the shared features, and $\widetilde{\mathcal{Z}_i} = \mathcal{Z}_j \setminus \mathcal{Z}_i$ and $\widetilde{\mathcal{Z}_j} = \mathcal{Z}_i \setminus \mathcal{Z}_j$ are unique. As is the case in graphical models [13] more generally, and as observed for this particular case in [11], when the cluster graph associated with the domains of the marginal occupation measures ξ forms a *junction tree*[3] local consistency implies global consistency. If the original cluster graph does not form a junction tree, it can be converted to one by applying the widely-used in inference techniques of *moralization* and *triangulation*.[4]

Using the marginal occupation measures, the LP (9) can be represented more compactly as the following LP defined

[3]A junction tree is an undirected tree with clusters of state features as nodes, with the property that if a feature appears in clusters A and B, it also appears in any cluster in the path from A to B.

[4]Note that while the problem of computing the junction tree with the smallest width is an NP-hard problem [1], there are efficient approximations.

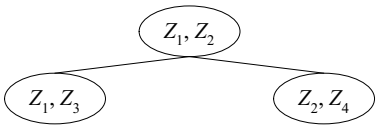


Figure 1: A cluster graph that forms a junction tree.

on marginal occupation measures $\xi_j(\mathbf{z}_j)$, one per each cluster in $\mathcal{Z}_{h_k}$, $\mathcal{Z}_{g_k}$, $\mathcal{Z}_{r_j}$.[5]

$$\begin{aligned}
&\max \sum_{m=1}^{M} \sum_{a} \sum_{\mathbf{z}_{r_j}} r_j(\mathbf{z}_{r_j}, a)\xi_{r_j}(\mathbf{z}_{r_j}, a) \qquad \text{s.t.:} \\
&\sum_{a} \xi_{h_k}(\mathbf{z}_{h_k}, a)h_k(\mathbf{z}_{h_k}) - \gamma \sum_{a} g_k(a)\xi_{g_k}(\mathbf{z}_{h_k}, a) = \\
&\qquad\qquad \sum_{\mathbf{z}_{h_k}} \alpha(\mathbf{z}_{h_k})h_k(\mathbf{z}_{h_k}), \quad \forall k \in [1, K]; \\
&\sum_{\widetilde{\mathbf{z}_j}} \xi_j(\mathbf{z}_j, a) = \sum_{\widetilde{\mathbf{z}_i}} \xi_i(\mathbf{z}_i, a), \\
&\quad \forall a, \xi_j, \xi_i, \mathbf{z}_j, \mathbf{z}_i, \ \widetilde{\mathcal{Z}_i} = \mathcal{Z}_j \setminus \mathcal{Z}_i, \ \widetilde{\mathcal{Z}_j} = \mathcal{Z}_i \setminus \mathcal{Z}_j; \\
&\sum_{a} \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j, a) = (1-\gamma)^{-1}, \qquad \forall j, \ \mathbf{z}_j; \\
&\xi_j(\mathbf{z}_j, a) \geq 0, \qquad \forall a, j, \mathbf{z}_j.
\end{aligned} \tag{11}$$

Given a solution $\xi_j(\mathbf{z}_j)$ to the above LP, the optimal policy for any given state can be computed in the following way, detailed in [12]. Given the dual solution $\xi_j(\mathbf{z}_j)$, compute the primal solution w that defines the value-function approximation. Given the current state, do a one-step lookahead and, for each possible next state, compute the value-function estimate based on w, and greedily select the action that maximizes the sum of the one-step reward and the expected value of the next step (as in the standard MDP).

Another way of obtaining the policy is to calculate it directly from the marginal occupation measures ξ. Each marginal occupation measure forms a valid distribution on its domain, so we can use a variant of the junction-tree inference algorithm for calculating the distribution over actions, given the current state. The procedure is described in Algorithm 1 and works by simply traversing the cluster tree from the root and factoring in the marginal occupation measures along the way, while normalizing appropriately. Note that because the marginal occupation measures (even if globally consistent) do not necessarily correspond to a valid occupation measure, there is not a one-to-one correspondence between the marginals ξ and policy π, as between x and π in the flat case.

EXAMPLE 1. *Consider the cluster graph in Figure 1. The marginal occupation measures are: $\xi(z_1, z_2, a)$, $\xi(z_1, z_3, a)$, and $\xi(z_2, z_4, a)$. This graph forms a junction tree, so global consistency of marginal occupation measures is guaranteed*

[5]Here and below, for simplicity we assume the cluster graph is a junction tree. If not, the LP will have to undergo a preprocessing step, as described above.

Algorithm 1: Policy from occupation measures.

Function PolicyForState
in : $\mathbf{z}$ – current state
out : $\pi(\mathbf{z}, a)$ – policy for current state

$\pi(\mathbf{z}, a) \leftarrow$ PolicyForStateRec($\mathbf{z}$, root-cluster, $\mathbf{1}$, $\varnothing$)
if $\sum_a \pi(\mathbf{z}, a) = 0$ **then**
  /* `use any action supported by marginals` */
  $\pi(\mathbf{z}, a) \leftarrow 1$ for any a, s.t., $\exists j, \mathbf{z}'_j : \ \xi_j(\mathbf{z}'_j, a) > 0$
end

Function PolicyForStateRec
in : $\mathbf{z}$ – current state
: j – cluster id
: $\pi(\mathbf{z}, a)$ – marginal policy
: Ω – features already factored into policy
out : $\pi'(\mathbf{z}, a)$ – marginal policy
: Ω' – features already factored into policy

/* `update features already factored in` */
$\Omega' \leftarrow \Omega \bigcup \mathcal{Z}_{\xi_j}$
/* `factor in` j^{th} `marginal` */
$\pi'(\mathbf{z}, a) \leftarrow \pi(\mathbf{z}, a)\xi(\mathbf{z}_j, a)$
/* `normalize by overlapping features` */
$\mathcal{Z}' \leftarrow \Omega \bigcap \mathcal{Z}_{\xi_j}$
$\pi'(\mathbf{z}, a) \leftarrow \pi'(\mathbf{z}, a)/\xi(z')$
/* `factor in children` */
forall $m \in Children(j)$ **do**
  $\pi'' \leftarrow PolicyForStateRec(j, \pi', \Omega)$
  $\pi'(\mathbf{z}, a) \leftarrow \pi'(\mathbf{z}, a)\pi''(\mathbf{z}, a)$
end

whenever local consistency constraints are satisfied:

$$\sum_{z_2} \xi(z_1, z_2, a) = \sum_{z_3} \xi(z_1, z_3, a), \qquad \forall z_1, a;$$
$$\sum_{z_1} \xi(z_1, z_2, a) = \sum_{z_4} \xi(z_2, z_4, a), \qquad \forall z_2, a.$$

The global distribution for this problem is:

$$x(z_1, z_2, z_3, z_4, a) = \frac{\xi(z_1, z_2, a)\xi(z_1, z_3, a)\xi(z_2, z_4, a)}{\sum_{z_1} \xi(z_1, z_2, a) \sum_{z_2} \xi(z_1, z_2, a)},$$

which is what would be computed by Algorithm 1. □

3. SINGLE-AGENT MODEL

We now briefly describe the model of the capacity-limited agent with an MDP whose action set is parameterized by the resources available to the agent. This is a slight extension of the model described in [8] to factored problems and arbitrary resource requirements.

The justification behind the model described in this section is that often an agent has many capabilities that are all in principle available to it, but not all combinations of these capabilities are realizable within the agent's architectural limitations, because choosing to enable some of the capabilities might seize the resources needed to enable others. In other words, a particular policy might not be feasible

because the agent's architecture does not support the combination of capabilities required for that policy.

We model this constrained policy-optimization problem as follows. The agent has a set of actions that are potentially executable, and each action requires a certain combination of resources. All resources have capacity costs, and each agent has capacity constraints that limit what resources it can make use of. For example, a delivery company needs vehicles and loading equipment (resources) to make its deliveries (execute actions). However, all equipment costs money and requires manpower to operate it (capacity costs). Therefore, the amount of equipment the agent can acquire and successfully utilize is constrained by factors such as its budget and limited manpower (capacity bounds).

Given the above, the goal is to assign to the agent a subset of the available resources that does not violate its capacity constraints, such that the optimal feasible policy under that resource assignment yields the highest expected utility. We can model such an optimization problem as $\langle \mathcal{S}, \mathcal{A}, P, R, \alpha, \mathcal{O}, \rho, \mathcal{C}, \kappa, \widehat{\kappa}, \rangle$, where:

- $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma, \alpha \rangle$ are the components of a factored MDP, represented in a factored manner, as defined in Section 2.1.
- $\mathcal{O} = \{o\}$ is the set of resource types (e.g., $\mathcal{O} = \{$production equipment, vehicle , $\ldots\}$).
- $\rho : \mathcal{A} \times \mathcal{O} \mapsto \mathbb{R}$ is a function that specifies the resource requirements of all actions; $\rho(a, o)$ defines how much of resource o is needed for action a (e.g., $\rho(a, \text{vehicle}) = 1$ means that action a requires one vehicle to be executable).
- $\mathcal{C} = \{c\}$ is the set of capacities of our agent (e.g., $\mathcal{C} =$ space, money, manpower, $\ldots\}$).
- $\kappa : \mathcal{O} \times \mathcal{C} \mapsto \mathbb{R}$ is a function that specifies the capacity costs of resources; $\kappa(o, c)$ defines how much of capacity c a unit of resource o consumes (e.g., $\kappa(\text{vehicle}, \text{money}) = \$50000$ defines the cost of a vehicle).
- $\widehat{\kappa} : \mathcal{C} \mapsto \mathbb{R}$ specifies the upper bounds on capacities; $\widehat{\kappa}(c)$ gives the upper bound on capacity c (e.g., $\widehat{\kappa}(\text{money}) =$ \$1,000,000 defines the agent's budget constraints).

Given the above, the agent's optimization problem is to find the set of resources whose capacity costs satisfy the constraints imposed by $\widehat{\kappa}$, such that the best realizable policy, given these resources (as induced by action resource requirements ρ) yields the highest expected discounted reward. As we have previously shown in [8] (for flat MDPs and binary resource-requirements), this problem is NP-complete.[6] This complexity result still holds for factored MDPs (the models are equivalent in expressiveness and worst-case space requirements), as well as for non-binary resource requirements (binary is a special case). Furthermore, in [8] we presented a reduction of the problem to a mixed integer linear program (MILP). We now develop an analogous reduction for the factored model of the single-agent capacity-constrained MDPs. This factored MILP will serve as the basis for the resource-allocation algorithm in the multiagent setting, discussed in Section 4.

Given a flat occupation measure x, the agent's capacity constraints can be formulated as:

$$\sum_o \kappa(o, c) \max_a \Big\{ \rho(a, o) H\big(\sum_s x(s, a) \big) \Big\} \leq \widehat{\kappa}(c), \qquad \forall c \in \mathcal{C}, \tag{12}$$

[6]It is essentially a multidimensional KNAPSACK problem with item values defined by the MDP.

where H is the Heaviside "step" function of a nonnegative argument, defined as

$$H(z) = \begin{cases} 0 & z = 0; \\ 1 & z > 0. \end{cases}$$

To describe (12) in turns of marginal occupation measures, we use the following Lemma.

LEMMA 1. *Consider a set of marginal occupation measures $\{\xi_j\}$ that are globally consistent and a corresponding policy $\pi(\mathbf{z}, a)$ computer via Algorithm 1. Then*

$$\forall \mathbf{z}, a: \quad \pi(\mathbf{z}, a) > 0 \iff \exists j: \quad \xi_j(\mathbf{z}_j, a) > 0, \tag{13}$$

i.e., the policy prescribes non-zero probability of executing action a in state $\mathbf{z}$ if and only if there is a marginal that prescribes a non-zero probability to that action in same state.

PROOF. The proof is essentially be definition. The " $\Rightarrow$ " direction follows from how policy π is constructed in Algorithm 1: in order for the global distribution to assign a non-zero probability to some event, there must exist a marginal that assigns a non-zero probability to a subset of features corresponding to the same event. The " $\Leftarrow$ " direction follows from global consistency of the marginals. Indeed, assume the opposite: $\xi_j(\mathbf{z}_j, a) > 0$ and $\pi(\mathbf{z}, a) = 0$. This would violate the following consistency constraint:

$$(1 - \gamma)^{-1} \xi_j(\mathbf{z}_j, a) = \sum_{\widetilde{\mathbf{z}_j}} \pi(\mathbf{z}_j, a),$$

where $\widetilde{\mathcal{Z}_j} = \mathcal{Z} \backslash \mathcal{Z}_j$ are the features not in the scope of ξ_j. □

It immediately follows from Lemma 1 that the flat constraints (12) can be equivalently represented on the marginal occupation measures ξ:

$$\sum_o \kappa(o, c) \max_a \Big\{ \rho(a, o) H\big(\sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j, a) \big) \Big\} \leq \widehat{\kappa}(c), \forall c \in \mathcal{C}. \tag{14}$$

The difficulty is that these constraints are non-linear due to the maximization over a and the Heaviside function H. To get rid of the first source of nonlinearity, let us observe that a constraint with a single max over a finite set of discrete values can be trivially linearized by expanding out the set over which the maximization is taken:

$$\max_{z \in \mathcal{Z}} f(z) \leq a \iff f(z) \leq a, \quad \forall z \in \mathcal{Z}.$$

In (14), we have a finite sum of max's, but it can also be linearized by observing that the inequality

$$\sum_i^n g(u_i) \max_{z \in \mathcal{Z}} f(z, u_i) = g(u_1) \max_{z \in \mathcal{Z}} f(z, u_1) + \ldots + g(u_n) \max_{z \in \mathcal{Z}} f(z, u_n) \leq a$$

is equivalent to the system of $|\mathcal{Z}|^n$ linear inequalities:

$$g(u_1) f(z_1, u_1) + g(u_2) f(z_2, u_2) + \ldots + g(u_2) f(z_2, u_2) \leq a, \quad \forall z_1, z_2, z_n \in \mathcal{Z}.$$

Applying this to the constraints in (14), we can express the original system of $|\mathcal{C}|$ nonlinear constraints:

$$\sum_o \kappa(o, c) \max_a \Big\{ \rho(a, o) H\big(\sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j, a) \big) \Big\} \leq \widehat{\kappa}(c), \forall c \in \mathcal{C},$$

as the following system of $|\mathcal{C}||\mathcal{A}|^{|\mathcal{O}|}$ linear constraints:

$$\sum_o \kappa(o,c)\rho(a_o,o)H\big(\sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j,a_o)\big) \le \widehat{\kappa}(c), \quad \forall c \in \mathcal{C}, a_{o_1}, a_{o_2}, \ldots \in \mathcal{A}. \tag{15}$$

Notice that this straightforward way of eliminating the maximization exponentially increases the number of constraints, because the above expansion enumerates all possible actions for each resource: it enumerates policies where each resource o is used by action a_1, where it is used by action a_2, action a_3, etc. However, in many problems not all resources are used by all actions. In such cases, most of the above constraints become redundant, and the number of constraints can be reduced from $|\mathcal{C}||\mathcal{A}|^{|\mathcal{O}|}$ to $|\mathcal{C}|\prod_o |\mathcal{A}_o|$, where $\mathcal{A}_o$ is the number of actions that use resource o. Furthermore, in a special case, where the resource requirements of actions are binary, the number of constraints can be reduced much more significantly, as discussed in Section 3.1.

To linearize the Heaviside function, we augment the original optimization variables x with a set of $|\mathcal{A}|$ binary variables $\Delta(a) \in \{0,1\}$, where

$$\Delta(a) = H\Big(\sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j,a_o)\Big). \tag{16}$$

In other words, $\Delta(a)$ is an indicator variable that shows whether action a is used in the policy with non-zero probability. Using Δ and expanding the max as above, we can rewrite the resource constraints in (14) as:

$$\sum_o \kappa(o,c)\rho(a_o,o)\Delta(a_o) \le \widehat{\kappa}(c), \quad \forall c \in \mathcal{C}, a_{o_1}, a_{o_2}, \ldots \in \mathcal{A}, \tag{17}$$

which are linear in Δ. We can then synchronize Δ and each of the marginal occupation measures ξ_j via the following linear inequalities:

$$\sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j,a)(1-\gamma) \le \Delta(a), \qquad \forall a \in \mathcal{A}. \tag{18}$$

Indeed, $\xi_j(\mathbf{z}_j,a)(1-\gamma) \le 1$, because $\sum_{\mathbf{z}_j,a} \xi_j(\mathbf{z}_j,a) = (1-\gamma)^{-1}$, which ensures that when any marginal $\xi(\cdot,a)$ assigns a non-zero probability to action a, this drives the corresponding $\Delta(a)$ to be 1. On the other hand, when all $\xi(\cdot,a)$ are 0 for a given a, both $\Delta(a)=0$ and $\Delta(a)=1$ satisfy (18), in which case $\Delta(a)$ will be driven down to $\Delta(a)=0$ by the capacity constraints (if they are binding; if they are not, the value of $\Delta(a)$ is irrelevant).

Putting it all together, the single-agent factored MDP with resources and capacity constraints can be formulated as an MILP by augmenting the LP (11) with binary variables $\Delta(a)$ and constraints (17) and (18).

3.1 Binary Resource Requirements

As a special case of the problem discussed in the previous section, consider a domain where the resource requirements are binary: $\rho(a,o) = \{0,1\}$. Under these conditions the single-agent MILP (11,17,18) can be simplified significantly.

First, observe that, when $\rho(a,o) = \{0,1\}$, the total resource requirements of a policy can be simplified as follows:

$$\max_a \Big\{\rho(a,o)H\big(\sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j,a)\big)\Big\} = H\Big(\sum_a \rho(a,o) \sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j,a)\Big).$$

Therefore, instead of introducing $|\mathcal{A}|$ binary variables $\Delta(a)$ that specify whether action a is used in the policy, it suffices to use $|\mathcal{O}|$ variables $\delta(o) \in \{0,1\}$:

$$\delta(o) = H\Big(\sum_a \rho(a,o) \sum_j \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j,a) \le \widehat{\kappa}(c)\Big), \tag{19}$$

where $\delta(o)$ indicates whether the policy requires resource o.

We can then synchronize the marginals ξ and the binary variables $\delta(o)$ similarly to $\Delta(a)$ above, leading to the following equivalent of (17,18) for binary resources:

$$\begin{aligned} &\sum_o \kappa(o,c)\delta(o) \le \widehat{\kappa}(c), && \forall c \in \mathcal{C}; \\ &\sum_a \rho(a,o) \sum_{j=1} \sum_{\mathbf{z}_j} \xi_j(\mathbf{z}_j,a) \le \delta(o), && \forall o \in \mathcal{O}. \end{aligned} \tag{20}$$

4. RESOURCE ALLOCATION

We now consider the multiagent problem of resource allocation between several agents, where the resource preferences of the agents are defined by factored MDPs with capacity-constraints, as described in the previous section. As mentioned previously, we assume that agents are weakly-coupled [16], i.e., they only interact through the shared resources, and once the resources are allocated, the agents' transitions and rewards are independent. Also, to simplify the following discussion, we assume that agents' resource requirements are binary (as discussed in Section 3.1), but all results directly map to to the non-binary case by making straightforward substitutions from Section 3.

The problem of distributing a set of indivisible resources $\mathcal{O}$ among a set of agents $\Lambda = \{\lambda\}$ is described as follows:

- $\{\langle \mathcal{S}, \mathcal{A}, P^\lambda, R^\lambda, \alpha^\lambda, \rho^\lambda, \widehat{\kappa}^\lambda\rangle\}$ is the collection of single-agent factored weakly-coupled MDPs, as defined in Section 3. For simplicity, but without loss of generality we assume that all agents have the same state and action spaces $\mathcal{S}$ and $\mathcal{A}$, but each has its own transition and reward functions P^λ and R^λ, initial conditions α^λ, as well as its own resource requirements $\rho^\lambda : \mathcal{A} \times \mathcal{O} \mapsto \{0,1\}$ and capacity bounds $\widehat{\kappa}^\lambda : \mathcal{C} \mapsto \mathbb{R}$.
- $\widehat{\rho} : \mathcal{O} \mapsto \mathbb{R}$ specifies the upper bound on the amounts of the shared resources (this defines the additional bound for the multiagent problem).

Given the above model, the goal is to find an allocation of the available resources among the agents that maximizes the sum of the expected discounted rewards of the agents' optimal policies under the resource allocation. This can be achieved via a straightforward combination of the single-agent MILPs from the previous section, augmented with an additional constraint that ensures that the allocation does not exceed the bounds on the total amounts of resources $\widehat{\rho}$.

Putting it all together, we arrive at an MILP that achieves the goal we set forth in this work – an algorithm for simultaneously solving the resource-allocation and the resource-parameterized policy-optimization problem for the factored

Markov model. The following MILP has, for every agent λ, a set of marginal occupation measures ξ^λ and a set of binary variables $\delta^\lambda(o)$. Each agent has its own basis functions h_k^λ, and therefore its own backprojections g_k^λ.

$$\max \sum_\lambda \sum_a \sum_j^{J_R} \sum_{\mathbf{z}_{r_j}^\lambda} r_j^\lambda(\mathbf{z}_{r_j}^\lambda, a)\xi_{r_j}^\lambda(\mathbf{z}_{r_j}^\lambda, a) \qquad \text{s.t.:}$$

$$\sum_a \xi_{h_k}^\lambda(\mathbf{z}_{h_k}^\lambda, a)h_k^\lambda(\mathbf{z}_{h_k}^\lambda) - \gamma \sum_a g_k^\lambda(a)\xi_{g_k}^\lambda(\mathbf{z}_{h_k}^\lambda, a) = \sum_{\mathbf{z}_{h_k}} \alpha^\lambda(\mathbf{z}_{h_k}^\lambda)h_k^\lambda(\mathbf{z}_{h_k}^\lambda), \quad \forall k \in [1, K],\ \lambda \in \Lambda;$$

$$\sum_{\widetilde{\mathbf{z}_j^\lambda}} \xi_j^\lambda(\mathbf{z}_j^\lambda, a) = \sum_{\widetilde{\mathbf{z}_i^\lambda}} \xi_i^\lambda(\mathbf{z}_i^\lambda, a), \quad \forall \lambda \in \Lambda, a \in \mathcal{A}, \xi_j^\lambda, \xi_i^\lambda, \mathbf{z}_j^\lambda, \mathbf{z}_i^\lambda, \widetilde{\mathcal{Z}_i^\lambda} = \mathcal{Z}_j^\lambda \setminus \mathcal{Z}_i^\lambda,\ \widetilde{\mathcal{Z}_j^\lambda} = \mathcal{Z}_i^\lambda \setminus \mathcal{Z}_j^\lambda;$$

$$\sum_a \sum_{\mathbf{z}_j^\lambda} \xi_j^\lambda(\mathbf{z}_j^\lambda, a) = (1-\gamma)^{-1}, \quad \forall j,\ \mathbf{z}_j;$$

$$\sum_o \kappa(o, c)\delta^\lambda(o) \le \widehat{\kappa}^\lambda(c), \quad \forall \lambda \in \Lambda, c \in \mathcal{C};$$

$$\sum_{j=1}^{J_e} \sum_a \sum_{\mathbf{z}_{\rho_j}^\lambda} \rho_j^\lambda(\mathbf{z}_{\rho_j}^\lambda, a)\xi_{\rho_j}^\lambda(\mathbf{z}_{\rho_j}^\lambda, a) \le \delta^\lambda(o), \quad \forall \mathbf{z}_j^\lambda, o; \tag{21}$$

As in the flat-MDP model of [8], this MILP has $|\Lambda||\mathcal{O}|$ binary variables, which is an exponential reduction from a naive resource-allocation scheme that has a decision variable per resource bundle (thus requiring on the order of $|\Lambda|2^{|\mathcal{O}|}$ variables). Furthermore, the number of occupation-measure variables can be drastically reduced in this factored MILP, compared to the flat variant. The actual gains in efficiency and the associated losses in quality depend on what basis functions h_k are used and on how well-structured the MDP is (which ultimately defines the domain sizes of ξ).

5. EXPERIMENTAL RESULTS

We investigated the behavior of the resource-allocation MILP (21) on the factored "SysAdmin" problem [12], which we extended to include resources and capacities.

The problem involves a network of N computers, each of which might fail with probability that depends on the status of the neighboring computers. The job of the decision-maker (the SysAdmin) is to keep the network running. The state of the system is defined by a set of N binary features, each corresponding to the status of a computer. At each time step the SysAdmin can reboot at most one computer and receives a reward for every computer that is functioning properly. We used a network with a unidirectional-ring topology, where each computer influenced and was influenced by the status of a single other computer. The parameters of the transition and reward model are described in detail in [12].

We extended the domain to the multiagent setting with resources as follows. Each SysAdmin agent was in charge of running its own network of N computers. In our domain, computer failures were caused by more serious malfunctions (e.g., a virus infection) that could not be fixed by simply rebooting the computer, and thus performing the recovery actions required resources (e.g., specialized software, hardware, or trained personnel). Further, the computers were heterogeneous (mail servers, web servers, data storage, etc.), and thus the recovery of each computer required a different subset of the resources. The total quantity of the resources was limited, and each SysAdmin also had a limited budget, which constrained the resources it could acquire.

For a local MDP with N computers, there were $|\mathcal{O}| = N$ resources, and each recovery action required two resources, which were randomly selected at the time of problem generation. There was a single capacity cost c (money) and all resources had a unit capacity cost $\kappa(o, c) = 1$.

A thorough empirical analysis of ALP solutions to the SysAdmin problem is presented in [12], and it discusses the quality of solutions and the computational complexity of solving the ALP for various network topologies and different ways of selecting the basis functions. In particular, the experiments in [12] demonstrate that using basis functions over singletons of features leads to high-quality approximations at a very low computational cost. Therefore, we used the same basis functions and focused on evaluating the computational complexity of the resource-allocation MILP (21).

Figure 2a shows the running time of the MILP (21) as a function of the number of state features (computers). As can be seen, the factored MDP approach allows us to easily scale the resource-allocation problem up to MDPs with more than 10^{15} states. Figure 2b demonstrates the corresponding exponential gain in efficiency of the factored MILP (21), compared to the one with a flat-MDP model. Note that the flat MILP [8] in itself commonly achieves exponential speedup over a straightforward resource-allocation mechanism that enumerates all resource bundles [9].

Finally, Figure 2c shows the complexity of the factored MILP as a function of the number of agents. The graph demonstrates that although the number of binary variables in the MILP is proportional to the number of agents, the approach shows promise for scaling to large numbers of agents.

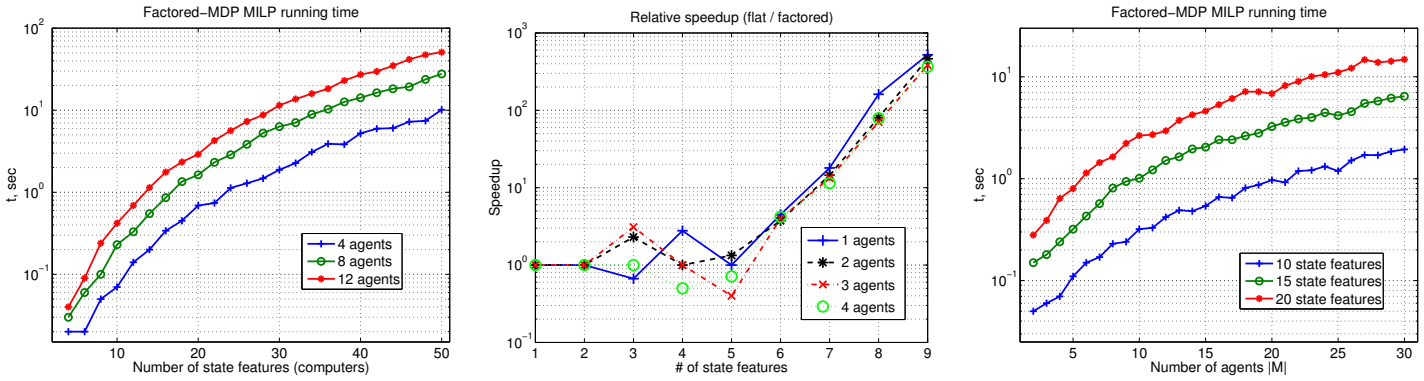
Overall, the trends in these results support the analytical claims about the efficiency gains of using factored models and ALP in the resource-allocation problem.

6. CONCLUSIONS

Recent work on resource-allocation problems has demonstrated that it can be extremely beneficial from the standpoint of computational efficiency to have a good model of the processes that define agents' preferences over resources. For a realistic agent that has to deal with uncertainty, Markov decision processes with action spaces that depend on the resources available to the agent provide a natural model of the agent's utility-formation mechanism. Moreover, resource preferences induced by MDPs are well-structured, which enables the construction of resource-allocation mechanisms that are very computationally efficient[8, 9].

We presented a mechanism that fruitfully exploits the structure in agents' utility functions that is due to the underlying MDPs, as well as the structure within the MDPs themselves (via an approximation). For well-structured MDPs, the latter can lead to another order of exponential speedup over flat MDP methods, as supported by our experiments.

Some interesting directions of our ongoing and future work in this area include exploring factored resource-allocation MILPs on cluster graphs that are not junction trees, as well as looking at other types of structure in agent's MDPs (e.g., context-specific structure [12]).



(a) (b) (c)

Figure 2: Efficiency of resource allocation with factored MDPs. (a): Running time of factored MILP as a function of the number of state features. (b): Relative speedup of factored MILP, compared to the flat MILP (ratio of running times). (c): Running time of the factored MILP as a function of the number of agents.

7. ACKNOWLEDGMENTS

This material is based upon work supported in part by the DARPA/IPTO COORDINATORs program and the Air Force Research Laboratory under Contract No. FA8750–05–C–0030. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

8. REFERENCES

[1] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, 1987.

[2] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[3] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the 14th International Joint Conference on AI*, pages 1104–1111, 1995.

[4] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *AI*, 121(1-2):49–107, 2000.

[5] D. de Farias and B. Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478, 2004.

[6] D. P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6), 2003.

[7] D. A. Dolgov and E. H. Durfee. Graphical models in local, asymmetric multi-agent Markov decision processes. In *Proc. of the 3rd Int. Joint Conf on Autonomous Agents and Multiagent Systems*, 2004.

[8] D. A. Dolgov and E. H. Durfee. Optimal resource allocation and policy formulation in loosely-coupled Markov decision processes. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling*, pages 315–324, 2004.

[9] D. A. Dolgov and E. H. Durfee. Computationally efficient combinatorial auctions for resource allocation in weakly-coupled MDPs. In *Proc. of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-05)*, 2005.

[10] D. A. Dolgov and E. H. Durfee. Symmetric primal-dual approximate linear programming for factored MDPs. In *Proc. of the Ninth Int. Symposium on AI and Math. (AI&M-06)*, January 2006.

[11] C. Guestrin. *Planning Under Uncertainty in Complex Structured Environments*. PhD thesis, CS Dept., Stanford University, August 2003.

[12] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of AI Research*, 19:399–468, 2003.

[13] M. I. Jordan. Graphical models. *Statistical Science (Special Issue on Bayesian Stat.)*, 19:140–155, 2004.

[14] D. Koller and R. Parr. Computing factored value functions for policies in structured MDPs. In *Proc. of the Sixteenth International Conference on Artificial Intelligence IJCAI-99*, pages 1332–1339, 1999.

[15] K. Larson and T. Sandholm. Mechanism design and deliberative agents. In *Proc. of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 650–656. ACM Press, 2005.

[16] N. Meuleau, M. Hauskrecht, K.-E. Kim, L. Peshkin, L. Kaelbling, T. Dean, and C. Boutilier. Solving very large weakly coupled Markov decision processes. In *AAAI/IAAI*, pages 165–172, 1998.

[17] R. Patrascu, P. Poupart, D. Schuurmans, C. Boutilier, and C. Guestrin. Greedy linear value-approximation for factored Markov decision processes. In *Proc. of the 18th National Conf. on AI*, pages 285–291, 2002.

[18] M. L. Puterman. *Markov Decision Processes*. John Wiley & Sons, New York, 1994.

[19] M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinational auctions. *Management Science*, 44(8):1131–1147, 1998.

[20] P. Schweitzer and A. Seidmann. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110:568 582, 1985.

Multi-Agent Based Peer-to-Peer Information Retrieval Systems With Concurrent Search Sessions

Haizheng Zhang, and Victor Lesser
Dept. of Computer Science, University of Massachusetts Amherst, MA 01003
hzhang@cs.umass.edu, lesser@cs.umass.edu

ABSTRACT

In cooperative peer-to-peer information retrieval systems, each node can be considered an intelligent agent and these agents work collectively to provide an information retrieval service. In order to effectively support multiple and concurrent search sessions in the network, we propose two traffic engineering techniques that minimize processing and communication bottlenecks. One is a novel agent control mechanism whose elements include resource selection, local search scheduling, and feedback-based load control. The other is a new two-phase query routing algorithm based on organizational knowledge. Experimental results show that this framework can reduce congestion situations, increase system throughput, and improve considerably the overall system utility.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms

Keywords

peer-to-peer information retrieval, agents, control

1. INTRODUCTION

In recent years, peer-to-peer based information retrieval(IR) systems have begun to receive considerable attention. A peer-to-peer based information retrieval system consists of a set of nodes connected in a peer-to-peer fashion. Each node hosts a document collection to share with other nodes and these nodes work together to provide information retrieval service to users. In this paper, we consider an information retrieval task as a search session during which nodes forward queries to neighbors, perform IR operations and/or return search results. The lack of complete, up-to-date information about the states of other nodes in the network requires sophisticated coordination strategies. In addition, the presence of concurrent search sessions adds another level of complication due to bandwidth and processing capacity limitations: nodes may not be able to complete forwarding and local searches in a timely fashion for all queries they have received. Therefore, nodes have to decide what actions to take for each search session and in what order to execute those actions in order to maximize overall system utility. These local decisions, taking place at each agent, collectively determine the way concurrent queries are distributed and processed in the network.

In this paper, we extend a multi-agent based infrastructure [11, 12] developed for a single search session such that it can handle multiple, concurrent search sessions. These extensions required the development of a new agent control mechanism and a novel coodinated query routing algorithm based on a queueing theory analysis. Both mechanisms control the flow of information from different perspectives in the network to exploit available resources while reducing the occurrence of congestion situations. The purpose of the local control mechanism, based only on the agents' local observations of their neighboring agents, is to improve the average effective propagation speed of search sessions, a concept that will be introduced in later sections. The elements of such a control mechanism include resource selection, local search scheduling and feedback-based load control. The coordinated search algorithm is a modified version of a previously developed two-phase query routing algorithm [12] that exploits static agent organizational knowledge to locate relevant documents quickly while at the same time minimizing hot-spots caused by concurrent search sessions. Experimental results show that this framework can reduce the frequency of congestion situations, increase system throughput, and improve overall system quality.

The contributions of this work include: (1) a novel agent control mechanism based on a queueing theoretical analysis; (2) a simple feedback-based probabilistic load control component to avoid congestion situations while dynamically exploiting available communication and processing bandwidth; (3) a distributed, two-phase query routing algorithm based on a hierarchical structure of agents that balances search traffic in the network.

We make the following assumptions in this paper. First, each agent maintains an independent index and an IR search engine for its local document collection. However, we do not introduce any further restrictions on the local search engines and thus the network can be populated by agents having very different local search engines. Second, the experimental

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

results presented are based on local search engines that are "perfect" in that they return all relevant documents in the collection for a given query. Third, we assume there is a third-party protocol in place to merge the returned results. Thus, our protocol does not have to deal with the merging of the returned lists. Fourth, we assume that the document collections hosted on agents are disjoint, and therefore the total number of relevant documents for a certain query is the sum of the relevant documents returned from each agent. Lastly, we assume that agents are cooperative in that they all agree to use the same protocols for propagating resource descriptions among each other, accepting queries from peers and finally returning search results to the originators of the queries

The remainder of this paper is organized as follows: Section 2 discusses related research; Sections 3 and 4 introduce the system architecture of a P2P information retrieval system and the individual agent structure respectively. Section 5 presents the local scheduling algorithm in each agent. Section 6 introduces a two-phase search algorithm based on a hierarchical agent organization; Section 7 provides the experimental settings and explains the results; Section 8 discusses future work, and Section 9 summarizes the major contributions of this paper.

2. RELATED WORK

This paper is related to two lines of research: (1) query routing algorithms in peer-to-peer information retrieval systems, and (2) traffic engineering techniques in IP-Level and application-level network research.

The first line of relevant research includes recent studies on pure peer-to-peer information retrieval systems and hierarchical peer-to-peer systems [10, 7, 11, 12]. Lu proposed a framework including resource selection, resource representation and result merging in a hierarchical P2P environment[7]. In particular, a language model based search algorithm is employed in a top-down fashion. Indeed, as a query routing process can be considered as a distributed search process, it is natural to study this system in a multi-agent framework [5, 11, 12]. However, most of these research efforts do not consider situations with multiple, concurrent search sessions which is the focus of this paper.

The second line of relevant research is on network traffic engineering. Traffic engineering, which has mostly concerned IP-level routing problems, involves adapting the routing of traffic to network conditions, with the joint goals of good user performance and efficient use of network resources. The most well studied work along this line is on congestion control which attempts to minimize delays in Internet routing[3]. While query routing in P2P based information retrieval differs considerably from IP level routing problems, there are similarities that can be exploited in query routing algorithms.

3. PROBLEM DESCRIPTION

All agents in a peer-to-peer information retrieval system constitute a graph $G(A, E)$. Set A is the set of all agents in the system while set E includes all connections among agents. A search session s_i starts when an agent A_j receives a query qry_k from a user at time t_l. During the search process, upon receipt of the query qry_k, agents conduct local searches, forward the query to their neighbors and return search results to agent A_j. Agent A_j keeps a set of result tuples $T_m(s_i, A_m, t, Q_{s_i,A_m})$, with each element in the set specifying the fact that an agent contributes a certain amount of quality, Q_{s_i,A_m} in this case, to search session s_i at time t. The search conducted by agent A_m can be evaluated off-line by considering the ratio of the number of relevant documents returned from agent A_m to the total number of relevant documents in the network. Notice that the size of the result tuple set may keep growing with more search results returned from agents. The cumulative quality for a search session at time t is therefore the sum of the quality fields in the result set T_k obtained at time t. Formally, it is defined as:

$$Q_{s_i}(t) = \sum_{A_k} Q_{s_i,A_k} \text{ where } T_k.t < t$$

In the search process, agents discard those queries they have previously processed thereby preventing queries from looping in the system forever. The search session ends when all the agents that receive the query drop it. We define the period from a query entering the system to its leaving the system as the duration of the search session, t^*. Therefore, the maximal quantity for a search session is defined as $Q_{s_i}(t^*)$. The duration of a search session depends on factors including the connectivity of the agent organization, the query routing algorithm and local scheduling algorithms in the agents. In general, a long search session duration indicates the occurrence of hot spots in the network. These hot spots are usually characterized by communication congestion or long query queue length in agents.

In this paper, we focus on open information retrieval systems, meaning that queries can enter the network at any agent from outside users, thereby forming multiple concurrent search sessions in the system. In our model, we define the arrival rate of jobs from outside to the ith node of the network to be λ_{0i}. The distribution of the incoming queries to a single agent conforms to a Poisson distribution, and the overall arrival rate λ from outside to an open network is:

$$\lambda = \sum_{i=1}^{N} \lambda_{0,i}$$

In the presence of multiple concurrent search sessions, we define the overall system utility, GQ, as the sum of the cumulative utility for all the search sessions including both the outstanding search sessions and finished search sessions under the assumption that all search sessions are considered equally important:

$$GQ(t) = \sum_{s_i} Q_{s_i}(t)$$

4. AGENT INTERNAL STRUCTURE

In this section, we describe the internal structure and local scheduling algorithms of each individual agent. Figure 1 illustrates the four components of our agent architecture: a document collection with its associated collection descriptor, a search engine, an agent view structure and a control unit. This section describes the first three components while Section 5 describes the control unit.

The document collection hosted on an agent includes a set of documents to share with other peers. The local search engine allows each agent to conduct local searches and return

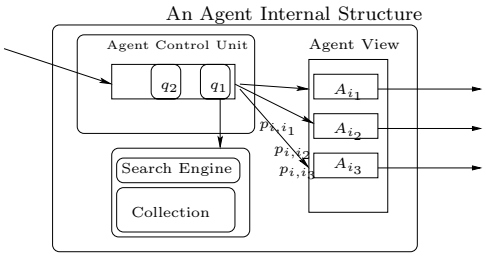


Figure 1: Internal Agent Structure

relevant documents upon receiving user requests. The collection can be described by a collection descriptor, which can be considered as the "signature" of the collection. By distributing collection descriptors, agents can gain knowledge about how content is distributed in the agent network. A collection descriptor characterizes the distribution of the vocabulary in the collection. It is based on the language model concept[9] that has proved effective in various previous studies in the distributed IR field [1, 7, 11]. A language model has many interesting properties which are easily exploitable in the peer-to-peer network system. Most importantly, it is a lightweight, concise and accurate descriptor of the document collection. Additionally, the size of the collection model is almost independent of the size of the document collection.

The agent view structure is an agent's restricted view of the network and determines the underlying topology of the agent "virtual" organization. Each agent's view structure contains the collection descriptors of a set of other agents' collections as well as the addresses of these agents and other related information such as the connection type. The design of the agent view structure depends on the underlying topological agent organization. In a power law based flat agent organization, only one connection type is needed as all agents play the same role. On the other hand, there can be multiple connection types in a hierarchical peer-to-peer agent organization, including links connecting upper level agents to lower level agents, lower level agents to upper level agents and agents at the same level. The specifics of the connection type information can be used in deciding how to route queries. More information about the agent view structure and the role it plays in search process is detailed in [11, 12].

5. AGENT LOCAL CONTROL MECHANISMS

The information retrieval performance, when there are concurrent search sessions, depends on how resources are allocated among these search sessions. The lack of centralized control of these concurrent searches can lead to an uneven distribution of query load among agents; this uneven load can then cause the situation where some agents are flooded with incoming queries while other agents are largely idle. Mitigating this problem requires agents to be able to control the message flow from a local perspective to prevent congestion and speed up the propagation speed of search sessions in the network. In addition to congestion prevention, in order to maximize overall system utility, agents should disseminate queries only to relevant agents so as to exploit bandwidth efficiently and reduce communication and processing load.

In this section, we present an analytical model based on queueing theory for agent control. In particular, the control unit of an agent makes decisions regarding which queries to process locally, in what order to process them (local search), and whether and to whom should the queries be forwarded (resource selection).

5.1 Resource Selection Algorithms

The resource selection algorithm chooses a subset of neighboring agents (that may contain relevant documents or have information about the location of relevant documents) to forward a query for further processing. Resource selection algorithms are characterized by their IR calculation overhead, the communication load they induce and how "focused" the search is. Existing resource selection algorithms include broadcast based routing algorithms, probabilistic based random routing algorithms and language model based algorithms[11, 12, 7]. Each resource selection algorithm incurs a certain amount of cost depending on its specific operations. For instance, a broadcast algorithm needs the least calculation, incurs minimal selection overhead but can easily saturate the network with messages, whereas a probabilistic random approach saves communication load but increases local computation load. A language model based approach has a different trade-off. It requires relatively expensive comparison operations but can significantly reduce the number of messages propagating in the system. Expensive computation operations can reduce agents' throughput which will be described in the next sections. Therefore, in the presence of concurrent search sessions, it is not always preferable to use a resource selection algorithm that generates minimal communication load but requires expensive comparison operations. The trade-off between message number and the amount of computation will be analyzed in the following sections.

5.2 A local scheduling mechanism

Each individual agent control unit maintains several queues including message queues, which contain messages to be forwarded to specific agents, and a local query queue, which contain queries waiting for further local processing. The set of agent control units in the network constitutes a queueing network system. In this section, two approaches for organizing the queueing network are introduced and compared based on their utilization rate and throughput.

Several assumptions are made in order to simplify the analysis.

(1) The service times at each queue including resource selection, local search scheduling and message forwarding are exponentially distributed.

(2) The inter-arrival times of queries are random and have Poisson distribution.

(3) The channel capacity between two arbitrary agents, A_i and A_j, is fixed, denoted as C_{ij}.

(4) The length of queries in the system, L, is considered a constant. This assumption simplifies the calculation of the throughput of communication channels.

Figure 2 shows an agent's control unit and indicates how resource selection, local searches and forwarding for parallel queries are related. This model consists of Q_{rs}, a queue that contains queries to be forwarded to other neighboring agents, Q_{ls}, a queue that contains queries waiting for further local search operations, and message forwarding queues Q_{1j} for each agent A_j in the agent view structure. These queues are considered as M/M/1 queues based on the above assumptions.

The arrival rate λ_i for queue Q_{rs} and Q_{ls} in agent A_i is calculated based on the departure rate, λ_{ji}, from its upstream agent A_j, and the arrival rate, λ_{0i}, from the outside users to agent A_i,

$$\lambda_i = \lambda_{0i} + \sum_{j=1}^{N} \lambda_{ji} \quad (1)$$

for i = 1, .., N

However, in a practical system, agents discard those queries that have been processed previously to ensure loopless communication. We assume the arrival rate of non-redundant queries is $\lambda_i^{'}$.

The resource selection time spent on a query is denoted as t_{rs} and the service time for local search is t_{ls}. Recall that both service times satisfy exponential distribution.

Queue Q_{rs} and Q_{ls} have the same arrival rate as specified in Equation 1. The throughput for Q_{rs} is $\frac{1}{t_{rs}}$ and for Q_{ls} is $\frac{1}{t_{ls}}$. Note that by separating the queue into two queues, the local search operation is no longer the bottleneck. Even when the incoming rate is higher than the service rate of Q_{ls}, an agent can buffer the incoming queries for later processing.

The arrival rate for each message queue Q_{ij}, λ_{ij} is the number of messages departing from Q_s in a time unit, i.e $\lambda_{ij} < \lambda_i^{'}$. The utilization rate for message queue Q_{ij} is $\rho_{Q_{ij}} = \frac{\lambda_{ij}}{\mu_{Q_{ij}}}$ And the throughput for Q_{ij} is

$$\mu_{Q_{ij}} = C_{ij}/L \quad (2)$$

.

Because each agent contributes to the overall system utility for each finished local search, the performance of the scheduling algorithm for the local search queue determines how fast the overall utility is accumulated. Moreover, because the selection of one query is always at the expense of executing other search sessions at a later time, this algorithm also affects the fairness of the network search process. It is worth mentioning that local fairness does not necessarily lead to global fairness.

A simple scheduling algorithm for the local query queue is first come first serve(FCFS) where agents always process queries that enter the system earliest. Although the FCFS algorithm is fair from an agent's local perspective, it can cause significant delay for queries that enter the system at a later time. An alternative scheduling approach is based on a greedy algorithm that aims to maximize the overall utility from an agent's local perspective.

$$GQ(t) = \sum_{s_i} Q_{s_i}(t) = \sum_{A_i} Q_{A_i}(t)$$

$Q_{A_i}(t) = \sum_{s_i} Q_{s_i,A_i}$ is defined as the local accrued utility of agent A_i. Therefore, in order to maximize the overall utility, each agent selects the most relevant subset of queries to process. In order to maximize the local utility, we use a heuristic based sorting algorithm to order the queries queue according to the Kullback-Leibler (KL) distance between the query model and the local collection model. KL distance is a metric to evaluate the distance of two distributions. The calculation detail is introduced in [12].

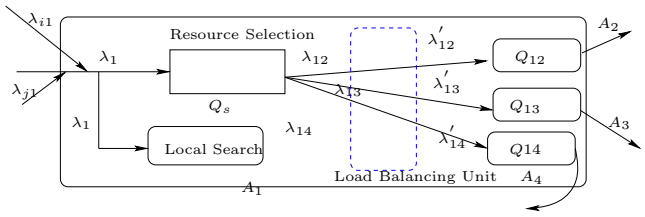


Figure 2: The agent internal structure with a parallel control mechanism

5.3 A Feedback-based, Probabilistic Load Control Unit

The above section analyzed the behavior when the queueing network is stable. However, when the query arrival rate is more than the service rate, the queue length will be ever expanding. In this section, we first analyze the cause of "hot spots" in the above model and then present a feedback based probabilistic load control mechanism in order to avoid or mitigate congestion situations.

5.3.1 The creation of "Hot Spots"

In the above model, there are two cases that would create a congestion situation: (1)The arrival rate for message queues (Q_{12}, Q_{13}, etc) is above the service rate, i.e

$$\lambda_{ij}^{'} > \frac{C_{ij}}{L}$$

(2)The arrival rate for queue Q_{rs} is more than the service rate, i.e:

$$\lambda_i > \frac{1}{t_{rs}}$$

From equation 1, it is equivalent to:

$$\lambda_i = \lambda_{0,i} + \sum_{j=1}^{N} \lambda_{ji} > \frac{1}{t_{rs}}$$

In the following section, we propose a load control mechanism to avoid both situations, thereby improving overall system performance.

5.3.2 A Probabilistic Load Control Algorithm

In order to control the communication load based on limited observations of the network, agents can employ a feedback-based, probabilistic algorithm. Specifically, when an agent forwards queries to its neighboring agents, it not only considers the capacity of its own communication channels, but also takes into account its neighboring agents' service rates, which are acquired dynamically by analyzing the feedback information sent out by its neighboring agents periodically.

Let QS_{ij} be the query set that the resource selection algorithm of agent A_i selects to forward to agent A_j and QS_i

be the query set that agent A_i receives and has not processed previously in a certain time period t. Assume that in a certain period, queue Q_s receives non-redundant query set QS_i and the load control mechanism randomly selects $p_{ij} * |QS_{ij}|$ from the query set QS_{ij} and $q_{ij} * |QS_i - QS_{ij}|$ from the set $QS_i - QS_{ij}$. Notice that $|QS_{ij}| = \lambda_i^{'} * t$ and $|QS_{ij}| = \lambda_{ij} * t$

Agent A_i determines probability p_{ij} and q_{ij} in the following way:

(1) $q_{ij} >= 0$ only if $p_{ij} = 1$.

(2) $p_{ij} = min(p_1, p_2, 1)$ where

$$p_1 = \frac{\frac{C_{ij}}{L}}{|QS_{ij}|}$$

and

$$p_2 = \frac{f_{ij}}{|QS_{ij}|}$$

(3) q_{ij} is determined by:

$$q_{ij} = \frac{min(f_{ij}, \frac{C_{ij}}{L}) * t - |QS_{ij}|}{|QS_i - QS_{ij}|}$$

(4) f_{ij} is determined by agent A_j and forwarded to agent A_i as feedback to control the number of messages to be forwarded to agent A_j from agent A_i. In our work, f_{ij} is determined by

$$f_{ij} = t_{ij} * \lambda_{ij} * t$$

and

$$t_{ij} = \frac{\frac{1}{t_{rs}} - \lambda_{0j}}{\lambda_j - \lambda_{0j}}$$

Recall that λ_{0i} is the arrival rate of queries from outside users instead of other agents. t_{ij} characterizes the overload situation at agent A_j

(1) specifies that agents selected by the resource selection algorithm have higher priority over other agents. Agent A_i would only forward a query to more agents when the utilization rate of communication channel is lower than 1.

(2) prevents the situation where too many queries are scheduled for the communication channel between A_i and A_j. In particular, p_1 is the maximal capacity of the communication channel and p_2 is the maximal number of queries that agent A_j can process in a timely fashion for agent A_i.

(3) decides that when the query arrival rate is low, the load control unit should increase the utilization of the communication channel and attempt to increase the speed that search sessions propagate in the system by forwarding more queries to agent A_j than the set that was selected by the resource selection algorithm.

(4) describes how the feedback information is generated. Agent A_j requests the upstream agents to reduce communication load by t_{ij} in order to prevent congestion situations.

6. A BALANCED TWO-PHASE QUERY ROUTING ALGORITHM

The control mechanism described above engineers traffic flow in a distributed information retrieval system from a local perspective. The effect of this mechanism, however, could be limited by the fact that agents only possess a narrowly defined non-local view of content network traffic. In this section, we introduce a query routing algorithm which exploits a more encompassing but static view of the network topology to route queries. It is the interaction among these two mechanisms that is key to our approach.

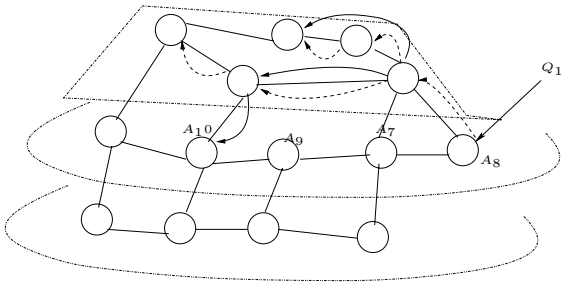


Figure 3: A Two-Phase Query Routing Algorithm In Hierarchical P2P IR System

In our previous work[12], we proposed a hierarchical agent topology and a two-phase search algorithm for use in routing queries. In this section, we extend this work to make it suitable for concurrent query processing in the network. In our hierarchical agent society, agents have two roles: group-mediator and query-processor[12]. All non-leaf agents in the organization take on both roles while leaf agents only take on the role of query-processor. Each mediator manages a group of agents and takes on a central role in group management including decisions on whether a new agent should be added to the group, when to reorganize the group, the selection of group members to handle a query and the propagation of queries to non-group members.

To take advantage of hierarchical agent organization, a two-phase hierarchical search algorithm was developed in [12]. In this protocol, as illustrated in Figure 3, agents in the first phase forward queries to top-level mediators so as to find appropriate starting points for the search. In the second phase, a concurrent search is initiated from these starting points. The intention of this algorithm was to conduct searches in a focused fashion to reduce communication messages. Notice that the load may become unbalanced in such a network. The arrival rate for a top-level mediator A_i is:

$$\lambda_i = \lambda_{0,i} + \sum_{j \in \{A_i^{'}\}} \lambda_{0,j} + \sum_{j=1}^{N} \lambda_j p_{ji}$$

for i = 1, .., N. Here we denote $\{A_i^{'}\}$ as the set of all direct or indirect members of agent A_i. In the above equation, $\lambda_{0,i}$ is the query arrival rate from users, while $\sum_{j \in \{A_i^{'}\}} \lambda_{0,j}$ is the query arrival rate coming from the member set of agent A_i; $\sum_{j=1}^{N} \lambda_j p_{ji}$ is the query arrival rate coming from the neighbors set. This makes the resource selection algorithms very expensive since the selection algorithm at top-level mediators must perform comparisons of query model and collection model of each agent. This algorithm is not a balanced approach since it tends to increase dramatically the burden of top-level mediators, thereby creating hot spots in the network. To resolve this problem, we propose an improved

two-phase search algorithm as demonstrated in Figure 4.

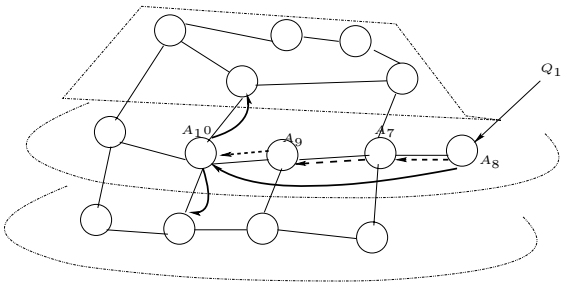


Figure 4: A Balanced Two-Phase Query Routing Algorithm

The first phase of this query routing algorithm is primarily conducted in the horizonal direction. It starts when the initiator of the query, A_8, forwards the query similarity probe messages with a certain TTL (Time To Live) value along the lateral links to the agents at the same level to locate relevant clusters. Upon receipt of these probe messages, agents return back the similarity values of their collections with the queries in question. After the TTL value expires, an agent does not forward the query any further, thus stopping further search along this path. After comparing the similarity returned by the agents, the initiator, $A8$, selects the K most similar agents to proceed to the second phase of the search. The second search phase is primarily conducted inside each group and information flows in the vertical direction. In particular, agents only forward the query along upward links or downward links during the second phase. For evaluation purposes, there is no explicit recognition by individual agents that the query is no longer being processed by any agent in the network. This process continues in the network until all the agents receiving the query drop the message or there are no other agents to forward to. In reality, this phase can also be controlled by a TTL value to limit search efforts. This way, the traffic load is likely to be more evenly distributed among the various levels of the hierarchy given that the entry points of the queries are randomly distributed in the network. Hence, the new search strategy mitigates the hot spot problem encountered in the previous algorithm.

7. EXPERIMENTAL SETTINGS AND RESULTS ANALYSIS

7.1 TRANO Testbed

TRANO(Task Routing on Agent Network Organization) is a multi-agent based information retrieval testbed. TRANO is built on top of the Farm simulator[6] that provides a data dissemination framework for large scale distributed multi-agent organizations. TRANO supports importation and exportation of agent organization profiles including topological connections and other features. Each TRANO agent is composed of an agent view structure and a control unit. In simulation, each agent is pulsed regularly and the agent checks the incoming message queues, performs local operations and then forwards messages to other agents .

7.2 Experimental Settings and Results Analysis

In our experiment, we use TREC-VLC-921 dataset which contains 921 sub-collections to simulate the collections hosted on agents. TREC-VLC-921 was originally split from TREC VLC1 collection by data sources in order to create testbed for distributed information retrieval research[2]. TREC VLC1 is part of the TREC collections which are distributed by the National Institute of Standards and Technology (NIST) for testing and comparing current text retrieval techniques. TREC VLC1 (very large collection) includes documents from 18 different data sources, such as news, patents, and the Web[4]. The formation of hierarchical agent organization can be achieved by a hierarchical clustering algorithm mentioned in [7]. However, this is under the assumption that global information about the content distribution is available during the topology formation. In the absence of global information, [12] proposed a distributed hierarchical clustering algorithm to form a hierarchical organization which is exploited in this paper. During the topology generation process, degree information of each agent is estimated by the algorithm introduced in [8] with parameters $\alpha = 0.5$ and $\beta = 0.6$. In our experiments, we estimate the upward limit and downward degree limit using linear discount factors 0.5, 0.8 and 1.0. Once the topology is built, queries randomly selected from the query set $301 - 350$ on TREC-VLC-921 are injected to the system based on a Poisson distribution

$$P(N(t) = n) = \frac{(\lambda t)^n}{n!} e^{-\lambda}$$

In addition, we assume that all agents have an equal chance of getting queries from the environment, i.e, λ is the same for every agent. In our experiments, λ is set as 0.00543 so that the mean of the incoming queries from the environment to the agent network is 5 per time unit. The service time for communication queue Q_s and Q_{rs}, i.e $t_{Q_{ij}}$ and t_{rs}, is set as 0.2 time unit. The service time for a local search queue, t_{ls} is 3 time units.

We use KL divergence to measure the distance among collection models or between collection models and query models. The formula is:

$$D(p||q) = \sum_i p(i) \log \frac{p(i)}{q(i)}$$

An approximation approach is used to speed up the calculation of KL distance and convert it to a similarity measure. The calculation details can be found in [11].

7.3 Performance Measures

The following measures were used to compare the different search strategies:

Propagation Speed of a search session s_i, $ps_{s_i}(t)$ is defined as the ratio of the number of agents visited at time t, $N(t)$, over the size of the agent society N, i.e $ps_{s_i}(t) = \frac{N(t)}{N}$. Average propagation speed is defined as average of the propagation speed of all the search sessions in the network. Notice that t is the amount of time after a search session starts. The maximal proportion of agents that a message visits is $ps_{s_i}(t^*)$.

Effective Propagation Speed of a search session s_i, $eps_{s_i}(t)$ is defined as the ratio of the number of relevant documents hosted on visited agents at time t,

Table 1: Experimental Results For Various Search Strategies

Name	Topology	RS	LC	λ	t^*	$ps_{s_i}(t^*) * t^*$	$eps_{s_i}(t^*) * t^*$	GU	MsgNumber	$GQ(t^*)$
PBN	**P**owerLaw	**B**roadcast	**N**o	0.00543	32	97%	97%	92%	2603022	77.29
PBY	**P**owerLaw	**B**roadcast	**Y**es	0.00543	31	97%	96%	94%	2563046	78.99
HTN	**H**ierarchical	**T**woPhase	**N**o	0.00543	17	29%	33.5%	44%	353446	38.01
HTY	**H**ierarchical	**T**woPhase	**Y**es	0.00543	16	32%	35%	50%	407242	45.20
HBN	**H**ierarchical	**B**alanced	**N**o	0.00543	26	89%	92%	80%	1512011	83.94
HBY	**H**ierarchical	**B**alanced	**Y**es	0.00543	24	89%	98%	85%	1552011	88.94

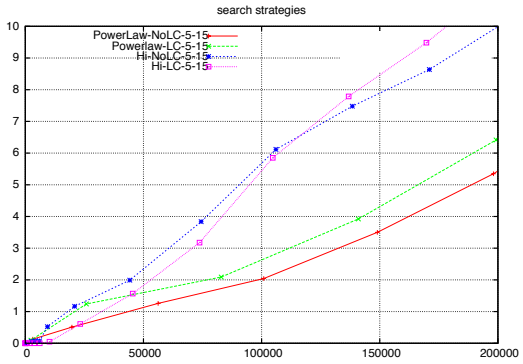


Figure 5: $GQ(t)$ versus the message number

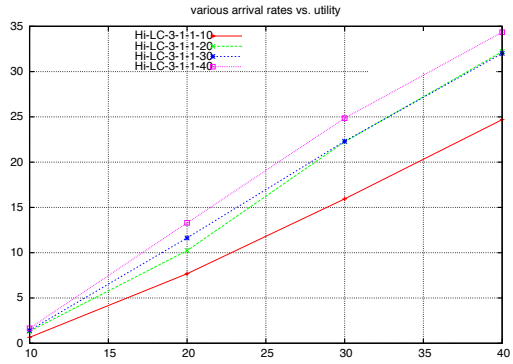


Figure 6: utility increase overtime for various arrival rates

$R(t)$, over the total number of relevant documents in the network R, i.e $eps_{s_i}(t) = \frac{R(t)}{R}$. Note that this ratio is not equal to recall ratio as the local searches may not be scheduled or finished by time t. Average effective propagation speed is defined as the average of the effective propagation speed of all the search sessions in the network. The maximal proportion of agents that a message visits in session s_i is $eps_{s_i}(t^*)$.

Global Utilization Ratio (GU) is defined as the average of the utilization rates of agents in the network. The utilization ratio of an agent is defined as the probability that the agent is busy and its queue is nonempty. A low global utilization ratio can be attributed to either a low query arrival rate or a poorly designed resource selection process and an agent search topology that creates unbalanced data flow in the network, leaving some agents with no queue entries while flooding others.

7.4 Results analysis and evaluation

Table 1 lists the experimental results of 6 different search strategies for 60 time units. These strategies explore the importance of using different topological agent organizations (powerlaw topology and hierarchical topology), search strategies (broadcast, two-phase hierarchical, and a balanced hierarchical search strategy). Fig. 5 depicts the global quality of the system as a function of the number of messages transmitted. *Hi-LC-5-15* is the curve for hierarchical, balanced two-phase search appraoch with local balancing techniques and *Powerlaw-LC-5-15* shows the performance for the broadcast search algorithm running on the powerlaw topology. In the following paragraphs, we analyze these experimental results and results shown in Fig. 6 that indicate how one specific search strategy is affected by the different arrival rates of the external queries.

Whether the load balance mechanism was used or not, the overall results were that the balanced hierarchical(details in Table 1) achieved highest overall utility while limiting the amount of communication relative to the broadcast approach. Additionally, the following findings were also revealed by the experimental results:

First, while the two-phase search algorithm was designed to take advantage of the hierarchical agent organizations, the way queries are forwarded in the system makes it easy to form hot spots that slow down the propagation of queries in the network. Therefore, from Table 1, the two-phase search algorithm produces low search quality. On the other hand, a balanced, hierarchical algorithm taking advantage of lateral links achieves the best search quality among the tested approaches in terms of both the number of communication messages and the overall utility. This is due to the fact that the queries are largely forwarded inside the relevant groups rather than being immediately forwarded to top level mediators in the two-phase search. This avoids the top-level mediators hot spots.

Secondly, a load balancing mechanism tends to improve system performance by reducing the number of messages when query arrival rate increases and congestion situations occur while increasing the number of messages in the network when the service load and utilization of the communication channel is low. From the experimental results, this load balancing mechanism improves the two-phase search algorithm the most. This can be attributed to the fact that the routing in two-phase search algorithm is very unbalanced with the query arrival rate used in these experiments. The preliminary experimental results also demon-

strate that the contribution of load balancing depends on the local computational capacity. Particularly, the traffic engineering techniques contribute to the performance more when local searching is not a bottleneck. When the local search queue size increases to a certain level, optimizing the traffic does not significantly improve search performance. It remains as future work to further discover the correlation between traffic flow and global search performance.

Thirdly, Fig. 6 shows that the overall system performance increases steadily with the increase of query arrival rates. This demonstrates that the system is a stable one.

Finally, the broadcast approach, to our surprise, performs quite well. It generates a lot of extraneous messages which are then filtered out locally by the language model based greedy local search scheduling algorithm. Therefore, this large number of messages did not distract the agents' local searches.

8. DISCUSSION AND FUTURE WORK

One of the directions that we would like to pursue is to provide differentiated service for search sessions by using selectively different search engines and parameters that trade off the likelihood of finding relevant documents to the time required for the search. We also want to be able to specify that some search sessions are more important than others, and expand the model so that the utility function of each search session may not necessarily be linear to the quality accrued over time. Rather, we assume that the utility functions increase faster at the beginning and the increase slows down when a certain amount of quality has been accrued.

$Q_{s_i}(t)$ is an objective measure which reflects the recall ratio at time t for search session s_i. Another measure, user utility, is also defined to capture the preferences that users have on the number of relevant documents returned at different periods during the search process.

$$U_{s_i}(t) = \mu(Q_{s_i}(t), t)$$

The overall system utility, $GU(t)$ is defined as the sum of the utility for each search session in a certain time period:

$$GU(t) = \sum_{s_i} \mu(Q_{s_i}(t), t)$$

By selecting an appropriate μ function, users can specify their preferences for search session s_i, i.e whether they prefer quick response but with a relative low recall ratio or vice versa. In this paper, we take $\mu(x) = x$ and therefore we consider that user satisfaction linearly increases as more relevant documents are returned.

9. CONCLUSIONS

In cooperative peer-to-peer information retrieval systems, each node can be considered as an intelligent agent and these agents work collectively to provide an information retrieval service. In order to effectively support multiple and concurrent search sessions in the network, we propose two traffic engineering techniques that minimize processing and communication bottlenecks. One is a novel agent control mechanism whose elements include resource selection, local search scheduling, and feedback-based load control and the other is a new two-phase query routing algorithm based on organizational knowledge. Experimental results show that this framework can reduce congestion situations, increase system throughput, and improve considerably the overall system utility.

10. REFERENCES

[1] J. Callan. *Distributed information retrieval*. Kluwer Academic Publishers, Reading, Massachusetts, 2000.

[2] J. C. French, A. L. Powell, J. P. Callan, C. L. Viles, T. Emmitt, K. J. Prey, and Y. Mou. Comparing the performance of database selection algorithms. In *Research and Development in Information Retrieval*, pages 238–245, 1999.

[3] R. Gallager. A minimum delay routing algorithm using distributed computation. In *IEEE transactions on communications*, pages 73–85, Jan 1977.

[4] D. Hawking, N. Craswell, and P. Thistlewaite. Overview of trec-6 very large collection track. In *In Proceedings of the Tenth Text Retrieval Conference TREC*, pages 93–105, 1997.

[5] B. Horling. *Modeling and Designing Multi-Agent Systems Through Explicit Organizational Design*. PhD thesis, University of Massachusetts at Amherst, Amherst, Massachusetts, 2005.

[6] R. Horling, Bryan; Mailler and V. Lesser. Farm: A scalable environment for multi-agent development and evaluation.

[7] J. Lu and J. Callan. Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In *In Proceedings of the Twenty-Seventh European Conference on Information Retrieval Research (ECIR'05)*.

[8] C. R. Palmer and J. G. Steffan. Generating network topologies that obey power laws. In *Proceedings of GLOBECOM '2000*, November 2000.

[9] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM Press, 1998.

[10] M. E. Renda and J. Callan. The robustness of content-based search in hierarchical peer to peer networks. In *CIKM '04: Proceedings of the thirteenth ACM conference on Information and knowledge management*, pages 562–570, New York, NY, USA, 2004. ACM Press.

[11] H. Zhang, W. B. Croft, B. Levine, and V. Lesser. A multi-agent approach for peer-to-peer information retrieval. In *Proceedings of Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*, July 2004.

[12] H. Zhang and V. R. Lesser. A dynamically formed hierarchical agent organization for a distributed content sharing system. In *2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2004), 20-24 September 2004, Beijing, China*, pages 169–175. IEEE Computer Society, 2004.

A Hierarchical Approach to Efficient Reinforcement Learning in Deterministic Domains

Carlos Diuk
Department of Computer Science
Rutgers University
Piscataway, NJ 08854-8019
USA
cdiuk@cs.rutgers.edu

Alexander L. Strehl
Department of Computer Science
Rutgers University
Piscataway, NJ 08854-8019
USA
strehl@cs.rutgers.edu

Michael L. Littman
Department of Computer Science
Rutgers University
Piscataway, NJ 08854-8019
USA
mlittman@cs.rutgers.edu

ABSTRACT

Factored representations, model-based learning, and hierarchies are well-studied techniques for improving the learning efficiency of reinforcement-learning algorithms in large-scale state spaces. We bring these three ideas together in a new algorithm. Our algorithm tackles two open problems from the reinforcement-learning literature, and provides a solution to those problems in deterministic domains. First, it shows how models can improve learning speed in the hierarchy-based MaxQ framework without disrupting opportunities for state abstraction. Second, we show how hierarchies can augment existing factored exploration algorithms to achieve not only low sample complexity for learning, but provably efficient planning as well. We illustrate the resulting performance gains in example domains. We prove polynomial bounds on the computational effort needed to attain near optimal performance within the hierarchy.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Artificial Intelligence-Learning

General Terms

Algorithms

Keywords

Reinforcement Learning, Hierarchical reinforcement learning, Factored representations, Sample complexity

1. INTRODUCTION

In the Markov decision process (MDP) formalization of the reinforcement-learning (RL) problem (Sutton and Barto 1998), a decision maker interacts with an MDP environment consisting of a finite state space S and action space A. Transitions are controlled by a Markov function where $P(s, a, s') = \Pr(s'|s, a)$ is the probability of reaching state $s' \in S$ after action $a \in A$ in state $s \in S$. The decision maker receives reward value $R(s, a)$ for action a in state s and attempts to maximize the expected cumulative reward.

In the current paper, we treat only MDPs with deterministic transitions, meaning that for each state-action pair there exists a unique s' such that $P(s, a, s') = 1$. We assume they also consist of only negative rewards ($R(s, a) < 0$) except for a set of final states $F \subseteq S$ that end the process when reached. We further assume that there are *short* solutions—policies that terminate after at most T steps for some polynomial size T. These conditions imply that the optimal value function is unique. Even though we present results only for the deterministic case, we stick with the standard notation of MDPs used in the reinforcement-learning community, which assumes stochastic domains.

A number of reinforcement-learning algorithms have been studied (Sutton and Barto 1998), some of which have been shown to find optimal policies under well-understood conditions. In analyzing algorithms, there are two main sources of complexity to be considered. The first, *sample complexity*, defines the amount of real-world experience needed by an algorithm to achieve near optimal results. The second, *computational complexity*, specifies the amount of computational work required per experience (Kakade 2003). We seek algorithms with low sample and computational complexity—both bounded by polynomials in critical parameters of the environment.

The best known reinforcement-learning algorithm is Q-learning. It is a sample-based learning rule with very low computational complexity, but it provably converges (Watkins and Dayan 1992) to the solution to the Bellman equations $Q^*(s, a) = R(s, a) + \gamma \sum_{s'} T(s, a, s') \max_{a'} Q^*(s', a')$, and therefore can be used to find optimal behavior. Unfortunately, Q-learning often exhibits very high sample complexity, making it practical principally in simulated domains in which experience is cheap to generate. To mitigate the high sample complexity of Q-learning, researchers have proposed algorithms that strike other tradeoffs between computational and sample complexity.

DYNA (Sutton 1990) and prioritized sweeping (Moore and Atkeson 1993) showed how learning transition models can empirically decrease sample complexity at increased computational complexity. Recent work (Kearns

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

and Singh 2002; Brafman and Tennenholtz 2002) has shown how model-based methods can provide formal polynomial-time bounds on both sample and computational complexity in MDPs.

Structure has long been recognized as important in computationally efficient sequential decision making (Boutilier *et al.* 1999). Dynamic Bayes Nets (DBNs) have emerged as a popular formalism for succinctly representing and solving large-scale MDPs (Koller and Parr 2000) and we adopt this factored-state framework in this paper.

In addition to structure in the state space, many researchers have studied algorithms that exploit hierarchical structure in the policy space (Kaelbling 1993; Hengst 2002). These methods have empirically provided relatively modest improvements in sample complexity over baseline RL approaches.

Kearns and Koller (1999) showed how model-based learning can be combined with factored states to provide polynomial sample-complexity bounds. An open problem from this work is achieving a polynomial computational-complexity bound. Dietterich (2000c) showed how factored states and hierarchy could be combined effectively, empirically improving sample complexity and reducing the number of parameters to be learned. Dietterich (2000b) also recognized the importance of combining models, factored states, and hierarchy, but found that the resulting learned hierarchical models no longer benefited from the factored representation.

The contribution of the current work is a novel combination of factored states, hierarchy and models, resulting in progress towards the solution of two important open problems. First, it shows how models can be combined with hierarchy without disrupting the benefits of factored states. The resulting algorithm exhibits greatly reduced sample complexity compared to model-free learning. Second, our combination of methods retains the polynomial sample complexity of existing model-based methods with factored states, with the hierarchy providing an approach to efficient planning in the learned models. Thus, we present the first factored-state reinforcement-learning algorithm with both polynomial sample *and* computational complexity. Section 6 illustrates our algorithm on a class of MDPs.

2. FACTORED-STATE MDPS

A factored-state MDP is one in which the state variables are factored into independently specified components. Let X be the set of state factors and, for all $x \in X$, $D(x)$ is the domain of values that factor x can take on. We write $v = \Phi_x(s)$ as the value of factor x in state s.

For a factored representation to provide a representational advantage over the standard tabular MDP representation, it is important that the transition probabilities and rewards support a structured representation. The assumption we adopt here, which generalizes the standard DBN representation, is that the probability that a factor x takes on a particular value v after a state transition in action a is a function of the *cluster* c of the state s.[1] We write $K(s,a,x)$ as the cluster for state s and factor x under action a and $\bar{P}^x(c,a,v)$ as the probability that a state from cluster c will transition to one that has factor x equal to v, under action a. Using this notation, for each action $a \in A$ and factor $x \in X$, transition probabilities are represented as $\Pr(s'|s,a) = \prod_{x \in X} \Pr(\Phi_x(s')|s,a) = \prod_{x \in X} \Pr(\Phi_x(s')|K(s,a,x),a) = \prod_{x \in X} \bar{P}^x(c_x,a,v)$ where $v = \Phi_x(s')$ and $c_x = K(s,a,x)$.

Similarly, we assume that reward functions are specified using the state clusters: $R(s,a) = \sum_{x \in X} \bar{R}(c_x,a)$ where $c_x = K(s,a,x)$.

2.1 Factored Rmax

Rmax is a reinforcement-learning algorithm introduced by Brafman and Tennenholtz (2002) and shown to have PAC sample complexity by Kakade (2003) (Brafman and Tennenholtz showed it was PAC in a slightly different setting). Factored Rmax is the direct generalization to factored MDPs (Guestrin *et al.* 2002). Factored Rmax is model based in that it maintains a model M' of the underlying factored MDP and, at each step, acts according to some optimal policy of its model. In this section, we'll describe the model used by Factored Rmax.

To motivate the model used by Factored Rmax, we'll describe at a high level the main intuition of the algorithm. Consider a fixed state factor x, action a, and cluster c ($c = K(s,a,x)$ for some state s). There exists an associated distribution $\bar{P}^x(c,a,\cdot)$ and reward $\bar{R}(c,a)$. The agent doesn't have access to these values, so they must be learned.

The trick behind Factored Rmax is to use the agent's experience only when there is enough of it to ensure decent accuracy, with high probability.

Let κ be some user-defined constant that is given to Rmax as input at the beginning of a run. In the determinstic setting $\kappa = 1$. For each distribution $\bar{P}^x(c,a,\cdot)$, Rmax maintains a count $\kappa_a(x,v,c)$ of the number of times it has taken action a from a state s for which $K(s,a,x) = c$ and $v = \Phi_x(s)$. As long as $\kappa_a(x,v,c) < \kappa$, Rmax assumes that any transition from s under a causes state value x to become e, an additional value added to each domain $D(x)$. Once a state value becomes e, a transition under any action cannot change it from being equal to e. Additionally, the reward for any state that has a state variable with value e is equal to $R_{\max}$, the maximum possible reward (zero in our case). On the first timestep such that $\kappa_a(x,v,c) = \kappa$, Rmax updates its model to use the empirical distribution as an approximation of $\bar{P}^x(c,a,\cdot)$, and the empirical reward for $\bar{R}(c,a)$.

3. THE MAXQ VALUE-FUNCTION DECOMPOSITION

One model of a task hierarchy in an MDP is to decompose the main task of maximizing reward en route to a terminal state into subtasks, each with its own terminal states and perhaps subtasks of its own.

Each task $1 \leq i \leq I$ in the hierarchy can be viewed as a self-contained MDP with final states F_i and action set A_i.[2] Actions $j \in A_i$ can be either the primitive actions of the MDP or subtasks $j > i$. The root task $i = 1$ uses $F_i = F$, the final states of the actual MDP.

We also define $T_i \in \mathbb{N}$, for each task i, such that for any given set of fixed polices for subtasks A_i, there exists some (hierarchical) policy for subtask i that terminates within T_i steps from any abstract non-terminal state.

[1] In DBNs, the cluster is determined by the joint assignment of the parents of x.

[2] We view primitive actions as tasks that don't have a set of final states but instead terminate after one timestep.

A *hierarchical policy* $\pi = \langle \pi_1, \ldots, \pi_I \rangle$ is a policy for each task i, $\pi_i : S \to A_i$. Policy π_i is considered locally optimal if it achieves maximum expected reward given subtask policies π_j for $j > i$.

If local optimality holds for all tasks, the corresponding hierarchical policy is called *recursively optimal*.

3.1 Recursive Solution

The state (V) and state-action (Q) forms of the Bellman equations are well known (Sutton and Barto 1998). Dietterich (2000a) implicitly proposes an alternative—the completion-function form:

$$C(s,a) = \sum_{s'} P(s,a,s') \max_{a'}(R(s',a') + C(s',a')).$$

Given a representation for the reward function, the completion function can be used to recover $Q(s,a) = R(s,a) + C(s,a)$, $V(s) = \max_a Q(s,a)$, and $\pi(s) = \text{argmax}_a Q(s,a)$.

Given ϵ-approximate transition and reward functions $\hat{P}$ and $\hat{R}$ and the task hierarchy, we can compute a hierarchically ϵ-optimal policy by defining a hierarchical completion function as follows. We consider the set of tasks i in reverse order from $i = I$ to $i = 1$. For each, we determine a completion function C^i. If i is the task for primitive action a, we define $C^i(s,a) = 0$.

For higher-level tasks i, we solve the MDP with actions A_i, states S, and final states F_i, where the transition function for action $j \in A_i$ is $P^j(s,s')$ and its reward function is $R^j(s)$. In other words, each subtask j is treated like an action that has a reward and a probabilistic transition to some state $s' \in F_j$. The MDP solution produces C^i. Specifically, for $s \in F_i$ we define $C^i(s,j) := 0$. Otherwise, we define the task completion function as follows:

$$C^i(s,j) = \sum_{s' \in F_j} P^j(s,s') \max_{j' \in A_i} (R^{j'}(s') + C^i(s',j')).$$

Computing $R^j(s)$ and $P^j(s,s')$ can be achieved by solving systems of linear equations: if $s \in F_j$, $P^j(s,s) = 1$ and $R^j(s) = 0$; otherwise,

$$\begin{aligned} P^j(s,s') &= \sum_{s_1} P^k(s,s_1) P^j(s_1,s') \quad \text{and} \\ R^j(s) &= R^k(s) + C^j(s,k) \end{aligned} \tag{1}$$

where $k = \pi^j(s)$.

In this construction, each task adopts an optimal policy given the subtasks, so $\pi = \langle \pi_1, \ldots, \pi_I \rangle$ is a recursively optimal policy for the MDP specified by $\hat{P}$ and $\hat{R}$.

3.2 Abstraction

Dietterich (2000c) described how to combine state abstraction with a hierarchical task decomposition. For each task i, define $Z_i \subseteq S$ to be the set of abstract-state representatives, and define an *abstraction function* $\phi_i : S \to Z_i$ mapping states to their abstract representatives. An abstract policy and an abstract completion function differ only for states with different representatives.

Similar to Dietterich (2000c), we say an abstraction is *valid* if the abstract completion function does not incur any approximation under any abstract policy. Define the abstract transition function $\mathbb{P}_i^j : S \times Z_i \to \mathbb{R}$ as $\mathbb{P}_i^j(s,z') = \sum_{s' \in S \text{ s.t. } \phi_i(s') = z'} P^j(s,s')$. We adopt the following concrete assumptions, which imply a valid abstraction:

1. For all $s \in S$, $z' \in Z_i$ and $j \in A_i$, $\mathbb{P}_i^j(s,z') = \mathbb{P}_i^j(\phi_i(s),z')$. That is, the total probability of a transition to states in an abstract class is dependent only on the abstract class of the current state.

2. For all $s, s' \in S$, $j, k \in A_i$, if $P_i^j(s,s') > 0$, then $R^k(s') = R^k(\phi_i(s'))$. That is, for any reachable next state, the reward function depends only on the abstract class.

3. For all $s, s' \in S$ and $k \in A_i$, if $R^k(s) \neq R^k(s')$ and $\phi_i(s) = \phi_i(s')$, then there is no sequence of actions in A_i that takes the agent from s to s'. That is, if two states in the same abstract class have different reward functions then they are not reachable from each other.

These assumptions imply a valid abstraction because the unique solution to the abstract completion function, defined by

$$\mathbb{C}^i(z,j) = \sum_{z' \in Z_i} \mathbb{P}_i^j(z,z') \max_{k \in A_i} (R^k(z') + \mathbb{C}^i(z',k)) \tag{2}$$

matches the completion function for all states $z \in Z_i$. This result follows from the uniqueness of the value function and algebraic substitution using the assumptions above.

The abstract completion function can be used to compute task rewards and policies as follows. First, we extend the abstract completion function to all states in the obvious way: $\mathbb{C}^i(s,j) = \mathbb{C}^i(\phi_i(s),j)$. Next, we can define the reward function recursively by $R^i(s) = \hat{R}(s,a)$ if i is subtask for primitive action a, $R^i(s) = \max_{j \in A_i}(R^j(s) + \mathbb{C}^i(s,j))$ otherwise. Finally, these quantities can be used to define a policy via $\pi_i(s) = \text{argmax}_{j \in A_i}(R^j(s) + \mathbb{C}^i(s,j))$. Note that these computations can be performed in polynomial time in the size of the task hierarchy as long as results are cached (each task should only compute its reward function once for a state).

3.3 An Example Domain

The main MDP environment we used for testing is called *bitflip*. This domain has several qualities that make it suitable for our tests, including an exponential and scalable state space, an exponential set of distinct Q values, linear-size set of completion values, and abstractions that can potentially reduce the size by an exponential factor. For the domain *bitflip*(n), meaning an instance of *bitflip* with n bits, the states are all binary n-bit numbers. There are n actions, with action flip(i) corresponding to "flip the ith bit". The actions don't always succeed. In particular, if action flip(i) is executed, the behavior depends on the bits to the left of i (j such that $j > i$). If any of these bits are set to 1, the action fails and all bits $j \geq i$ are set to 1. Otherwise, the ith bit is flipped. The reward function is deterministic; whenever action i is executed, a reward of -2^i is received. There is one final state, the state where all bits are equal to zero. Note that $V(s) = s$ where s is the state interpreted as an integer in binary. Thus, there are 2^n distinct V and Q values. The optimal policy is to flip the 1 bits, in order, from left to right.

We now describe the hierarchy for *bitflip*. For each i there is a subtask, clear(i), which terminates when the leftmost

i bits are all zero. The subtasks available to clear(i) are flip(i) and clear($i-1$). The subtask clear(n) is the root task. The abstraction function for flip(i) maps all states to four representative states. They correspond to the combinations of bit i being 0 or 1, and all bits to the left of i being 0 or at least one having a 1. This abstraction is valid and results in an abstract completion function with fewer than $8n$ distinct values.

4. DSHP: A DETERMINISTIC SAMPLE-BASED HIERARCHICAL PLANNER

Equation 2 provides an alternative formulation of the completion function based on abstraction that can result in a considerably smaller representation at each task. Solving for the abstract completion function $\mathbb{C}$ requires knowledge of the abstract transition function $\mathbb{P}$ and the reward function R for each of the abstract state representatives. Unfortunately, the analog of Equation 1, which defines MDP models of subtasks, can no longer be formulated in the abstract setting since each task may have its own abstract state space. In this section, we provide a method that overcomes this hurdle in deterministic domains.

Our goal is to develop a planning algorithm that takes as input a model M' of the true MDP M, and produces a hierarchical policy. We make no assumptions on the structure of M' or M, for example they can be factored or flat MDPs. We only require that given a state-action pair (s, a) query, our model M' will either return the correct next state, $T(s, a)$ and reward $R(s, a)$, or a special value $s_{\max}$, which indicates the model is unsure of the behavior resulting from action a in state s. We call such a state-action pair, which leads to $s_{\max}$ in the model M', an "unknown" state-action pair.

Let l be the height (length of the longest root to leaf path) of the MaxQ task hierarchy, and L_i be a list of the tasks on level i ($L_0 = \{1\}$). Let $U_j = \max_{i \in L_j} T_i$ be the maximum of T_i for tasks i on level j. Define $T = \prod_{j=0}^{l} U_j$. With no knowledge of the current MDP M and hierarchy H, we cannot expect any hierarchical policy to take less than T steps (primitive actions) to achieve its goal (whether that be to solve the task or to reach a certain part of the state space). In fact, we may need up to $O(T \cdot l)$ table lookups to execute any such policy for T steps.

At each step, the agent must ensure that it is either executing a recursively near-optimal policy, or it learning something new. Then the goal of the algorithm is to compute a hierarchical policy $\pi_{M'}$ that satisfies one of the following two conditions:

1. The policy $\pi_{M'}$ is recursively optimal in M.
2. By following policy $\pi_{M'}$ for T steps in M, the agent will reach an "unknown" state-action pair (a transition to $s_{\max}$ in M').

The algorithm solves the abstract MDP for the root task. Since doing so requires that the children of the root task have fixed policies, the algorithm first recursively calls itself for each of the root's children (thereby solving them and fixing optimal policies).

One problem with this approach is that the algorithm doesn't have access to the MDP M, but instead a model M'. Thus, for two states s, s' in the same cluster for task i (meaning $\phi_i(s) = \phi_i(s')$), it may be the case that execution of the same abstract action may take one of these states to $s_{\max}$ (in M') but not the other.

Our solution is to solve the MDP in a bottom-up fashion, solving higher level tasks after all lower level subtasks have been solved and to maintain a partially learned recursively optimal policy. In addition, we perform value updates on only those abstract state-action pairs for abstract states that are reachable from the current state. For each task i and for each reachable abstract state $z \in Z_i$, we keep track of the relevant primitive state s, such that $\phi_i(s) = z$, that will be reached if the current learned (partial) policy is performed (from the current state). To do so, we postpone the evaluation of abstract state $z' \in Z_i$ until a path from the current state is found to z', making it vital to know the value of z'. When z' is found, we have the precise primitive state (which abstracts to z') that would be reached on this path. If it is ever the case that a path p to $s_{\max}$ is found, the algorithm immediately halts and returns the policy that generates p. Otherwise, the entire MDP is solved and a recursively optimal policy is found.

Pseudocode for the algorithm is presented below. The core recursive procedure is `Solve`. The `RunTrajectory` routine executes an abstract action from a primitive state in the MDP M' and returns the resulting state and reward.

To simplify the exposition, we assume that every abstract state for subtask i is reachable from every other abstract state for subtask i in T_i steps. We also assume that terminal states of M cannot be reached by any hierarchical policy unless that hierarchical policy also terminates (meaning the root reaches a terminal abstract state). Elimination of these assumptions is straightforward.

4.1 Data Structures

Let H be the set of all tasks in the hierarchy, Z be the set of all abstract states, and $\mathcal{A}$ be the set of all abstract actions.

The model M' is a generative model and can be accessed via the two following functions:

- `NextState` : $S \times A \rightarrow S \cup \{s_{\max}\}$,
- `Reward` : $S \times A \rightarrow \mathbb{R} \cup \{\varnothing\}$.

The model of the hierarchy consists of:

- `Solved` : $H \rightarrow \{True, False\}$: Indicates whether we have solved the task or not. Initialized to $Solved(\cdot) = False$.
- `HAction` : $H \times Z \rightarrow \mathcal{A}$: represents the learned hierarchical policy by giving the next abstract action to take from the given task and abstract state.
- `HNextState` : $H \times Z \times \mathcal{A} \rightarrow Z$: Returns the next abstract state reached by executing the given abstract action in the given task and abstract state.
- `HReward` : $H \times Z \times \mathcal{A} \rightarrow \mathbb{R}$: Returns reward obtained by executing the given abstract action in the given task and abstract state.
- `HValue` : $H \times Z \rightarrow \mathbb{R}$: Returns the total value of the hierarchical policy starting in the given task from the given abstract state.

The input to the algorithm is the model MDP M', the hierarchy $\mathcal{H}$, and the current state s. The output of the algorithm will be a hierarchical policy $\pi_{M'}$.

4.2 Routines

The planning algorithm consists of the following four routines: `Main`, `Solve`, `RunTrajectory`, `HValueIteration`.

`Main`()
1: Set up data structures
2: Call `Solve`(`1`, `s`)
3: Output policy represented by `HAction`.

`Solve`(`i`, `s`)
`Input` : `i` $\in$ `H`, `s` $\in$ `S`
`Output` : `foundSmax` $\in \{$`True`, `False`$\}$
1: Let $\mathcal{Q}$ be the trajectory tree created with root $(s, \phi_i(s))$
2: `KNOWN` $= \{\}$
3: **for** each node N $= (s',\ z' := \phi_i(s'))$ in $\mathcal{Q}$ **do**
4: **for** each action $a_{ij} \in A_i$ **do**
5: **if** a_{ij} is not primitive and `Solved`(`a`$_{ij}$) $=$ `F` **then**
6: Call `foundSmax` $=$ `Solve`(`a`$_{ij}$, `s`$'$)
7: **if** `foundSmax` **then**
8: Set `HAction` to reach a_{ij} from root in $\mathcal{Q}$.
9: return `True` and `HALT`
10: **end if**
11: **end if**
12: (`s`$''$, `r`) $\leftarrow$ `RunTrajectory`(`a`$_{ij}$, `s`$'$)
13: **if** `s`$'' =$ `s`$_{\max}$ **then**
14: Goto line 8
15: **end if**
16: `HNextState`(`i`, `z`$'$, `a`$_{ij}$) $\leftarrow \phi_i$(`s`$''$)
17: `HReward`(`i`, `z`$'$, `a`$_{ij}$) $\leftarrow$ `r`
18: **if** ϕ_i(`s`$''$) $\notin$ `KNOWN` **then**
19: Add node (`s`$''$, ϕ_i(`s`$''$)) to $\mathcal{Q}$ as child to N
20: **end if**
21: **end for**
22: `KNOWN` $=$ `KNOWN` $\cup \{$`z`$'\}$
23: **end for**
24: Call `HValueIteration`(`i`)
25: `Solved`(`i`) $\leftarrow$ `True`
26: Return `False` and `HALT`

`RunTrajectory`(`i`, `s`)
`Input` : `i` $\in$ `H`, `s` $\in$ `S`
`Output` : (`s`$'$, `r`) $\in$ `S` $\times \mathbb{R}$
1: **if** Task i is a leaf task **then**
2: `s`$' \leftarrow$ `NextState`(`s`, `HAction`(`i`, ϕ_i(`s`)))
3: `r` $\leftarrow$ `Reward`(`s`, `HAction`(`i`))
4: return (s', r) and `HALT`
5: **else**
6: `R`$_{total}$ $\leftarrow$ `0`
7: (`s`$_{temp}$, `r`$_{temp}$) $\leftarrow$ `RunTrajectory`(`HAction`(`i`, ϕ_i(`s`)), `s`)
8: `R`$_{total}$ $\leftarrow$ `R`$_{total}$ $+$ `r`$_{temp}$
9: **if** `s`$_{temp} \in$ `F`$_i \cup \{$`s`$_{\max}\}$ (`s`$_{temp}$ is a terminal state) **then**
10: return (`s`$_{temp}$, `R`$_{total}$) and `HALT`
11: **else**
12: `s` $\leftarrow$ `s`$_{temp}$
13: goto line 7
14: **end if**
15: **end if**

`HValueIteration`(`i`)
`Input` : `i` $\in$ `H`
`Output` : updated values

This algorithm simply runs value iteration on the abstract state space Z_i. It uses the functions `HNextState`(`i`, $\cdot$, $\cdot$) and `HReward`(`i`, $\cdot$, $\cdot$), which have already been learned. It then learns the function `HValue`(`i`, $\cdot$) and uses it to compute some optimal policy, which it stores in `HAction`(`i`, $\cdot$).

4.3 Correctness of Algorithm

The `Solve` routine when called with a leaf task (one with only primitive actions), will clearly make no recursive calls. In this case, for each abstract state and action, `RunTrajectory` is used to learn the transition and reward function. If, during any call to `RunTrajectory`, a path to $s_{\max}$ is found the function ends and returns a policy leading to $s_{\max}$. Otherwise, value iteration is used to completely solve the abstract MDP.

If `Solve`(`i`, `s`) is called with a task i that has non-primitive actions (actions that are other tasks), then the correctness follows from a similar argument to the one above. The only difference is that, for each non-primitive action (from each abstract state), a recursive call to `Solve` is made. However, as the tasks called recursively exist below (as the children of) i, we can inductively assume these calls either return `F`, and have learned optimal policies for the subtask, or they return `T` and have learned a policy to reach $s_{\max}$ from s.

4.4 Analysis of DSHP

Our claim is that the runtime of DSHP is polynomial in the relevant input quantities of the reinforcement-learning problem. Specifically, let $VI(i)$ be the time it takes to run value iteration for task i, then DSHP runs in time $O(\sum_{i=1}^{I} (|Z_i| \cdot |A_i| \cdot T + VI(i)))$. To see this result, note that:

- Besides the initial call of `Solve` within `Main`, the only other calls to `Solve` are recursive.
- A call to `Solve`(`i`, $\cdot$) will only make recursive calls `Solve`(`j`, $\cdot$) where task j is below task i in the MaxQ task hierarchy.
- When `Solve`(`i`, $\cdot$) completes (without returning $True$, which halts the entire algorithm), we have that `Solved`(`i`) $=$ `True`.
- Thus, for any task i, `Solve`(`i`, $\cdot$) will never be called more than once during the entire algorithm. Therefore, the number of calls to `Solve` cannot be greater than I, the number of tasks.
- A single call to `Solve`(`i`, $\cdot$) has two main iterations, the outer **for** loop of Line 3 and the inner **for** loop of Line 4. Due to the **if** statement of Line 18, the outer **for** loop will iterate at most $|Z_i|$ times, while the inner **for** loop clearly iterates A_i times. The computation that takes place within the inner **for** loop, not counting the computation within other calls to `Solve` (which we have already counted), is a constant plus the time it takes to call `RunTrajectory`, which happens to be $O(T)$. After the two loops terminate, `Solve`(`i`, $\cdot$) calls `HValueIteration`, which takes time $VI(i)$. Thus, the total complexity of entire algorithm

is $O(\sum_{i=1}^{I}(|Z_i| \cdot |A_i| \cdot T + VI(i)))$. We can make this bound tighter by noting that `RunTrajectory(i,·)` takes less time as the depth in the tree of i increases.

5. USING DSHP TO LEARN AN MDP

Given a solution to the planning problem of Section 4, the learning algorithm is simple: Use experience to generate M', then plan. If DSHP outputs a hierarchical-optimal policy we're done. Otherwise, run the policy until an "unknown" (s, a) is found and repeat. It's clear that the learning algorithm, which we call the **Deterministic Sample-Based Hierarchical Learner (DSHL)** will have sample complexity bounded by $O(T \cdot U(M))$, with $U(M)$ being the number of "unknown" state-actions that can be found, until the model M' is identical to M. For a flat and deterministic MDP M, $U(M) = |S| \cdot |A|$. In factored representations, $U(M)$ is often much smaller. By separating the hierarchy from the underlying representation, DSHL reaps the benefits of both forms of abstraction simultaneously.

6. EMPIRICAL RESULTS

We tested MaxQ, Factored Rmax and DSHL in the *bitflip* and the taxi domain ((Dietterich 2000c)). The results show that, as expected, Rmax has poor computational complexity and MaxQ has poor sample complexity. DSHL achieves both low computational and sample complexity.

6.1 Bitflip domain

Our testing methodology was to evaluate the agent's learned policy after each episode on a number of example states of the environment. The evaluation states were picked to follow a standard structure based on the number of bits n: all 1s (11..11), single 1 in the center (0..010..0), first half 1s (1..10..0), second half 1s (0..01..1), by quarters (1.10.01.10.0) and alternating 1s (101010...). Once an optimal policy is found, the trial stops and we record the cost (number of action choices) and computer time for the trial. Each experiment is repeated for 20 trials and all the results are averaged. We tested each algorithm on *bitflip*(n), where we allowed n to increase until one of two termination conditions were satisfied: when a value of n is reached for which the algorithm took longer than thirty minutes on a single trial, or when the number of action choices reached thirty thousand.

Each algorithm has a number of parameters. For Factored Rmax, the main parameter is the Rmax constant κ. As we are dealing with a deterministic domain, this parameter was set to 1. For MaxQ, one must specify the value of ϵ for exploration and its decay rate. We did a search of these parameters (from 0.1 to 0.3, on 0.05 increments) and chose the setting that gave the best performance for each problem size. DSHL requires no parameter tuning.

As we can see from the Figure 1, the sample complexity of both Factored Rmax and DSHL grows linearly in the number of factors of the domain, whereas the sample complexity of MaxQ grows exponentially. On the other hand, the computational cost for Factored Rmax grows exponentially (Figure 2), while both MaxQ and DSHL show polynomial growth rates. So, DSHL is the only algorithm to show polynomial growth rates in both sample and computational complexities, as predicted by our analyses.

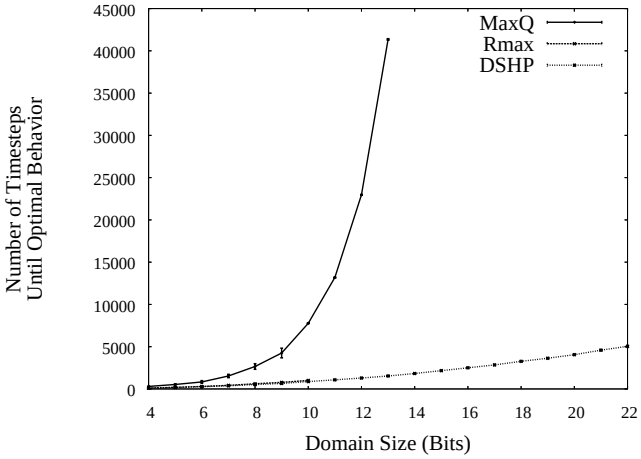


Figure 1: Number of steps per problem size until algorithm reaches optimal policy for the given start states. Rmax can only solve problems upto size 10, given the amount of time allocated.

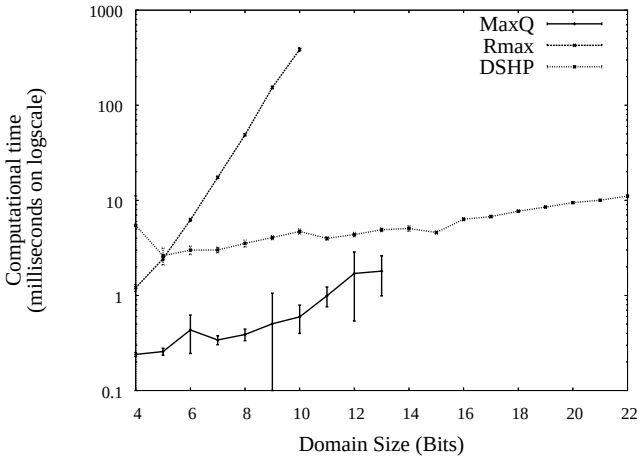


Figure 2: Computational time per step, per problem size. Plotted on a log scale, due to huge disparity between algorithms.

6.2 Taxi domain

We used a similar testing methodology for the taxi domain described by Dietterich (2000c). We used the standard problem (no fuel, no fickle passenger) on a $5x5$ grid. We used the same hierarchical structure and the same kind of abstractions used by MaxQ. The size of the problem in this case is fixed, so we only expose the good performance of DSHL in the two types of complexities.

As in the previous problem, we evaluate the agent's learned policy after each episode on a set of six starting combinations of <taxi (x,y) location, passenger location, passenger destination>. The start states used were: $\{(2,2), Y, R\}$, $\{(2,2), Y, G\}$, $\{(2,2), Y, B\}$, $\{(2,2), R, B\}$, $\{(0,4), Y, R\}$, $\{(0,3), B, G\}$.

The results are shown in the following table:

	Number of steps	Time per step
MaxQ	6298	9.570ms
Factored Rmax	1839	97.780ms
DSHL	319	16.876ms

As can be seen, DSHL significantly improves the sample complexity of both MaxQ and Factored Rmax by combining models, factored state spaces and a hierarchal structure, while attaining low computational complexity comparable to that of MaxQ.

7. CONCLUSION

We have designed a novel algorithm that combines factored representations, model-based learning, and hierarchies to provide a formal guarantee on learning time in many large deterministic reinforcement-learning domains. Empirical results indicate that, for a simple extensible class of MDPs, existing algorithms fail to provide either efficient sample complexity or efficient computational complexity. The results demonstrate, on the other hand, how the theoretical assurances of our hybrid algorithm translate to practical advantages.

8. REFERENCES

[Boutilier *et al.*, 1999] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[Brafman and Tennenholtz, 2002] Ronen I. Brafman and Moshe Tennenholtz. R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

[Dietterich, 2000a] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[Dietterich, 2000b] Thomas G. Dietterich. An overview of MAXQ hierarchical reinforcement learning. In *Proceedings of the Symposium on Abstraction, Reformulation and Approximation SARA 2000*, pages 26–44, 2000.

[Dietterich, 2000c] Thomas G. Dietterich. State abstraction in MAXQ hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems 12*, pages 994–1000, 2000.

[Guestrin *et al.*, 2002] Carlos Guestrin, Relu Patrascu, and Dale Schuurmans. Algorithm-directed exploration for model-based reinforcement learning in factored MDPs. In *Proceedings of the International Conference on Machine Learning*, pages 235–242, 2002.

[Hengst, 2002] B. Hengst. Discovering hierarchy in reinforcement learning with HEXQ. In *Maching Learning: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 243–250. Morgan Kaufmann, 2002.

[Kaelbling, 1993] Leslie Pack Kaelbling. Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the Tenth International Conference on Machine Learning*, Amherst, MA, 1993. Morgan Kaufmann.

[Kakade, 2003] Sham M. Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

[Kearns and Koller, 1999] Michael J. Kearns and Daphne Koller. Efficient reinforcement learning in factored MDPs. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 740–747, 1999.

[Kearns and Singh, 2002] Michael J. Kearns and Satinder P. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2–3):209–232, 2002.

[Koller and Parr, 2000] Daphne Koller and Ronald Parr. Policy iteration for factored MDPs. In *Uncertainty in Artificial Intelligence, Proceedings of the Sixteenth Conference (UAI 2000)*, pages 326–334, 2000.

[Moore and Atkeson, 1993] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130, 1993.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[Sutton, 1990] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224, Austin, TX, 1990. Morgan Kaufmann.

[Watkins and Dayan, 1992] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

A Smart Home Agent for Plan Recognition

Bruno Bouchard
Université de Sherbrooke
2500, boul. de l'Université,
Sherbrooke (Québec), Canada
Bruno.Bouchard@usherbrooke.ca

Abdenour Bouzouane
Université du Québec Chicoutimi
555, boul. de l'Université
Chicoutimi (Québec), Canada
abouzoua@uqac.ca

Sylvain Giroux
Université de Sherbrooke
2500, boul. de l'Université,
Sherbrooke (Québec), Canada
Sylvain.Giroux@usherbrooke.ca

ABSTRACT

We propose a formal framework to keyhole plan recognition based on lattice theory and action description logic. This framework minimizes the uncertainty about the prediction of the observed agent's behaviour by dynamically generating new implicit intentions. This approach offers an effective solution to actual recognition of ADL in a smart home, in order to provide assistance to persons suffering from cognitive deficiencies.

Keywords

Description logic, lattice theory, plan recognition, smart home

1. INTRODUCTION

Recent developments in information technology and increasing problems in the health field, including population aging and medical staff shortages, have opened the way to a whole set of new and promising research avenues, most notably, work on smart homes. A growing literature [2][5][9][10] has explored the process by which cognitive assistance, inside a smart home, is provided to occupants suffering from cognitive deficiencies such as Alzheimer's disease and schizophrenia, for the performance of their Activities of Daily Living (ADL). One of the major difficulties inherent to cognitive assistance is to recognize and predict the inhabitant ADL from their basic observed actions and the events produced by these actions. This problem is referred to as plan recognition in the field of artificial intelligence [4]. In the context of cognitive assistance, these predictions are used to identify the various ways a smart home (observer agent) may help its occupants (patients). An important assumption underlying the plan recognition is that the observed agent is rational, i.e., all of his carried actions and intentions are coherent. However, for patients suffering from cognitive deficiencies, rationality might indeed be a strong assumption. The purpose of this paper is then to initiate the development of a generic approach to plan recognition for a smart home, that could be applied to people suffering from cognitive impairments. Hence, we deal with keyhole plan recognition which is, strictly speaking, more general than intended plan recognition, in that it makes no assumption about efforts to cooperate on the part of the observed agent [12]. The literature related to keyhole plan recognition [1][4][8][10], in particular the logical approaches [7][12], share a significant limitation in that they do not take into account intra-dependencies between the possible plans in the recognition process. These intra-dependencies result from the fact that, even if possible plans might seem connected, the intentions concerning two distinct observations are not necessarily related. Our proposal relies on lattice theory and action description logic [3] to define algebraic tools taking into account this problem. These tools also allow us to formalize the inferential process of plan recognition in a model of reasoning by classification through a lattice structure. This interpretation model defines a recognition space. This space will not only serve to characterize the uncertainty in the prediction of a future action. It will also serve to determine the appropriate time when the assisting agent could be brought in to increase his autonomy in order to execute an assistance action in the habitat by taking over the ADL patient control.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00

2. RECOGNITION SPACE MODEL

A plan recognition process consists, for an observer agent, of interpreting the set of the observed actions, performed by a human or another observed agent in action, with the aim of predicting his future actions and thus the plausible plans that explain his behaviour. Let $A = \{a, b, \ldots\}$ be the set of actions that an observed agent is able to perform and let $P = \{\alpha, \beta, \ldots\}$ be the set of known plans of the observer. Let O be the set of observations such that $O = \{o \mid \exists a \in A \rightarrow a(o)\}$. The assertion $a(o)$ means that observation o corresponds to an a-type action. The set of possible plans that would explain the set of observations O, according to the agent's knowledge, is expressed by $P_s^o = \{\alpha \in P \mid \exists (a,o) \in \alpha \times O \rightarrow a(o)\}$. Starting with this set of possible plans, we can deduce that the agent will at least perform one of them. However, its intentions can go beyond the set of possible plans. In order to generate all agent's intentions, we enhance P_s^o by dynamically generating extra plans (new intentions) based on the composition operation $\alpha \oplus \beta$ between each pair of incomparable possible plans $(\alpha, \beta) \in P_s^o$. We define the set of plausible intentions P_l^o as the union of the composition pairs of possible plans. This set is an interpretation model for O if it forms a lattice structure $< P_l^o, \prec_p, \Delta, \nabla >$ ordered by the subsumption relation of plans $\prec_p$ and each couple of plans admits an upper bound ∇, corresponding to their least common partial subsumer, which is minimally composed of the observed actions. Also, each couple of plans admits a lower bound Δ, corresponding to a variable plan obtained by disunifying the incomparable possible plans using the first-order logic disunification operation [6]. The interest on computing this lower bound is to find a new intention schema. It is defined as a plan that can contain action variables serving to characterize the uncertainty in the prediction of a future action. A plan

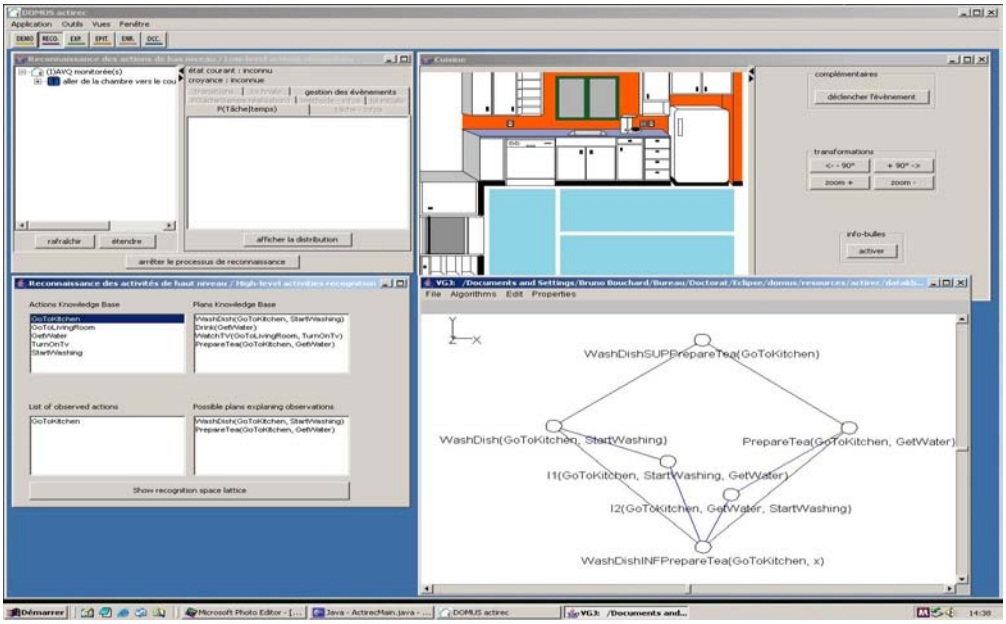


Figure 1. DOMUS application of activities recognition.

$\alpha\Delta\beta$ is defined as a sequence of actions $a_1,\ldots,x,\ldots,a_n$, denoted $\alpha\Delta\beta(a_n \circ a_{n-1} \circ \cdots \circ x \circ \cdots \circ a_1)$, where $\circ$ is a sequence operator and x is an action variable. $\alpha\Delta\beta$ is a variable plan if and only if there exists a substitution $\sigma(x) \in A^+$ such that each new intention $\iota(a_n \circ \cdots \circ \sigma(x) \circ \cdots \circ a_1)$ satisfy the two following consistency properties. The first one is the $\oplus$-stability of ι which means that each intention $\iota \in \alpha \oplus \beta$ is formed by: (i) a set of partial plans included in the knowledge base P of the observer, (ii) at least one action common to plan α and to plan β, and (iii) a composition of actions that are component of α or component of β. There is no possibility of introducing other external actions. The second criteria is the $\oplus$-closure propriety which expresses that each intention ι must admit an upper bound $\alpha\nabla\beta$ and a lower bound $\alpha\Delta\beta$. Hence, the new intentions must be included in the interval $[\alpha\Delta\beta, \alpha\nabla\beta]$. One can ask question regarding the computational complexity of this composition operation. The answer is that the combination of the incomparable possible plans is not done blindly. First, we only consider the consistent possible plans, which satisfy the stability and closure criteria (first filter). Second, the possible plans that we consider are those which are in the lattice structure bounded by the upper and lower bounds (second filter). Finally, for each pair of incomparable possible plans, we combine them by using the disunification operation (third filter). These filters allow us to reduce and control the computational complexity of the composition operation. Therefore, the recognition process can be seen as the generation of all consistent plans, resulting from the substitution of their action variables, which are ordered by the subsumption relation of plans and include between the infinimum and the supremum of the recognition space. Of course, it is assumed that all the observed actions are related. Consequently, we build a lattice structure starting from the first observation that will be refined when new observations will be detected. This refinement will be computed by extracting a sub-lattice (a new refined recognition space) from the initial lattice structure, and so on.

3. RECOGNITION OF ACTIVITIES IN A SMART HOME

The DOMUS[1] lab consists of a standard apartment with a kitchen, living room, dining room, bedroom, and bathroom that are equipped with sensors, smart tags (RFID), location and identification systems for objects and people, audio and video devices, etc. As we can see on the right of the Figure 1, we developed a graphical tool (here showing the kitchen) that allows us to simulate the activation of the various environment sensors by clicking on the corresponding graphical objects, where most of devices are monitored and controlled through a Crestron-based infrastructure. Basic events are generated by sensors as shown on the left of this figure, and are directly sent to low-level activity recognition (LAR) agent which owns a virtual representation of the habitat's environment encoded in a description logic knowledge base described with PowerLoom [11]. When new inputs are received from hardware sensors, the LAR agent updates the state of the habitat and creates an action structure, representing the changes that happened to the environment, according to our model of action described in [3]. Thereafter, the LAR agent notifies the High-Level Recognition (HLR) agent (cognitive assistant) that a new low-level action is detected and it sends the action's type. To be able to plan a future assistance task, the HLR agent must initially understand patient's intentions by building a minimal interpretation model describing his plausible intentions at this specific moment. This model takes the form of a lattice built following a cooking scenario, as shown in the bottom right of Figure 1. The recognition space is composed of all the plans that can be classified between the lattice infinimum and supremum, such as $\{$*WashDish*, *PrepareTea*, *WashDish*∇*PrepareTea*, *WashDish*Δ*PrepareTea*, I_1, $I_2\}$ according to the observed action o_1 detected by LAR agent, in this case, only one action

[1]The DOMUS lab is sponsored by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Canadian Foundation for Innovation (CFI).

of the *GoToKitchen* type, such as $O = \{GoToKitchen(o_1)\}$. The lattice infinimum corresponds to $(x \circ GoToKitchen)$, which enables us to generate, by the substitution process of the action variable x and the disunification-composition operations on the incomparable possible plans, two new consistent intentions $I_1(GoToKitchen \circ StartWashing \circ GetWater)$ and $I_2(GoToKitchen \circ GetWater \circ StartWashing)$ that did not exist beforehand in *P*, as shown in the bottom right of Figure 1. In this context, the smart home would be authorized to perform an action of assistance, like reminding the occupant of the procedure to achieve I_1 in the event of a memory lapse (i.e. Alzheimer's disease).

4. RELATED WORKS

Several approaches have been followed to explore solutions to keyhole plan recognition, such as the probabilistic approach of [1][4], the learning approach of [8][10] and the logical approach of [7][12]. The probabilistic methods, primarily based on the Bayesian networks [1] and on the Dempster-Shafer theory [4], use a set of probabilistic rules enabling to update the probability attributed to each plausible hypothesis following an observation. The conclusion drawn from the recognition process by the system is simply the hypothesis having the highest probability. The weakness of the probabilistic approaches stems from the heuristic methods, dependent on the context, used to compute the probability of each competing hypothesis [4]. The learning approaches of [8][10] seek to identify patterns from the observed actions in order to build a probabilistic predictive model of the observed agent behaviour. The main limitation of this kind of approaches is due to the fact that the generalization learned rule might lead to inconsistent behaviour and to a very large amount of training data. The logical approaches of Kautz [7] and Wobke [12] are closer to our work. In these two theories, the observer starts with a collection of plan schemas, forming an interconnected abstraction/decomposition hierarchy expressing the plan library of the observer agent. This hierarchy is represented as axioms in first-order logic and it is supposed to be complete, i.e., it contains the set of all the actions that may be observed. Kautz proposes a set of hypotheses (exhaustiveness, disjointedness, component/use, minimum cardinality), based on McCarty's circumscription theory, for the hierarchy of actions. These hypotheses take the form of inference rules that enable to extract a minimal covering model of interpretation from the hierarchy, for the observed actions. The weakness of Kautz's approach comes from the fact that all plans inferred as possible through the covering model are considered equiprobable. Wobke [12] has proposed a solution to this limitation using situation theory. His proposal, based on Kautz's work, consists of defining a partial order relation between plans and organize the elements of the hierarchy that are defined in terms of situations, by level of plausibility. This classification enables him to draw more refined conclusions than those of Kautz by retaining only the most plausible class among the possible plans. A significant limitation of Wobke's work is created by the situation semantic (a particular case of possible worlds semantic) for which the formal apparatus is complex to make operational in a real context. Finally, these previously explored approaches supposed that the observed actions are induced by the same intention, even if those may be carried out for different purposes.

5. CONCLUSION

This paper has proposed an non-analytical approach, based on lattice theory and action description logic, for re-examining the main issues surrounding the problem of formalizing plan recognition. This combined approach provides a viable solution to plan recognition problems by minimizing uncertainty about the prediction of the observed agent's behaviour. This is achieved by dynamically generating implicit intentions resulting from intra-dependencies existing between possible plans. It should be emphasized that this initial framework is not meant to bring exhaustive answers to the issues raised by the multiple problems related to plan recognition. However, it can be considered as a first step towards developing a complete formal plan recognition theory, based on the classification paradigm. It should bring effective solutions to concrete problems such as the recognition of ADL in a smart home and the problem of completing the plans library, which indeed cannot be complete in any domain.

6. REFERENCES

[1] Albrecht D.W., Zukerman I., Nicholson A.: Bayesian Models for Keyhole Plan Recognition in an Adventure Game, *User Modelling and User-Adapted Interaction*, (1998), (8) 5-47.

[2] Bauchet J., Mayers A.: Modelisation of ADL in its Environment for Cognitive Assistance, In *Proc. of the 3rd International Conference on Smart homes and health Telematic (ICOST'05)*, Sherbrooke, Canada, (2005), 3-10.

[3] Bouzouane A.: Towards an Authority Sharing Based on The Description Logic of Action Model, In *Proceeding of the Fifth IEEE International Conference on Intelligent Agent Technology (IEEE-IAT'05)*, September 19-22, Compiègne, France, (2005), 277-280.

[4] Carberry S.: Techniques for Plan Recognition, *User Modeling and User Adapted-Interaction*, (2001), (11) 31-48.

[5] Fatiha L., Lefebvre B.: A cognitive system for a smart home dedicated to people in loss of autonomy, In *Proc. of the 3rd International Conference on Smart homes and health Telematic, (ICOST'05)*, Sherbrooke, Canada, (2005), 245-254.

[6] Huet G., Oppen D.C.: Equations and Rewrite Rules: A Survey, In *Formal Language Theory: Perspectives and Open Problems*, Academic Press, (1980), 349-405.

[7] Kautz H.: A Formal Theory of Plan Recognition and its Implementation, In *Reasoning About Plans*, Allen J., Pelavin R. and Tenenberg J. eds., Morgan Kaufmann, San Mateo, C.A., (1991), 69-125.

[8] Nicolescu M.N., Mataric M.J.: Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *Proc. of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'03)*, Melbourne, Australia, July 2003.

[9] Pigot H., Mayers A., Giroux S.: The intelligent habitat and everyday life activity support, In *Proc. of the 5th International Conference on Simulations in Biomedicine (ICSB'03)*, April 2-4, Slovenia, (2003), 507-516.

[10] Philipose M., Fishkin K., Perkowitz M., Patterson D., Kautz H., Hahnel D.: Inferring activities from interactions with objects, *IEEE Pervasive Computing Magazine: Mobile and Ubiquitous Systems*, 3(4):50-57, October-December 2004.

[11] PowerLoom: *PowerLoom Knowledge Representation System*, ISI, University of Southern California, http://www.isi.edu/isd/LOOM/PowerLoom/index.html, (2002).

[12] Wobke W.: Two Logical Theories of Plan Recognition, *Journal of Logic Computation*, Vol. 12 (3), (2002), 371-412.

Exploring Sensor Networks using Mobile Agents

Daniel Massaguer[†], Chien-Liang Fok[‡], Nalini Venkatasubramanian[†], Gruia-Catalin Roman[‡], and Chenyang Lu[‡]

[†]Donald Bren School of Information and Computer Science
University of California, Irvine
Irvine, CA 92697, USA
{dmassagu, nalini}@uci.edu

[‡]Department of Computer Science and Engineering
Washington University in Saint Louis
Saint Louis, MO, 63130, USA
{liang, roman, lu}@cse.wustl.edu

ABSTRACT

Wireless sensor networks are often difficult to program and unable to adapt to a changing environment. Mobile agent middleware promises to address both concerns by providing higher-level programming abstractions and the ability to inject new agents into a preexisting network. The unique characteristics of wireless sensor networks like resource scarcity and emphasis on spatial locality require new algorithms for controlling agent behavior. This paper presents a procedure for one specific behavior: network exploration. Network exploration is needed by many tasks ranging from simple data collection to network health monitoring. Our proposed procedure uses a genetic algorithm to determine the number of agents and their itineraries, followed by techniques for in-network adaptation to unpredictable situations like node failure. This paper presents a genetic algorithm and its adaptation strategies. The procedure is evaluated using a wireless sensor network consisting of 25 Mica2 motes running Agilla, a mobile agent middleware for wireless sensor networks.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence. Distributed Artificial Intelligence[Intelligent agents, Multiagent systems]

General Terms

Design, Algorithms, Experimentation

Keywords

Applications of autonomous agents and multi-agent systems, mobile agents, (multi-)agent planning, performance evaluation of agent systems

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

Wireless sensor networks (WSNs) have attracted tremendous interest in recent years as embedded systems, microsensors, and batteries improve. A WSN consists of numerous sensors (motes) embedded within the environment that autonomously form wireless ad hoc networks. By increasing the density and scale of traditional sensing systems, a WSN enables users to gain unprecedented levels of context-awareness. Current applications for WSNs include structural and habitat monitoring [6, 5] and fire fighting [2], etc.

Due to the nature of WSNs (nodes are faulty and resource-constraint) developers are confronted with a bewildering set of conflicting requirements, mostly related to software flexibility and efficiency. For example, many WSNs are deployed for long periods of time virtually guaranteeing that the user requirements will change. Unfortunately, most WSNs run statically installed software that is loaded prior to deployment, limiting network adaptivity to the tweaking of predefined parameters. Another conflicting property stems from the fact that WSNs are expected to cover a wide geographic area servicing many users that need to run different programs.

Mobile agents offer a solution to the problems mentioned above. Unlike traditional programs, mobile agents are capable of migrating across the network carrying code and state. As they migrate, they perform application-specific tasks like taking sensor readings, performing in-network data aggregation, and coordinating with other agents to achieve a common goal. This enables the development of fluid applications capable of adapting to the intricacies of WSNs (see an example of a fire tracking scenario in [2]). Mobile agents also allow WSNs to be dynamically reprogrammed by enabling users to inject new agents into the network and allowing old ones to die. Furthermore, they allow multiple applications to co-exist since multiple agents may reside on each node (for an example, see [4]).

Before mobile agents can be effectively used in a WSN, new algorithms must be created for controlling their behavior. One topic of particular interest is *network exploration*. The goal of network exploration is to visit every node and to do so efficiently in terms of time and energy. This paper addresses the specific issue of sensor network exploration using multiple mobile agents. Network exploration is used in many applications including data collection, network diagnostic and health monitoring, residual energy scanning, topology discovery, and global reprogramming. Network exploration requires that every region of the sensor network be visited at least once in a manner that is efficient in terms

of energy consumption and latency. The order and number of visitations does not matter, though redundant visitations in general decrease efficiency and should be limited. The goal of the network exploration procedure is to determine the number of agents and each of their itineraries, and the techniques for dealing with unexpected in-network circumstances, e.g., changes in topology.

2. PROBLEM DEFINITION

Using multiple mobile agents to explore a WSN offers the benefits of flexibility, load distribution, and asynchronous operation. However, since WSNs are large and resource-constrained, these mobile agents must be designed to minimize energy utilization. Furthermore, application requirements dictate that real-time guarantees be met. This section introduces the necessary notation, formulates a minimization problem, and presents a solution.

2.1 WSN Model and Assumptions

A WSN consists of numerous nodes, each containing a set of sensors, microcontroller, external memory, radio transceiver, and power source. Nodes can sense real-world phenomena, perform local computations, store information, receive data, and transmit data. Receiving and transmitting data costs more energy than performing local computations. Moreover, we assume that the WSN is connected to the rest of the world via a single base station called the *collection point*. Finally, we assume that the amount of time an agent spends processing data at each node and the time it takes to migrate is predictable. In practice, such information may be estimated based on empirical measurements (see section 4), where the measured *time per node* includes the processing, migration, and sleeping time. This knowledge is necessary to ensure that the network is explored within a certain time.

2.2 Problem Formulation

Given a sensor network, Ω, with a single collection point, the goal is to explore the entire network by visiting every region within bounded time while minimizing energy consumption. This can be done using multiple mobile agents operating autonomously and in parallel following unique routes. These routes must start and end at the collection point.

Many applications do not need every node to be visited, especially if the nodes are physically close. For example, it may not be necessary to visit two nodes separated by just one meter if the task is to determine the average temperature along a path. To account for this, we introduce a density variable d which specifies how many of the nodes within the network need to be visited for the network to be considered explored. Specifically, a network is considered explored if all nodes $m \in \Omega$ have either been visited, or have a neighbor within d hops that has been visited. Note that if $d = 0$, the exploration problem reduces to one where every node needs to be visited.

The time constraint t is specified by the maximum length of a path, expressed as the number of nodes. We do not use an actual time since the amount of time an agent spends on each node, and the amount of time it takes an agent to migrate, is predictable.

The cost of an agent, a, visiting a node, m, following a certain path, p, is denoted $C_a(m,p)$. It is defined in terms of the weight of the agent, $W_a(m,p)$, and the amount of energy remaining on the node, E_m, as follows:

$$C_a(m,p) = \frac{W_a(m,p)}{E_m} \tag{1}$$

Note that as the amount of energy available on a node increases, the cost decreases and vice-versa. This is necessary to balance the residual energy of the nodes and thereby extend the lifetime of the network.

The weight of agent a, $W_a(m,p)$, is a function that returns the weight, in bytes, of agent a, when visiting node m, following path p. The weight of an agent depends on the size of its code and the amount of data that it carries. The size of its code is fixed, whereas the size of the data varies every hop and depends on what the application does upon visiting each node.

Finally, let P be the set of all paths and P_m be the set of paths that include node m, where $P_m \subseteq P$, we can formulate the following minimization problem:

$$\texttt{min} \left(\sum_{m \in \Omega} \sum_{p \in P_m} \frac{W_a(m,p)}{E_m} \right) \tag{2}$$

such that $\forall p \in P : |p| \leq t$ and the WSN is covered with density d.

3. A GENETIC ALGORITHM SOLUTION

Given the problem's complexity and the size of the search space($O(2^{2^{3t}})$), a heuristic is necessary. In addition, the sensor network's topology may change due to nodes running out of energy, desynchronizing, or being affected by external forces (e.g., being moved by the wind). Continuously monitoring the network topology is energy consuming. Thus, in order to solve the formulated minimization problem defined by equation 2, we propose a procedure containing two phases: an initial route planning phase done by a genetic algorithm and an in-network route adaptation phase that is done by each agent autonomously after it has been injected into the network.

Figure 1 contains the pseudo-code of the genetic algorithm (GA). It takes as input the definition of a sensor network Ω (i.e., the topology), the time constraint t (i.e., the maximum number of nodes of the longest route), the density d, and the specification of the varying size of the agents $W_a(m,p)$. The outcome is the number of agents and their routes, such that the collection is achieved in an energy-aware manner and within a time deadline. The GA can be divided into the following subparts: *Generation of initial population*, *Crossing*, *Mutation*, *Selection*, and *Adding of new individuals*.

```
01: GeneticAlgorithm(Ω,d,t,Wa(m,p))
02:    CreateInitialPopulation(N);
03:     For generation=1 to MaxGenerations
04:        Cross();
05:        Mutate();
06:        NaturalSelection();
07:        AddNewIndividuals(I);
08:     endFor
09:    return AgentRoutes
10: endGeneticAlgorithm
```

Figure 1: The Genetic Algorithm

Generation of initial population and the addition

of new individuals: An initial set, P, of solutions is generated. These solutions are a collection of random paths that cover the network, and are generated by repeatedly calculating paths until the whole network, Ω, has been covered with a density of d. Each path is selected by starting at the collection point and iteratively selecting a random neighbor until the collection point is reached again. Only those neighboring nodes whose minimum distances (in number of hops) to the collection point are not greater than the remaining time will be selected. Since the time constraint is expressed as the maximum number of nodes per path, comparing a nodes distance to the collection point with the deadline is trivial. All feasible neighbors are equally likely to be selected, and the initial size of P, $|P|$, is a design parameter.

Crossing and Mutation: At every iteration, a new generation is added to P by selecting x individuals to be crossed. Two individuals are crossed by taking a portion of each individual's math and combining them to form a new individual. Depending of how those individuals are selected, and which portion of their paths are crossed, new more efficient paths may be formed. Mutation is done by crossing an individual with itself.

Evaluation and Selection: All solutions are evaluated by the function being minimized in 2. Upon evaluation, $k_n\%$ of the worst individuals are randomly selected and removed from the population. Note that to avoid loosing the best solutions generated, the five best solutions are never removed, irrespective of k_n.

3.1 In-Network Route Adaptation

Since the topology of WSNs changes, mobile agents must adapt their routes during runtime. In particular, if an agent cannot successfully migrate from the ith node of its route, to the $(i+1)^{th}$, it incrementally tries to migrate to the jth node such that $j > (i+1)$. Given a dense WSN, it is likely that at least one of the nodes that follow the $(i+1)^{th}$ node on the agent itinerary will be in the ith node's radio range, which allows the agent to resume its exploration. Furthermore, if the underlying mobile agent middleware supports multihop routing (e.g., geographic routing) such as Agilla [3], this solution is even more robust since an agent can migrate to any node in the WSN regardless of distance.

4. EVALUATION

To evaluate our system, we deployed it on an a wireless sensor network consisting of 25 Mica2 motes [1]running Agilla [3], a mobile agent middleware. Having a real implementation allowed us to define realistic $W_a(m,p)$ functions and empirically determine a translation from the deadline expressed in seconds to the maximum number of hops per path.

We measured the actual amount of time agent migration takes. We ran three different experiments with $t = 13$, 22, and 26. The GA generated twenty-five initial solutions, executed for five iterations, and added five new solutions every iteration thereafter. For each experiment, we averaged the three runs and plotted the percent of the network explored versus time. The results are shown in Figure 2.

As expected, Figure 2 shows that the genetic algorithm produces faster results with less variance when the maximum path length is more restrictive (shorter). With a time constraint of $t = 13$, 22, and 26, mobile agents spent an average of 14s, 22.33s, and 24s in the WSN, respectively. The

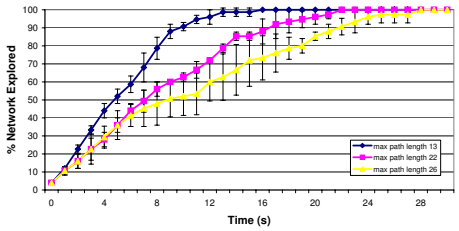


Figure 2: Percent Network Explored vs. Time

average time it takes to move an agent one hop is $1.233s$.

5. CONCLUSIONS

Software for WSNs needs to be more flexible. Mobile agents can provide this flexibility, but new algorithms are needed to control agent behavior. This paper addresses network exploration defined as visiting enough nodes in the network to achieve a certain level of coverage. We present a genetic algorithm that determines the number of agents necessary to explore the network and their paths. It includes contingency plans to account for the dynamic nature of WSNs. We show through experiments on a real WSN that our procedure is effective and that it is able to explore the network within the set deadline, while keeping energy consumption in check.

6. ACKNOWLEDGMENT

This research has been supported in part by the Generalitat de Catalunya, the Univ. of Girona, and the Univ. of California through a Girona fellowship, and by the US Office of Naval Research under MURI research contract N00014-02-1-0715.

7. REFERENCES

[1] Crossbow Technology, Inc. *MPR/MIB User's Manual*, 2004.

[2] C.-L. Fok, G.-C. Roman, and C. Lu. Mobile agent middleware for sensor networks: An application case study. In *Proc. of the 4th Int. Conf. on Information Processing in Sensor Networks (IPSN'05)*, pages 382–387. IEEE, April 2005.

[3] C.-L. Fok, G.-C. Roman, and C. Lu. Rapid development and flexible deployment of adaptive wireless sensor network applications. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'05)*, pages 653–662. IEEE, June 2005.

[4] G. Hackmann, C.-L. Fok, G.-C. Roman, C. Lu, C. Zuver, K. English, and J. Meier. Agile cargo tracking using mobile agents in wireless sensor networks. In *Proc. of the ACM SenSys*, page 303, 2005.

[5] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proc. of the 1st ACM Workshop on Wireless Sensor Networks and Applications*, pages 88–97, September 2002.

[6] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *Proc. of the ACM SenSys*, pages 13–24, 2004.

Stochastic Planning for Weakly-Coupled Distributed Agents

AnYuan Guo Victor Lesser
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
{anyuan,lesser}@cs.umass.edu

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems*

General Terms

Algorithms, Performance, Design, Experimentation

Keywords

planning, game theory

1. INTRODUCTION

Partially observable stochastic games (POSGs) provide a powerful framework for modeling multi-agent interactions. While elegant and expressive, this framework has been shown to be computationally intractable [1]. An exact dynamic programming algorithm for POSGs has been developed recently, but due to high computational demands, it has only been demonstrated to work on extremely small problems. Several approximate approaches have been developed [3, 5], but they lack strong theoretical guarantees. In light of these theoretical and practical limitations, there is a need to identify special classes of POSGs that can be solved tractably.

One dimension along which computational gain can be leveraged is by exploiting the independence present in the problem dynamics. In this paper, we examine a class of POSGs where the agents only interact loosely by restricting one another's available actions. Specifically, rather than having fixed action sets, each agent's action set is a function of the global state. The agents are independent from one another otherwise, i.e. they have separate transition and reward functions that do not interact. This class of problems arises frequently in practice. Many real world domains are inhabited by self-interested agents that act to achieve their individual goals. They may not affect each other in any way, except for occasionally putting restrictions on what each other can do.

The best-known solution concepts in game theory are that of computing Nash equilibrium and performing strategy elimination. Our work addresses both of these solution concepts. First, we show that finding a Nash equilibrium where each agent achieves reward at least k is NP-hard. Then, we present the MAAE algorithm, an exact algorithm that performs iterated elimination of dominated strategies. It is able to tackle much larger problems than algorithms for the general class by exploiting the independence among agents.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan
Copyright 2006 ACM 1-59593-303-4/06/000 ...$5.00.

2. FORMAL MODEL

In this section, we will present a formal definition of our model. An n-agent POSG with state-dependent action sets can be defined as $\langle\{S_i\},\{s_i^0\},\{A_i\},\{B_i\},\{P_i\},\{R_i\}\rangle$, where,

- S_i is the state space of agent i. $S = S_1 \times S_2 \times \ldots \times S_n$ denotes the joint state set.
- $s_i^0 \in \Delta(S_i)$ is the initial state distribution of agent i.
- A_i is the action set of agent i. $A = A_1 \times A_2 \times \ldots \times A_n$ denotes the joint action set.
- $B_i : S \rightarrow 2^{A_i}$ is the available action function that maps a joint state to the set of available actions for agent i. $B_i(s) \subset A_i$ for all $s \in S_i$.
- $P_i : S_i \times A_i \times S_i \rightarrow [0,1]$ is the transition function of agent i. The joint transition function is completely factored, where $P((s'_1 \ldots s'_n) \mid (s_1 \ldots s_n),(a_1 \ldots a_n)) = \prod_{i=1}^{n} P_i(s'_i|s_i,a_i)$.
- $R_i : S_i \times A_i \rightarrow \Re$ is the reward function for agent i. This states that the rewards of the agents depend only on the local state and the action taken by agent i.

DEFINITION 1. *A* local policy π_i *of agent i is a mapping from local states and available action sets $\langle s_i, B_i(s)\rangle$ to actions in the available action set $B_i(s)$. The joint state s is a tuple of local states, where s_i denotes the local state of agent i. A* joint policy, $\pi = \langle\pi_1 \ldots \pi_n\rangle$, *is defined to be a tuple of local policies, one for each agent.*

There are several key differences between our model and the general POSG model. First, the power of our model comes from the *available action function*, B, which maps joint states to available actions for each agent. Secondly, our

model does not have an observation function. Each agent has a complete view of its own local state. Partial observability arises as a result of an agent's limited view of the the states of *other* agents but not of its own. Finally, the local policy of each agent is no longer a function of the local state only but a function of the available action set as well.

Here is an example of a POSG with state-dependent action sets. Two autonomous rovers are exploring an unknown terrain and collecting rock samples. There is a river with a narrow bridge on it that only allows one rover to pass at a time. To simplify the illustration, we will assume the rovers have two states {*on land, on bridge*}, and three actions each {*move, get on bridge, pick up rock*}. The rovers receive reward for the number of rocks picked up. The state-dependent action set is used to constrain the rover's actions such that only one is allowed on the bridge at a time. For example, the available actions for rover 1 are specified as follows, $B_1(\langle s_1 = \textit{on land}, s_2 = \textit{on land}\rangle)$ = {*move, get on bridge, pick up rock*}, $B_1(\langle s_1 = \textit{on land}, s_2 = \textit{on bridge}\rangle)$ = {*move, pick up rock*}, and analogously for rover 2.

3. COMPLEXITY

We state our decision problem, denoted as POSG-NE, as follows: given a POSG with state-dependent action sets, $G = \langle \{S_i\}, \{s_i^0\}, \{A_i\}, \{B_i\}, \{P_i\}, \{R_i\}\rangle$, does there exist a Nash equilibrium where all agents have expected utility at least k?

THEOREM 1. *POSG-NE is NP-hard even in problems involving 2 agents and a horizon of 2.*

Proof: We will present a reduction from NFG-NE, a known NP-hard problem [2]. This is the problem of determining the existence of a Nash equilibrium where all agents have expected utility at least k in a 2-agent normal form game, $N = \langle \Sigma_1, \Sigma_2, U_1, U_2\rangle$.

The basis for the reduction rests on the observation that normal form games can be thought of as degenerate POSGs in which each agent only has one state and is allowed to take only one action from that state. When encoding the utility function of the normal form game using constructs in our problem, the challenge lies in the fact that the utility is specified as a function of the actions of *both* agents, whereas, in our model, both the transition and the reward function only have access to the *local* state and action.

To solve this problem, we add auxiliary states to memorize the action taken at each state. We also add composite actions to the action set that corresponds to all possible pairs of actions by each agent. The state-dependent action set restricts the legal actions from the auxiliary states to be the single composite action that corresponds to the pair of actions taken by the agents. The reward for taking the composite action simply corresponds to the utility of the pair of component primitive actions in the normal form game. Therefore, there exists a Nash equilibrium where all agents have expected utility at least k in the POSG if and only if the same is true in the normal form game. □

4. THE MAAE ALGORITHM

We present an algorithm that performs iterated elimination of dominated strategies. The algorithm is able to work directly with the compact representation of a POSG without first converting it to the double exponentially large normal form representation. In this algorithm, first, we first fix the action sets of each agent at each state to the optimistic and pessimistic action sets. The idea behind this is that although an agent cannot predict exactly what actions will be available at each state, it can find out what actions would be available in the best and worst case scenario. In the best case, none of the actions that can be restricted by the state of the other agents are, and in the worst case, all of the actions that can be restricted are in fact unavailable.

For each agent i:

1. For each state s_k, compute the optimistic and pessimistic action sets. Here, $D = \{s \mid s^i = s_k^i\}$.

$$A_{opt}(s_k^i) = \bigcup_{s \in D} B_i(s)$$

$$A_{pes}(s_k^i) = \bigcap_{s \in D} B_i(s)$$

2. Compute the value functions of the 2 MDPs that correspond to alternately fixing the available action sets to the A_{opt} and A_{pes}.

$$U(s) = \max_{a \in A_{opt}(s)} \left[R(s,a) + \sum_{s'} P(s'|s,a) U(s') \right]$$

$$L(s) = \max_{a \in A_{pes}(s)} \left[R(s,a) + \sum_{s'} P(s'|s,a) L(s') \right]$$

3. For each action in each state, derive upper and lower bounds on the action value from the value function bounds $U(s)$ and $L(s)$.

$$Q_U(s,a) = R(s,a) + \sum_{s'} P(s'|s,a) U(s)$$

$$Q_L(s,a) = R(s,a) + \sum_{s'} P(s'|s,a) L(s)$$

4. At each state, eliminate all actions a' whose action value is dominated by another action a, $Q_L(s,a) \geq Q_U(s,a')$.

5. Repeat steps 1 to 4 until no more actions can be eliminated at any state.

Table 1: The iterated action elimination algorithm.

Once we fix the available action sets of each agent at all the states, the agents no longer depend on each other in any way. Our model decomposes to a set of MDPs. For each agent, we solve two MDPs, one using the optimistic action sets and the other using the pessimistic action sets. The solutions will provide us with an upper and a lower bound on the actual value function of each agent. With these upper and lower bounds on the expected values of each state, we can now derive bounds on the expected action values. At each state, dominated actions are pruned. We iterate the action elimination procedure until no more actions can be pruned. Since the number of actions at each state is finite, the procedure will eventually converge.

THEOREM 2. *The iterated action elimination algorithm corresponds to the iterated elimination of very weakly dominated strategies.*

5. EXPERIMENTS

We tested the algorithm on two domains involving machine maintenance and autonomous rover exploration. In manufacturing settings, factories frequently have to make maintenance and repair decisions. The question is how to weigh these two decisions at different stages of the machine life-cycle so as to minimize the likelihood of a machine malfunction.

The detailed specifications for the problem are as follows. There are ten factories, or agents, each owns and operates a machine. Routine maintenance is performed on site. When the machine becomes very old, they need to be serviced by the machine's manufacturer who could only take on one job at a time. This resource constraint is modeled by our state dependent action sets. The state space of each agent is characterized by the current life-cycle of its machine {*new, worn, old, broken*}. Each year, the factory examines the current state of the machine and makes one of the following decisions: *do nothing*, *perform maintenance* or *make repairs*.

Furthermore, each factory has its own reward function. A sample reward function is as follows. There is a high cost, $100,000 associated with a catastrophe (machine breaking while in operation). The factory makes a profit of $5,000 each year when the machine is in operation. Machine maintenance cost $1,000 while repairs cost $10,000. The goal is to maximize the factory's profit.

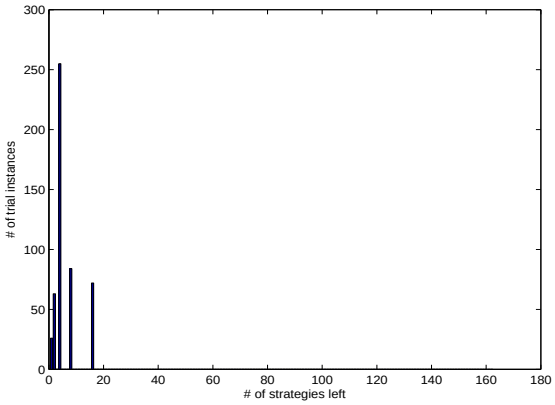


Figure 1: Machine maintenance problem: the distribution of the size of the policy set after the MAAE algorithm is run.

We performed 500 trials by randomly choosing the reward functions of each of the agents. The maintenance costs are chosen in the ranges of $1,000 to $10,000; the cost of a catastrophe ranges from $50,000 to $100,000; finally, the cost of repairs ranges from $10,000 to $50,000. The qualitative trends we found actually correspond well to our intuition. When the machine is relatively new, most of the policies would recommend to do nothing. Unless the penalty for a machine breaking is very high, in which case it adopts a more conservative approach of performing maintenance.

As the machine ages, the algorithm tries to weight the cost of maintenance and repairs. Overall, the MAAE algorithm drastically reduces the size of the policy space. The number of policies before pruning is 162 per agent, while the the number of policies after the algorithm is run, averaged over all trials, is 6. The distribution of the number of policies over all trial runs can be seen in Figure 1 below.

Aside from the large policy space reduction, another key advantage of this algorithm is that it scales well with the number of agents. There are ten agents in this experiment while the general POSG algorithm, due to its high computational demands, has so far only been shown to be effective on 2 agent problems [4]. Next, we tested the MAAE algorithm on a problem with much a larger state space. We examine how the size of the state space and the level of interaction affect the efficiency of the algorithm.

# states	before pruning	after pruning
6	972,000	13
8	34,992,000	168
9	2.1×10^8	69
12	4.5×10^{10}	774
16	5.9×10^{13}	2,198

Table 2: Average size of the policy space before and after action elimination for problems with 3 constrained states.

The second test domain is a simplified 2-agent autonomous rover exploration scenario. The size of each agent's local state space varies from 6 to 16. We introduced up to three constrained states in each agent's state space. 100 trials were run for each size of state space and number of constraints. Table 2 shows detailed results for problems with three constrained states. For problems with one and two constrained states, the final policy space ranges from 3 to 106 and 8 to 252 respectively. In all three cases, the iterated action elimination algorithm is able to reduce the size of the policy space by several orders of magnitude.

6. REFERENCES

[1] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27:4, November 2002.

[2] V. Conitzer and T. Sandholm. Complexity results about Nash equilibria. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

[3] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proceedings of the Third Joint Conference on Autonomous Agents and Multiagent Systems*, 2004.

[4] E. Hansen and D. Bernstein. Dynamic programming for partially observable stochastic games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 2004.

[5] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

Dynamic Cooperation Using Resource Based Planning

Utku Erdoğdu
Department of Computer Engineering
Midde East Technical University
Ankara, Turkey
utku@ceng.metu.edu.tr

Faruk Polat
Department of Computer Engineering
Midde East Technical University
Ankara, Turkey
polat@ceng.metu.edu.tr

ABSTRACT

In multi-agent planning, cooperation is a crucial interaction to reduce plan costs and complexities in order to satisfy non-conflicting goals. However, it is not generally natural to sacrifice from the autonomy of agents by letting agents share all of their local plans. Resource based planning is a planning framework similar to logic based planning based on resources and skills that map to states and actions of world state. Resource based planning is very functional in processing plan elements to find cooperation schemas. In this work we extended the simple resource based plan coordination method proposed by Tonino,et al [2]. Our framework provides not only a basis for modifying plan elements, but also functionality for manipulating plans dynamically. A negotiation algorithm using a resource based perspective provides the necessary ground for forming cooperation among agents. We illustrated the effectiveness of the proposed method on a modified postman domain, where agents try to minimize the cost of delivering the letters they are assigned.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*plan execution, formation, and generation*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Theory,Design

Keywords

Applications of Autonomous Agents and Multi-agent Systems, Multi-agent Planning, Cooperation and coordination Among Agents

1. INTRODUCTION

Planning is one of the classical, but popular research areas in Artificial Intelligence. If we consider a simple autonomous agent, we desire it to have its own judgement and decision mechanism by the definition of autonomy. Thus a simple autonomous agent should construct her own plan regardless of the framework used or the environment it acts in. Classical AI planners are designed for this sole purpose. However in multi-agent domains individual decisions are rarely enough for producing optimal plans for satisfying all the goals. The global optimum is nearly impossible with pure autonomy, so is individual optimum. Therefore, agents need to cooperate to generate the best multi-agent plan through sharing tentative solutions, exchanging subgoals, or having other agents to satisfy their goals.

The simplest approach to generate a multiagent plan is generating a single plan for all agents. The plan could be generated by a central authority. This perspective might be applicable for agent colonies where a vast number of agents have a single goal, but for autonomous agents with different, possibly conflicting goals, this approach is inadequate.

If the autonomy of the agents is vital for the application, a central planning unit is inappropriate regardless of the success of the plans it produces. It would be more effective if agents generate their plans independently and interact with each other during the execution to revise their plans in case of contingencies, or new goals.

Tonino et al. developed a resource based multi-agent planning method[2] similar to traditional planning. However the representation schema used allows sub-goals and plan elements to be processed as resources that can be exchanged, removed from and inserted into plans.

In this work, we extend the resource based planning framework to handle dynamic goals and complex interaction requirements that exist in many real-life problems. We applied the proposed approach to the modified postman problem where agents (postmen) are required to deliver a predetermined set of letters to some destinations. The domain is dynamic in that, besides the initial letters, agents are required to collect and deliver letters at specified locations in real-time. In order to keep their individual plan costs or total global cost nearly optimal, the agents should cooperate upon new goals by undertaking/exchanging subgoals. We propose a new cooperation mechanism independent of domain properties to handle real-time goals.

2. RESOURCE BASED PLAN FRAMEWORK

The resource based plan framework uses resources and skills instead of states and operators used by traditional planners. A resource is a real or abstract object used or produced by actions of agents. A skill is simply an action of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

an agent. A skill instance in a plan represents an application of a skill. Application of a skill consumes some resources and produces new resources. Consumed and produced resources might be identical.

Using these two concepts a plan can be represented by a bipartite directed graph. Resources are represented by resource nodes and skills are represented by skill nodes. Initial world state and the goal state are represented as two sets of resources. Initial resources are consumed by a plan in order to produce goal resources. Figure 1 shows an example plan graph where circles denote the resource nodes, squares denote the skill nodes and directed arcs denote consumption or production of a resource by a skill.

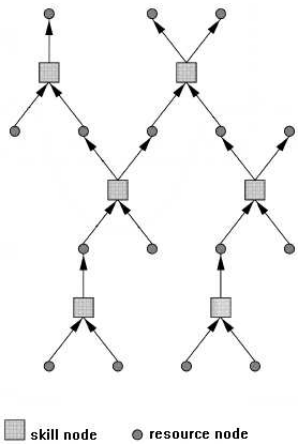


Figure 1: An example single agent Resource Based Plan Graph.

The graph representation is used for determining possibilities of cooperation between agents. Note that each skill produces resources. Some of these resources are goal resources, some are consumed by other skills and some are just side effects of the plan. We can safely remove a skill from a plan graph if we can find a way to generate the same resources that this skill produces.Tonino discusses a plan revision schema based on this principle[2].

They provide a simple protocol for compensation. This framework proves to be successful in establishing cooperation and generating a collaborative plan where total and individual costs of the plan are reduced as much as possible. The framework is applied to a transportation problem where trucks carry goods between cities. In the collaborative plans produced, trucks share routes and goods they carry, thus costs are reduced to minimum for the simple route topologies. However there are some issues that framework is not sufficient.

Although the framework is independent of the problem domain, it is not effective for domains having large state space. The selected application domain (transportation example) possesses a dense and small graph and hence agents produce many side effect resources which can be used by other agents. In most domains, producing extra resources may be cheap, but not trivial. A mechanism to enforce agents producing side effect resources is necessary.

Secondly, the framework mainly transfers the side effect resources of one agent to other relevant agents. There is also a need to finely tune this resource exchange. The simplest form of a collaboration is to transfer a useless resource from the acceptor to the proposer. There are domains in which no useless resource exists and thus it is not possible to simply transfer the resources, which increases the plan cost. The collaboration should allow the exchange of resources whose cost can be assessed by each of the agents participating in the exchange.

Our framework is designed to provide solution to these problems and improve the original framework in technical and theoretical levels. The details of our framework are given in the following sections.

3. DYNAMIC RESOURCE BASED COOPERATION FRAMEWORK

The postman problem is one of the classical example domains for multi-agent research[1, 3]. We changed the classical postman problem as follows. First, agents are required to collect letters from some locations and deliver them to the desired locations. A letter is represented with a source node (possibly the post office) and a destination node. Moreover, the final destination of agents is changed from post office to any node in the graph. It is shown in [1] that if the final destination node is post office, i.e., delivery of all letters is a cycle, then there may be trivial interaction schemas independent of the letter set. By changing the final destination to be any node, we prevent such trivial interactions.

When a new goal is assigned to an agent, possible choices are

- The goal can be already satisfied by the current plan,
- Another agent can satisfy the goal and
- The agent can modify her plan to satisfy the goal.

It is apparent that a rational agent selects one of the feasible alternatives in the order given above. If the new goal is already satisfied by the current plan without additional cost, then the plan is slightly revised to satisfy this new goal. Otherwise the agent computes the cost of modifying the current plan to satisfy the new goal. If the modification is costly, then the agent interacts with others to see whether she can benefit from any cooperation, or not. The cooperation alternative which does not bring additional cost is selected and agents involved in the cooperation revise their plans accordingly.

When an agent asks for another agent to handle one of her goals, the second agent attempts to handle the whole goal alone if it does not bring an extra cost. If this request cannot be satisfied, the proposer agent breaks the goal into subgoals and repeats the same process. In our framework, we have realized only one level decomposition but the this can further be extended to any level easily.

The acceptor agent could simply accept to satisfy the request, deny the request or accept to satisfy only some of the subgoals of the request. Moreover, the agent may satisfy the request because of two reasons; her plan might already satisfy the request with trivial modification or the requested goal might be inserted into her plan with modification which does not increase agent's cost. This structural decomposition of goals in decision making leads to a more interesting and fine tuned cooperation mechanism as we will illustrate in examples.

The acceptor might accept the request with the condition that the proposer handles one or some of the goals of the acceptor. To do this, agents could adopt their plans to partition the total goal set into meaningful subsets. In our system, an acceptor might request from a proposer to deliver one of her letters that contributes most to the cost of her plan.

In the previous section, we mentioned that there is need for a mechanism that is capable of producing side effect resources. We make use of a credibility metric in order to have this effect. Agents have credibility on each other. In case the proposer requests a new task from the acceptor, the acceptor will accept to satisfy this request if the cost of the modification in the acceptors' plan is less than or equal to the credibility of the proposer on the acceptor.

These are the salient features added into the original resource based framework [2]. The following sections describe and exemplify the proposed framework taking into consideration the domain properties.

4. APPLICATION OF DYNAMIC RESOURCE BASED PLANNING TO POSTMAN DOMAIN

Our framework starts with an initial planning phase where each agent constructs a plan for collecting and delivering the letters assigned to her. These plans are constructed independently and hence they do not take into consideration potential cooperations.

The initial plan computation algorithm has exponential complexity in terms of number of letters. The calculation of all pair shortest path and constructing the final path are all polynomial. However, an exhaustive search is necessary to find a minimum hamiltonian path on the constructed graph, which takes exponential time in terms of the number of nodes in the temporal ordering graph which is roughly equal to two times the number of letters.

In the plan graph each skill has a cost. Delivery and collection skills have unit cost, while cost of going from one node to another is naturally the weight of the edge between two nodes. In our framework we used four resources and five skills. Resources are:

- **Connected-Resource** that represents connectivity of graph nodes.
- **Location-Resource** that represents location of an agent.
- **Have-Letter-Resource** that represents having a letter.
- **Delivered-Letter** that represents having delivered a letter.

Skills are:

- **Go-Skill** that represents going from one node to another.
- **Collect-Skill** that represents collecting a letter.
- **Deliver-Skill** that represents delivering a letter.
- **Exchange-From-Skill** that represents getting a letter at rendezvous.
- **Exchange-To-Skill** that represents giving a letter at rendezvous.

Agents have initial credibility on each other. This credibility corresponds to utility concept as we discussed previously. We used a simple scalar credibility measure and set it to an empirical value. If an agent satisfies the request of another agent, she lowers other agent's credibility. The reduction amount is simply the cost of the request for the acceptor agent. The credibility value is chosen such that a given number of trivial deliveries could be handled.

In our work we used a negotiation protocol where agents propose bids for requests. The bids may also include counter proposals and counter requests. By negotiating, agents try to undertake requests for tasks if it is profitable for them. Since negotiation involves counter requests and the agent's utility is based on credibility measure, there are several possible outcomes of the negotiation, some of them increasing cost of a party slightly. For example if the whole task cannot be accepted by any other agent, the agent seeks alternatives to decompose the goal into subgoals .

If the agents agree on cooperation through exchanging letters, a rendezvous should be arranged to transfer the letter. We used an explicit rendezvous concept to build parts of the plan graph to denote that the agents meet at an agreed location and exchange the letter. The negotiation phase includes the arrangement of this rendezvous.

5. CONCLUSION AND FUTURE WORK

We attempted to extend the resource based planning framework of Tonino et.al. Our primary motivation was to extend this framework to allow contingency planning especially for realistic real life problems. The most important feature of this work is to use negotiation to revise plans to handle new goals. Refining the plans in real-time is vital if agents are faced with new goals in real-time. The framework is also designed to be robust, thus more complex interaction schemas can be introduced. For example in an auction based domain, agents might prefer to participate in auctions rather than negotiation. Different interaction mechanisms would result in interesting variations of the framework.

The proposed framework uses a novel representation and modeling schemas for effectively capturing and using distinguished features of more realistic domains. We used a simple utility measure to realize credibility of agents which could be generalized further. This simple measure is sufficiently functional for our example domain. If the agent interactions are more complex, each agent should have better models of others.

6. REFERENCES

[1] J. S. Rosenschein and G. Zlotkin. Designing conventions for automated negotiation. *AI Mag.*, 15(3):29–46, 1994.

[2] J. Tonino, A. Bos, M. M. de Weerdt, and C. Witteveen. Plan coordination by revision in collective agent-based systems. *Artificial Intelligence*, 142(2):121–145, 2002.

[3] H. Ulusoy. Forming coalitions through negotiation in multi-agent environments. Master's thesis, Middle East Technical University, Department Of Computer Engineering, December 1999.

Real Time Target Evaluation Search

Cagatay Undeger
Department of Computer Engineering
Middle East Technical University
06531 Ankara, Turkey

cundeger@ceng.metu.edu.tr

Faruk Polat
Department of Computer Engineering
Middle East Technical University
06531 Ankara, Turkey

polat@ceng.metu.edu.tr

ABSTRACT
In this paper we propose a real-time search algorithm called Real-Time Target Evaluation Search (RTTES) for the problem of searching a route in grid worlds from a starting point to a static or dynamic target point in real-time. The algorithm makes use of a new effective heuristic method which utilizes environmental information to successfully find solution paths to the target in dynamic and partially observable environments. The method requires analysis of obstacles to determine closed directions and estimate the goal relevance of open directions in order to identify the most beneficial move. The environment is assumed to be a planar grid and the agent has limited perception. In this paper, we compared RTTES with Real-Time A* (RTA*) and Real-Time Edge Follow (RTEF), and observed a significant improvement.

Categories and Subject Descriptors
I.2 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms
Algorithms, Performance, Experimentation

Keywords
Path planning, Grid world, Real-time search, Heuristic search

1. INTRODUCTION
Path planning can be described as finding a path from an initial point to a target point if there exists one. Path planning algorithms are either off-line or on-line. Off-line algorithms find the whole solution in advance before starting. These algorithms suffer from execution time in dynamic or partially observable environments due to frequent re-planning requirements. In on-line case, an agent repeatedly plans its next move in limited time and executes it, but they are not designed to be optimal and usually find poor solutions with respect to path length. Furthermore, there exist some hybrid solutions such as incremental heuristic search algorithms that are more efficient than off-line path planning. However, they are still slow for some real-time applications and are not applicable to moving targets. There are some on-line algorithms trying to overcome these difficulties. For instance, Real-Time Edge Follow (RTEF) [16, 17] uses a powerful heuristic function that can discard some non-promising alternative moving directions in real-time to guide the agent to a static or dynamic target. Although RTEF is able to determine the closed (non-promising) directions successfully, it is weak in selecting the right move from the rest of the alternatives since it uses poor Euclidian distance heuristic. Therefore, we focused on a new method for better selection.

In this paper, we propose a real-time search algorithm, Real-Time Target Evaluation Search (RTTES), and a heuristic method, Real-Time Target Evaluation (RTTE) capable of estimating distance to the target considering the obstacles. The method sends rays away from the agent in four directions, and determines obstacles that the rays hit. For each such obstacle, we extract its border and determine best direction that avoids the obstacle. Finally, by using a resolution mechanism, one of the proposed directions is chosen.

In the next section, the related work is given. In section 3, RTTES is described in brief. The performance analysis is presented in section 4. And finally, the conclusion is given in section 5.

2. RELATED WORK
Optimal [11, 14] and probabilistic [2, 4, 7, 9, 10] off-line path planning algorithms are hard to use for large dynamic environments because of their time requirements. One solution is to make off-line algorithms to be incremental [5, 6, 12, 13] to avoid re-planning from scratch. Although incremental algorithms are efficient in most cases, sometimes a small change in the environment may cause to re-plan almost from scratch. Due to the efficiency problems of off-line techniques, a number of on-line approaches such as Learning Real-Time A* (LRTA*), Real-Time A* (RTA*) [8], Real-Time Horizontal A* (RTHA*) [15], Execution Extended Rapidly Exploring Random Trees [1], and PRM with Kinodynamic Motion Planner [3] are proposed.

Recently, a new on-line path search algorithm (Real-Time Edge Follow - RTEF) [16, 17] is proposed for grid-type environments. RTEF uses a new heuristic (RTEF-ARM) that effectively makes use of global environmental information. With this heuristic, the agent can detect closed directions (directions that cannot reach the target) using perceptual data and tentative map, and determine its next move from open directions. Experimental study shows that RTEF performs better than RTA* and RTA* with n-look-ahead depth in terms of solution path and execution time [17].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8–12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005…$5.00.

3. RTTES & RTTE

We assume that the environment is a planar grid and is partially known by the agent. The agent located at a particular cell is required to reach a target cell avoiding obstacles in real-time. At any step, agent can move north, south, east or west, and is able to maintain a tentative map in its memory while exploration.

RTTES uses the heuristic method RTTE, which analyzes obstacles and proposes moving directions to reach the target through approximately short paths avoiding the obstacles. For this, RTTE geometrically analyzes the obstacles nearby and tries to estimate lengths of paths around obstacles to reach the target. To avoid infinite loops and re-visiting the same locations redundantly, RTTES uses two mechanisms (*visited count grid* & *history*). *Visited count grid* is a 2D matrix storing the number of visits to the cells, and *history* is the set of previously visited marked cells. History cells are assumed to be obstacle. When exploring an unknown environment, the agent may realize that all the ways leading to the target are blocked. In such a case the history should be cleared in order to let the agent backtrack as history cells may cause this. RTTES repeats the following steps until reaching the target or detecting that the target is inaccessible:

1) Use RTTE to compute proposed direction and utilities of neighbor cells
2) If a direction is proposed by RTTE:
   a) Select the neighbor cell with the highest utility from the set of non-obstacle neighbors with the smallest visited count
   b) Move to the selected direction, increment the visited count of the previous cell by one, and insert the cell into the *History*
3) Else if *History* is not empty, clear all the *History*
4) Else destination is unreachable, stop the search with failure.

The RTTE algorithm works as follows:

1) Mark all the moving directions as open
2) Propagate four rays away from the agent to north, south, east and west
2) For each ray hitting an obstacle:
   a) Trace and extract the border of the obstacle
   b) Analyze the border by re-tracing it from left and right sides
   c) Detect closed directions
   d) Evaluate results and determine a direction to avoid obstacle
3) Merge individual results, propose a direction to move, and compute utilities of neighbor cells

As seen above, RTTE propagates four rays from the agent location to north, south, east and west directions, and analyzes the obstacles these rays hit to find out the best direction to move. If a ray hits an obstacle before exceeding the maximum ray distance, the obstacle border is extracted by tracing cells on the border starting from the hit-point. Next the border is re-traced from both left and right sides to determine the geometric features of the obstacle. The closed moving directions are determined similar to RTEF [17]. The obstacle features are evaluated and a moving direction to avoid the obstacle is identified. After all the obstacles are evaluated, the results are merged in order to propose a final moving direction.

4. PEFORMANCE ANALYSIS

In this paper, we compared RTTES with RTA* and RTEF on randomly generated sample grids. We used RTTES and RTEF with two variations: "VC" that only uses visited counts and "VCH" that uses both history and visited counts. Thus, we tested five algorithms: RTTES-VC, RTTES-VCH, RTEF-VC, RTEF-VCH and RTA*. We used grids of three different types: *random*, *maze* and *U-type* with size 200x200. *Random* grids are generated randomly based on different obstacle ratios. *Maze* grids are the ones where every two non-obstacle cells are always connected through a path (usually one path). *U-type* grids are created by randomly putting U-shaped obstacles. We assume that the agent has limited vision (called *visual depth v)* and perceives only the grid cells within the square region *(2v+1)*(2v+1)* centered at agent's location. We take the visual depth as 20 or *infinite* (*full vision*) which means the agent knows the entire grid world in advance.

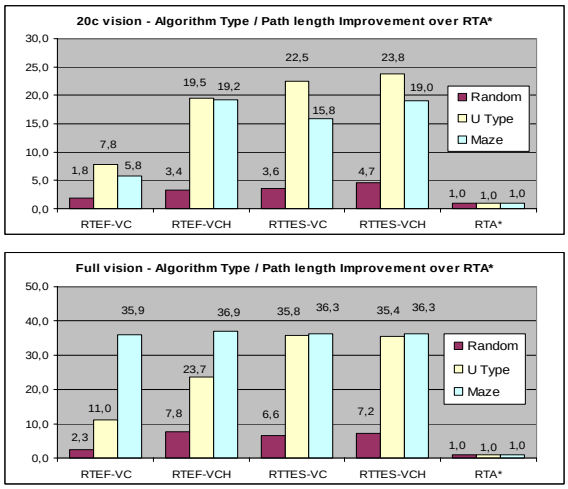


Figure 1. Path Length comparison of 20 cell and full vision

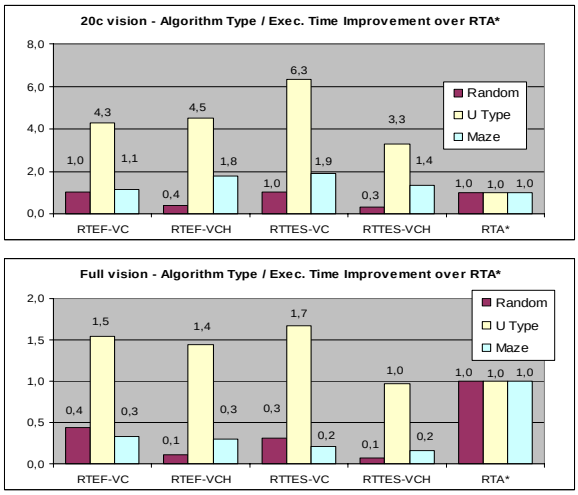


Figure 2. Execution Time comparison of 20 cell and full vision

Figure 1 contains test results that show improvements over RTA* in path length (i.e., path length of RTA* divided by these of compared algorithms) for different grid types. We observed that RTEF and RTTES performed significantly better than RTA* all the time. In random grids, RTEF-VCH and RTTES variations were almost head to head. In maze grids, all the RTEF and RTTES variations performed almost the same with full vision. With 20 cell vision, the results of RTEF-VCH and RTTES-VCH were very close to each other, and they performed better than RTEF-VC and RTTES-VC on the average. In U-type grids, the

results of RTTES variations were almost the same, and they performed better than RTEF variations on the average.

Figure 2 contains comparison of total execution times of previous runs similarly. We report ratios in terms of execution times taking RTA* as basis. In random grids with 20 cell vision RTEF-VC, RTTES-VC and RTA* performed almost the same, and were better than RTEF-VCH and RTTES-VCH. In random grids with full vision, RTA* is better since the efficiency of RTEF and RTTES usually drop in fully known grids. In maze grids, RTEF-VCH and RTTES-VC performed similarly, and were better than the others with 20 cell vision. With full vision, RTA* is better. In U-type grids, RTEF and RTTES usually performed much better than RTA*. The results of RTTES-VC were the best among all.

We also compared the path length of the algorithms with the optimal solutions generated by A* in fully known grids. The ratios of the algorithms' solution path lengths over optimal paths lengths are presented in Tables 1-2. The results show that solutions of both RTTES variations were only 1.14 times longer than the optimal ones on the average, whereas the solutions of RTEF-VC, RTEF-VCH and RTA* were 4.39, 1.5 and 33 times longer respectively. The standard deviations of RTTES algorithms were significantly less than RTEF algorithms. And we see that best improvement was obtained in U-type grids.

Table 1. Average ratios of algorithms' path lengths over optimal path lengths

	RT-VC	RT-VCH	RTTES-VC	RTTES-VCH	RTA*
Average	4,394	1,501	1,142	1,140	33,022
Standart Deviation	8,555	1,068	0,194	0,169	50,417

Table 2. Average ratios of algorithms' path lengths over optimal path lengths in random, U-type, and maze grids

	RT-VC	RT-VCH	RTTES-VC	RTTES-VCH	RTA*
Random Grids	3,577	1,432	1,369	1,339	10,532
Maze Grids	1,967	1,346	1,078	1,081	37,800
U-Type Grids	10,467	1,903	1,113	1,123	39,141

5. CONCLUSION

In this paper, we have focused on real-time search for grid-type problems, and presented an effective heuristic method (RTTE) and a real-time search algorithm (RTTES).

We have compared RTA*, RTEF and RTTES with the help of 1600 test runs. With respect to the path length, experimental results showed that RTTES is able to make use of environmental information very successfully to improve the solutions. In U-type grids, RTTES discovers much shorter paths compared to RTEF and RTA*. In random and maze grids, RTTES and RTEF are almost head to head, and both are much better than RTA*. With respect to execution time, we have observed that RTTES-VC is highly efficient in U-type grids. In random and maze grids, RTEF and RTTES almost perform the same, but it is hard to say about RTA* since the results change much depending on the vision range. With limited vision, the agent knows less about its environment, which increases the efficiency of RTEF and RTTES, and makes them perform usually better than RTA*. The performance decrease in case of unlimited vision is due to the fact that the agent knows more than it requires. We have also seen that RTTES converges to almost optimal solutions in fully known grids with respect to path length.

6. REFERENCES

[1] Bruce, J. and Veloso, M. *Real-time randomized path planning for robot navigation*. In Proceedings of IEEE Int'l Conf. on Intelligent Robots and Systems, pp. 2383-2388, Piscataway, N.J.: IEEE, 2002.

[2] Cheng, P. and LaValle, S. M. *Resolution complete rapidly-exploring random trees*. In Proceedings of IEEE Int'l Conf. on Robotics and Automation, pp. 267-272, 2002.

[3] Hsu, D., Kindel, R., Latombe, J.C. and Rock, S. *Randomized Kinodynamic Motion Planning with Moving Obstacles*. Int. J. Robotics Research, 21(3):233–255, 2002.

[4] Kavraki, L. and Latombe, J.C. *Probabilistic Roadmaps for Robot Path Planning*. In Practical Motion Planning in Robotics: Current and Future Directions, K. Gupta & A. del Pobil Eds., Addison-Wesley, 1998.

[5] Koenig, S. and Likhachev, M. *D* Lite*. In Proceedings of the National Conference on Artificial Intelligence, 2002.

[6] Koenig, S. and Likhachev, M. *Improved Fast Replanning For Robot Navigation in Unknown Terrain*. In Proceedings of the Int'l Conf. on Robotics and Automation, 2002.

[7] Konar, A. *Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of Human Brain*, CRC Press LLC, 2000.

[8] Korf, R.E. *Real-Time Heuristic Search*. Artificial Intelligence, Vol. 42, No: 2-3, pp. 189-211. 1990.

[9] LaValle, S. M. and Kuffner, J. J. *Rapidly-exploring random trees: Progress and prospects*. Algorithmic and Computational Robotics: New Directions, pp. 293-308. A K Peters, Wellesley, MA, 2001.

[10] Michalewicz, Z. *Genetic Algorithms + Data Structure = Evolution Programs*, Springer-Verlag, New York, 1986.

[11] Russell, S. and Norving, P. *Artificial Intelligence: a modern approach*. Prentice Hall, Inc. 1995.

[12] Stentz, A. *Optimal and Efficient Path Planning for Partially-Known Environments*. In Proceedings of IEEE Int'l Conf. on Robotics and Automation, May, 1994.

[13] Stentz, A. *The Focussed D* Algorithm for Real-Time Replanning*. In Proceedings of Int'l Joint Conf. on Artificial Intelligence, August, 1995.

[14] Tanenbaum, A.S. *Computer Networks*, Prentice-Hall, Inc., pp 349-351, 1996.

[15] Undeger, C. *Real-Time Mission Planning For Virtual Human Agents*. M.S. Thesis in Computer Engineering Department of Middle East Technical University. Jan 2001.

[16] Undeger, C., Polat, F. and Ipekkan, Z. *Real-Time Edge Follow: A New Paradigm to Real-Time Path Search*. In Proceedings of GAME-ON 2001, London, England, 2001.

[17] Undeger, C. and Polat, F. *Real-Time Edge Follow: A Real-Time Path Search Approach*. IEEE Transaction on Systems, Man and Cybernetics, Part C (Accepted for publication).

Organizational Self-Design in Semi-dynamic Environments

Sachin Kamboj and Keith S. Decker
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716
{kamboj, decker}@cis.udel.edu

ABSTRACT

In this paper we propose a run-time approach to organization that is contingent on the task structure of the problem being solved and the environmental conditions under which it is being solved. We use TÆMS as the underlying representation for our problems and describe a framework that uses Organizational Self-Design (OSD) to allocate tasks and resources to the agents and coordinate their activities.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms, Design, Performance

Keywords

Organizational-Self Design, Task and Resource Allocation

1. INTRODUCTION

In this paper, we are primarily interested in the organizational design of a multiagent system — the roles enacted by the agents, the coordination between the roles and the number and assignment of roles and resources to the individual agents. The organizational design is complicated by the fact that there is no best way to organize and all ways of organizing are not equally effective[1]. Instead, the optimal organizational structure depends both on the problem at hand and the environmental conditions under which the problem needs to be solved. The environmental conditions may not be known *a priori* or may change over time, which would preclude the use of a static organizational structure. On the other hand, all problem instances and environmental conditions are not always unique which would rule out the use of a new, bespoke organizational structure for every problem instance.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

In our approach we use Organizational Self-Design (OSD) to dynamically alter the organizational structure of the agents. We define two operators for OSD — agent spawning and composition — when an agent becomes overloaded, it spawns off a new agent to handle part of its task load/responsibility; when an agent lies idle for an extended period of time, it may decide to compose with another (underloaded) agent.

Our work builds on the work by [2]. The primary difference between their work and our work is that we use TÆMS [3] as the underlying representation for our problems. TÆMS is a computational framework that uses annotated hierarchical task networks (HTNs) to allow quantitative reasoning over the task structures. TÆMS allows us to explicitly reason about alternative ways of doing a task, arbitrary ways of combining subtasks, uncertainties, quality/cost tradeoffs, and non-local effects and is hence more general than the approach used by [2].

2. ORGANIZATIONAL SELF DESIGN

2.1 Agent Roles and Relationships

As explained in Section 1, the organizational structure is primarily composed of roles and the relationships between the roles. One or more agents may enact a particular role and one or more roles must be enacted by every agent. The roles may be thought of as the parts played by the agents enacting the roles in the solution to the problem and reflect the long-term commitments made by the agents in question to a certain course of action (that includes task responsibility, authority, and mechanisms for coordination). The relationships between the roles are the coordination relationships that exist between the subparts of a problem.

In our approach, the organizational design is directly contingent on the task structure and the environmental conditions under which the problems need to be solved. We define a role as a TÆMS subtree rooted at a particular node. Note, by definition, a role may consist of one or more other (sub-) roles as a particular TÆMS node may itself be made up of one or more subtrees. Hence, we will use the terms role, task node and task interchangeably.

We, also, differentiate between local and managed (non-local) roles. Local roles are roles that are the sole responsibility of a single agent, that is, the agent concerned is responsible for solving all the subproblems of the tree rooted at that node. For such roles, the agent concerned can do one or more subtasks, solely at its discretion and without consultation with any other agent. Managed roles, on the other hand, must be coordinated between two or more agents

as such roles will have two or more descendent local roles that are the responsibility of two or more separate agents. We achieve this coordination by assigning one of the agents as the manager responsible for enacting the non-local role. This manager is responsible for making the coordination decisions, often in consultation with the agents enacting the descendent sub-roles of that particular non-local role.

2.2 Organization Formation and Adaptation

To form or adapt their organizational structure, the agents use two organizational primitives: agent spawning and composition. These two primitives result in a change in the assignment of roles to the agents. Agent spawning is the generation of a new agent to handle a subset of the roles of the spawning agent. Agent composition, on the other hand, is orthogonal to agent spawning and involves the merging of two or more agents together — the combined agent is responsible for enacting all the roles of the agents being merged. Hence, OSD can be thought of as a search in the space of all the role assignments for a suitable role assignment that minimizes or maximizes a performance measure

In order to participate in the formation and adaption of an organization, the agents need to explicitly represent and reason about the role assignments. Hence, as a part of its organizational knowledge, each agent keeps a list of the local roles that it is enacting and the non-local roles that it is managing. Note that each agent only has limited organizational knowledge and is individually responsible for spawning off or combining with another agent, as needed, based on its estimate of its performance so far.

To see how the organizational primitives work, we first describe four rules that can be thought of as the organizational invariants which will always hold before and after any organizational change:

1. For a local role, all the descendent nodes of that role will be local.
2. Similarly, for a managed (non-local) role, all the ascendent nodes of that role will be managed.
3. If two local roles that are enacted by two different agents share a common ancestor, that ancestor will be a managed role.
4. If all the direct descendants of a role are local and the sole responsibility of a single agent, that role will be a local role.

When a new agent is spawned, the agent doing the spawning will assign one or more of its local roles to the newly spawned agent. To preserve invariant rules 2 and 3, the spawning agent will change the type of all the ascendent roles of the nodes assigned to the newly spawned agent from local to managed. Note that the spawning agent is only changing its local organizational knowledge and not the global organizational knowledge. At the same time, the spawning agent is taking on the task of managing the previously local roles. Similarly, the newly spawned agent will only know of its just assigned local roles.

When an agent (the *composing* agent) decides to compose with another agent (the *composed* agent), the organizational knowledge of the composing agent is merged with the organizational knowledge of the composed agent. To do this, the composed agent takes on the roles of all the local and managed tasks of the composing agent. Care is taken to preserve the organizational invariant rules 1 and 4.

2.3 Reasons for Organizational Change

As organizational change is expensive (requiring clock cycles, allocation/deallocation of resources, etc.) we want a stable organizational structure that is suited to the task and environmental conditions at hand. Hence, we wish to change the organizational structure only if the task structure and/or environmental conditions change. Also to allow temporary changes to the environmental conditions to be overlooked, we want the probability of an organizational change to be inversely proportional to the time since the last organizational change. If this time is relatively short, the agents are still adjusting to the changes in the environment - hence the probability of an agent initiating an organizational change should be high. Similarly, if the time since the last organizational change is relatively large, we wish to have a low probability of organizational change.

To allow this variation in probability of organizational change, we use simulated annealing to determine the probability of keeping an existing organizational structure. This probability is calculated using the annealing formula: $p = e^{-\frac{\Delta E}{kT}}$ where ΔE is the "amount" of overload/underload, T is the time since the last organizational change and k is a constant. The mechanism of computing ΔE is different for agent spawning than for agent composition and is described below. From this formula, if T is large, p, or the probability of keeping the existing organizational structure is large.

Agent spawning only occurs when the agent doing the spawning is too overloaded and cannot complete all the tasks in its task queue by the given deadlines of the tasks. To compute if spawning is necessary, we use the annealing equation with $\Delta E = \frac{1}{\alpha * Slack}$ where α is a constant and *Slack* is the difference between the total time available for completion of the outstanding tasks and the sum of the expected time required for completion of each task on the task queue.

Agent composition, on the other hand, is exactly orthogonal to agent spawning as agent composition only occurs when the agents are underloaded. In such a situation, some of the agents will be sitting idle waiting for tasks to arrive. These idle agents will either be utilizing resources while waiting, or more likely, will have resources allocated to them that could be used elsewhere in the system. In either case, it makes sense to combine some of the idle agents with other agents freeing precious resources.

To calculate if agent composition is necessary, we again use the simulated annealing equation. However, in this case, $\Delta E = \beta * Idle\_Time$, where β is a constant and $Idle\_Time$ is the amount of time for which the agent was idle. If the agent has been sitting idle for a long period of time, ΔE is large, which implies that p, the probability of keeping the existing organizational structure, is low.

3. EVALUATION

To evaluate our approach, we ran a series of experiments that simulated the operation of both the OSD agents and the Contract Net agents on various task structures with varied arrival rates and deadlines. At the start of each experiment, a random TÆMS task structure was generated with a specified depth and branching factor. During the course of the experiment, a series of task instances arrive at the organization and must be completed by the agents before their specified deadlines.

To directly compare the OSD approach with the Contract

Net approach, each experiment was repeated several times — using OSD agents on the first run and a different number of Contract Net agents on each subsequent run. We were careful to use the same task structure, task arrival times, task deadlines and random numbers for each of these trials.

We divided the experiments into two groups: experiments in which the environment was static (fixed task arrival rates and deadlines) and experiments in which the environment was dynamic (varying arrival rates and/or deadlines).

The two graphs in Figure 1, show the average performance of the OSD organization against the Contract Net organizations with 8, 10, 12 and 14 agents. The results shown are the averages of running 40 experiments. 20 of those experiments had a static environment with a fixed task arrival time of 15 cycles and a deadline window of 20 cycles. The remaining 20 experiments had a varying task arrival rate - the task arrival rate was changed from 15 cycles to 30 cycles and back to 15 cycles after every 20 tasks. In all the experiments, the task structures were randomly generated with a maximum depth of 4 and a maximum branching factor of 3. The runtime of all the experiments was 2500 cycles.

We tested several hypotheses relating to the comparative performance of our OSD approach using the *Wilcoxon Matched-Pair Signed-Rank* tests. *Matched-Pair* signifies that we are comparing the performance of each system on precisely the same randomized task set within each separate experiment. The tested hypothesis are:

The OSD organization requires fewer agents to complete an equal or larger number of tasks when compared to the Contract Net organization: To test this hypothesis, we tested the stronger null hypothesis that states that the contract net agents complete more tasks. This null hypothesis is rejected for all contract net organizations with less than 14 agents (static: $p < 0.0003$; dynamic: $p < 0.03$). For large contract net organizations, the number of tasks completed is statistically equivalent to the number completed by the OSD agents, however the *number of agents* used by the OSD organization is *smaller*: 9.59 agents (in the static case) and 7.38 agents (in the dynamic case) versus 14 contract net agents. Thus the original hypothesis, that OSD requires fewer agents to complete an equal or larger number of tasks, is upheld.

The OSD organizations achieve an equal or greater average quality than the Contract Net organizations: The null hypothesis is that the Contract Net agents achieve a greater average quality. We can reject the null hypothesis for contract net organizations with less than 12 agents (static: $p < 0.01$; dynamic: $p < 0.05$). For larger contract net organizations, the average quality is statistically equivalent to that achieved by OSD.

The OSD agents have a lower average response time as compared to the Contract Net agents: The null hypothesis that OSD has the same or higher response time is rejected for all contract net organizations (static: $p < 0.0002$; dynamic: $p < 0.0004$).

The OSD agents send less messages than the Contract Net Agents: The null hypothesis that OSD sends the same or more messages is rejected for all contract net organizations ($p < .0003$ in all cases except 8 contract net agents in a static environment where $p < 0.02$)

Hence, as demonstrated by the above tests, our agents perform better than the contract net agents as they complete a larger number of tasks, achieve a greater quality and also have a lower response time and communication overhead. These results make intuitive sense given our goals for the OSD approach. We expected the OSD organizations to have a faster average response time and to send less messages because the agents in the OSD organization are not wasting time and messages sending bid requests and replying to bids. The quality gained on the tasks is directly dependent on the number of tasks completed, hence the more the number of tasks completed, the greater average quality. The results of testing the first hypothesis were slightly more surprising. It appears that due to the inherent inefficiency of the contract net protocol in bidding for each and every task instance, a greater number of agents are needed to complete an equal number of tasks.

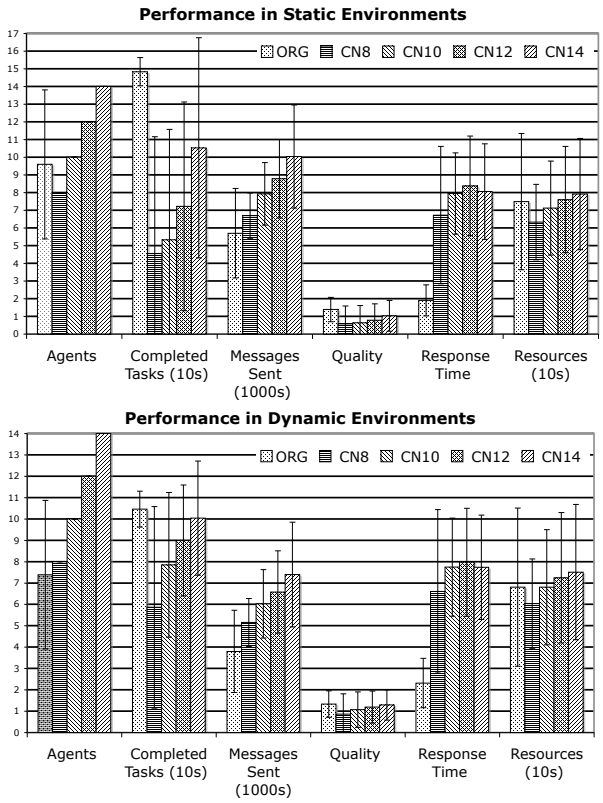


Figure 1: Graph comparing the average performance of the OSD organization with the Contract Net organizations (with 8, 10, 12 and 14 agents). The error bars show the standard deviations.

4. REFERENCES

[1] K. M. Carley and L. Gasser. Computational organization theory. In G. Wiess, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 299–330, MIT Press, 1999.

[2] L. Gasser and T. Ishida. A Dynamic Organizational Architecture for Adaptive Problem Solving. In *Ninth National Conference on Artificial Intelligence*, pp. 185–190, Menlo Park, California, 1991. AAAI Press.

[3] V. Lesser, et. al. Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. *AAMAS '02 (Plenary Lecture)*, pp. 1–2, 2002.

A Decision-Theoretic Approach to File Consistency in Constrained Peer-to-Peer Device Networks

David L. Roberts Sooraj Bhat Charles L. Isbell Jr.
Brian F. Cooper Jeffrey S. Pierce
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0706
{robertsd,sooraj,isbell,cooperb,jpierce}@cc.gatech.edu

ABSTRACT

As users interact with an increasing array of personal computing devices, maintaining data consistency across those devices becomes more dif cult. Typical solutions assume access to centralized servers, continual connectivity, or unbounded storage and CPU capacity. In practice, users' devices vary widely in capabilities, use intermittent or sparsely-connected networks and incur (asymmetric) transfer costs. We present a multi-agent system built upon a decision-theoretic approach to constructing and executing multiple plans to achieve consistency in a peer-to-peer, partially observable, non-deterministic environment. We analyze the performance in comparison to a standard epidemic replication algorithm.

Categories & Subject Descriptors

I.2.8 Plan execution, formation, and generation

Keywords

multiagent, decision theory, device networks

1. INTRODUCTION AND MOTIVATION

We consider the problem of transparently ensuring that a user's data remains consistent across all her devices. We construct a *Personal Information Environment* (PIE), a virtual data space not tied to any one physical device. This provides the user the illusion that every le is available on every device all the time.

Users could try to use a single storage unit (such as a USB key or iPod) to store their les, mounting it on any device they are currently using. There are several problems with this approach. First, it requires the user to carry the storage unit and plug it in to every device she uses. If the user forgets to plug it in she will be unable to access her les. Second, these devices often have less than 2GB of storage, so only a subset of the user's les may t.

Alternatively the user could synchronize with a central le server; however, many devices have only intermittent connectivity. Even users with broadband connections can be behind a rewall, making it dif cult for the home (or work) device to be a le server.

When we move toward a decentralized automated solution, several dif cult problems arise. We must know the les the user will need on any particular device. The topology of the device network is not known and may change as devices come and go. Further, devices may not have a xed IP address or may reside behind rewalls. In this situation, the role of limited capacity devices such as PDAs and cell phones frequently on a user's person becomes that of a crucial vessel to transfer data. This turns out to be a problem of propagating information in a resource-constrained, distributed and peer-to-peer non-stationary network that is only locally observable.

Described this way, the problem seems hopeless; however, we have several advantages. First, users are often at least locally predictable. They cannot be in many places at once, allowing time to move data between devices. Second, a limited number of les are touched in a given day and the number of devices is quite small so the problem admits practical computational tractability.

In our approach each device identi es important data it might possess, other devices that need that data, and constructs *plans* to deliver that data (using any intermediaries). Each device adapts and updates the plans of other devices as they become aware of them.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. RELATED WORK

One of the best-known systems for managing information on mobile devices is Coda [5]. Coda uses manually constructed or heuristically derived hoard pro les that determine which les should be replicated to which devices. Coda does not support multi-hop transfers over resource-constrained devices.

There has also been a large amount of other work on update consistency in disconnected or weakly connected networks. Examples include FICUS [8] and Thor [7] (amongst others). The goal of most of these systems is to manage updates, propagating them to a primary copy or to various clients. Our goal is somewhat different: to place cached copies of les where a user will need them as proactively as possible. Once the replicas are placed, standard techniques can be used or extended to manage consistency.

In the *joint intentions* framework [6], the implied contract between agents requires that each agent, upon obtaining knowledge of another agent's goal, assumes the goal as its own until it is achieved or it becomes irrelevant. Goal existence information and goal status information is communicated via peer-to-peer updates.

We do not explicitly represent goals like some multi-agent systems, but our agents operate in unison to achieve a globally useful if not globally consistent state with an implied contract to make their best effort to complete all multi-plans they receive. In our system, goals are represented as multi-plans . These multi-plans are sent from agent to agent along with whatever data they were constructed for. A multi-plan can become invalidated when a device receives information that there is a newer version of the data

associated with that multi-plan. This is identi ed during a *merge* process that can trigger replanning.

Because the devices that comprise a PIE have varied characteristics, it is important for agents to have some notion of the ability of their peers. [1] presents a set of criteria metrics that indicate characteristics of an agent's abilities and intentions in heterogeneous, complex planning architectures. These metrics consist are *commitment*, *responsibility*, *authority*, and *independence*. Agents in our system are all committed to joint goals; however, different types of devices in a PIE have varying levels of responsibility, authority, and independence. We describe devices using pro les that allow more powerful devices to aid in the operations for less powerful devices.

Our approach exploits the fact that the state space is limited by the number of devices in a typical PIE which allows for a brute force greedy approach. We improve over the limitations suggested by [2] of early decision-theoretic planning systems such as [9]) by de ning innovative utility functions to encode more complex information about the environment.

3. DEFINITIONS

Our goal is to place data (or les) on different devices. A *placement* is an ordered list of *unique* device IDs on which a le is to be located. For example, $l(1,2,3,4)$ is the placement corresponding to a le being located on devices 1, 2, 3, and 4. A *plan*, P_i, is an ordered list of placements: $\{l_{i,1}, l_{i,2}, ..., l_{i,k}\}$ where $l_{i,j}$ denotes the j-th placement of plan P_i. We constrain each plan so that each of its placements is a pre x of the placement that follows. A plan can be abbreviated by its nal placement (*e.g.* $P_i \equiv l_{i,k}$). Each placement has an associated *utility*; that is, there is some value to having a particular le on a particular device or set of devices. We de ne the utility of a plan from those placement utilities: $U(P_i, j)$ is the utility of the plan P_i at position j ($U(P_i, j) = U(l(i_1, i_2, ..., i_j))$)

Let $p_{i,j}$ and $c_{i,j}$ be the probability and cost of transferring some bits successfully from device i to device j respectively. Then we can calculate the *expected utility* of a plan in a straightforward way. For notational convenience, let $L(P_i, j) = \prod_{x=1}^{j-1} p_{i_x, i_{x+1}}$ be the likelihood of data reaching device i_j from device i_1 following plan P_i; let $C(P_i, j) = \sum_{x=1}^{j-1} c_{i_x, i_{x+1}}$ be the cost associated with data reaching device i_j from device i_1 following plan P_i; let $NL(P_i, j)$ be the net loss[1] of the plan P_i at its placement $l_{i,j}$; and let $NV(P_i, j) = U(P_i, j-1) - C(P_i, j-1) - NL(P_i, j)$ be the net value associated with data reaching device i_{j-1} from device i_1 but not reaching device i_j following plan P_i. The expected utility of a plan is:

$$EU(P_k) \equiv \quad EU(l(k_1, k_2, ..., k_n))$$

$$= \quad (1 - p_{k_1,k_2}) * -NL(P_k, 2) \tag{1}$$

$$+ \quad \sum_{i=3}^{n} L(P_k, i-1)(1 - p_{k_{i-1},k_i}) * NV(P_k, i) \tag{2}$$

$$+ \quad L(P_k, n) * (U(P_k, n) - C(P_k, n)) \tag{3}$$

Expression 1 is the cost of failure of the entire plan scaled by the probability of it failing during the rst step. Expression 3 offsets the utility of the success of the plan with the cost of its execution and scales it by the probability of success. Expression 2 is a similar computation to Expression 3 but it considers the effect of the plan failing at each intermediate step.

[1]In our experiments, it is the utility not realized by failing to complete the plan: $NL(P_i, j) = U(\{l_{i,j}, ..., l_{i,n}\}) = U(\{l_{i,1}, ..., l_{i,n}\}) - U(\{l_{i,1}, ..., l_{i,j-1}\})$, where n is the number of placements in a plan.

A *multi-plan* is a (ordered) set of plans: $M_i = \{P_{i,1}, ..., P_{i,n}\}$. Making an assumption about independence, we de ne the expected utility of a multi-plan to be the sum of the expected utilities of the plans it contains: $EU(M_i) = \sum_{P_k \in M_i} EU(P_k)$.

We seek a locally optimal (in expected utility) set of multi-plans to approximate a globally optimal set. Because multi-plans are constructed in a partially observable environment, it would be infeasible to consider all plan interaction. Assuming independence allows concise and computationally tractable plan construction.

We are in the process of integrating multi-plans into the Accord [3] middleware system we are developing which will enable us to derive values and costs from actual user data. Speci cally, the utility of a placement for a le is based on the af nity of that le for each of the devices in the placement, typically implemented as a linear combination of reads and writes for that le. Transfer probability is the empirical probability of the success of transferring bits. Costs are related to the actual bandwidth consumption during a transfer and the amount of a device's resources that would be consumed receiving, storing, and/or sending a le. Regardless of the speci c representation of probabilities, utilities, and costs, our algorithm will select a multi-plan to maximize expected utility.

While each placement must be considered during multi-plan construction, possible placements change very slowly and are enumerated before any connection. Similarly, utility calculation can occur of ine and incrementally. Since the values of the costs and utilities are based on information gathered from other devices, some recalculation must occur in response to new information.

Information obtained during a connection about the location of a le in the system can have two effects. First, it can indicate that the le has arrived at a destination for which there is a pending multi-plan, invalidating the multi-plan. Second, it can indicate that a device previously believed to have a le no longer does, triggering the creation of a new multi-plan. Fortunately, the number of devices tends to be small, so new muli-plans are constructed ef ciently. Additionally, we allow for short circuit evaluation of multi-plans.

Each agent in the system uses the following strategy which can be contracted to a peer device if its processing power is limited: Identify every le it contains that it believes is the most recent version and that is out of sync with other devices.[2] For each such le, construct a multi-plan to transfer the le to any device needing it. When a connection is established with another device, exchange and synchronize le location information and all multi-plans (replanning/merging if necessary). The underlying data is then transfered in decreasing order of associated multi-plan utility.

This ordering allows the agent to make a decision about which les are most important to move to a connected device. As we shall see below, the most bene t is realized when either connectivity between devices or the capacity of any crucial device is limited. When device capacity is limited, a device must decide which les and associated multi-plans to accept or reject during transfers. In our case, a limited capacity device always accepts newer versions of les it already contains, but when faced with a choice, accepts new les only if the multi-plans associated with the new le have higher utility than the lowest utility multi-plans for les already on the device. We later refer to this as the *smart eject/reject policy*.

4. BOOTSTRAPPING

We ran bootstrap simulations using ve devices with 200 and 400 les in total.Initially, les were distributed uniformly across the devices. Connections between devices intermittent not available and were randomly terminated after 3 to 10 seconds. We ran

[2]The Accord middleware provides this information.

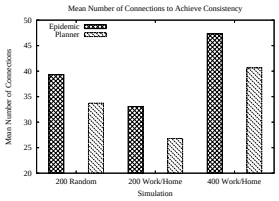


Figure 1: Performance during bootstrapping.

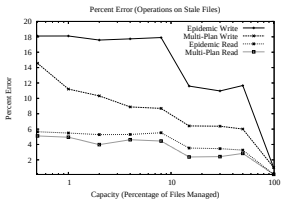


Figure 2: Performance as a function of "go between" capacity.

three sets of trials to determine performance in comparison to the epidemic replication algorithm [4], a popular mechanism for distributing data. In the epidemic model, each device simply transfers any files it has that are missing from any other device it comes in contact with. For each set of trials, the simulations were started from the same initial state and were considered complete when bootstrapping completed. For our experiments, one set of 15 trials was run with 200 files and a randomly generated connection scenario. The second set of 15 trials was run using 200 files and connection probabilities that encode a typical five device scenario: there are two machines in the work place behind a firewall, two machines at home behind a firewall, and one device that is carried between work and home that is the only means of connection between the two sets of machines. We refer to this as the bow tie scenario. The last set of 15 trials was run with 400 files and the bow tie connection scenario.

Figure 1 shows the results. There is a clear advantage to using our method in all cases. The multi-plans completed bootstrapping with an average 14.25% fewer connections for the randomly generated connection scenario, in an average of 14.08% fewer connections with 400 files, and an average of 18.95% fewer connections with 200 files and the bow tie scenario.

5. PERFORMANCE UNDER LOAD

Our simulator allows us to specify reads and writes on specific files over different periods of time.To capture meaningful *error*, we measured *stale* accesses: the number of reads and writes on files that are not the most recently modified version across all devices. In each scenario, a user travels between home and work in the bow tie. As before, the go-between is the only way to transfer data between home and work. Here, scenarios differ in the connectivity and storage characteristics of the go-between: a BlueTooth phone (low capacity, low connectivity); a BlueTooth PDA (low capacity, high connectivity when docked); and a laptop (high capacity, mixed connectivity depending on connection mode). All devices are capable of executing plans and using the smart eject/reject policy. When the epidemic is used, there is no utility data available so the limited capacity device randomly ejects files to accept new incoming files.

Scenario data was gathered over 20 trials using 1000 files. Only 450 of the files were eligible for reads and writes by the user. The capacity of the go-between device was varied from 0.05% to 100% of the number of files in the simulation. For simulations with capacity under 8%, the go-between was modeled as a cell phone; for simulations between 8% and 50%, as a PDA; for 100% capacity, as a laptop. For each test, we simulated eight days worth of user activity with a workload in which the working set of files changed after the first four days.

With the exception of the 100% capacity simulations, the multi-plans (in most cases significantly) outperformed the epidemic algorithm. With severely limited capacity (0.05%), the multi-plans did not reduce the error by as large a margin as with other capacities but still provided some benefit. In the best case (with the go-between cell phone having 8% capacity), the error rate on write operations was less than half of the error rate for the epidemic model.

6. CONCLUSION AND FUTURE WORK

We have described the problem of managing data across an array of resource-constrained devices. Because it is difficult for users to deal with this problem directly, we have argued for building a distributed support system to allow multiple agents to act on the user's behalf. We have presented motivation for the use of decision-theoretic multi-plans for file consistency management in a distributed peer-to-peer device network. We have shown that there is a distinct advantage to intelligently ordering file transfers in the presence of constrained device networks over current algorithms for distributed data replication.

Acknowledgments

We wish to thank Thomas M. Parks and Christopher Nevison for their donation of CPU time to gather data.

7. REFERENCES

[1] K. Barber and D. Han. Multi-agent planning under dynamic adaptive autonomy. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 1998.

[2] J. Blythe. Decision theoretic planning. *The AI Magazine*, 20(2):37–54, 1999.

[3] B. F. Cooper, C. L. Isbell, J. S. Pierce, D. L. Roberts, and S. Bhat. Accord: Middleware support for contextual, ubiquitous data management on user devices. Technical Report GIT-CC-06-06, College of Computing, Geogria Institute of Technology, 2006.

[4] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, August 1987.

[5] J. J. Kistler and M. Satyanarayanan. Disconnected operation in the Coda file system. *ACM Transactions on Computer Systems*, 10(1):3–25, Feb. 1992.

[6] H. J. Levesque, P. R. Cohen, and J. H. T. Nunes. On acting together. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 94–99, 1990.

[7] B. Liskov, P. Johnson, R. Gruber, and L. Shrira. A highly available object repository for use in a heterogeneous distributed system. In *Proceedings of the 4th International Workshop on Persistent Objects*, pages 255–266, 1990.

[8] P. L. Reiher, J. S. Heidemann, D. Ratner, G. Skinner, and G. J. Popek. Resolving file conflicts in the Ficus file system. In *USENIX Summer*, pages 183–195, 1994.

[9] M. Williamson and S. Hanks. Optimal planning with a goal-directed utility model. In K. J. Hammond, editor, *Proceedings of the Second International Conference on AI Planning Systems*, pages 176–181. American Association for Artificial Intelligence, 1994.

Solving POMDPs Using Quadratically Constrained Linear Programs

Christopher Amato
camato@cs.umass.edu

Daniel S. Bernstein
bern@cs.umass.edu

Shlomo Zilberstein
shlomo@cs.umass.edu

Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; G.1.6 [**Numerical Analysis**]: Optimization—*Constrained Optimization*

General Terms

Algorithms

Keywords

POMDPs, Planning under uncertainty, Optimization

1. INTRODUCTION

Since the early 1990's, Markov decision processes (MDPs) and their partially observable counterparts (POMDPs) have been widely used by the AI community for planning under uncertainty. POMDPs offer a rich language to describe situations involving uncertainty about the domain, stochastic actions, noisy observations, and a variety of possible objective functions. Even though an optimal solution may be concise, current exact algorithms that use dynamic programming often require an intractable amount of space. POMDP approximation algorithms can operate with a limited amount of memory, but as a consequence they provide very weak theoretical guarantees. In contrast, we describe a new approach that addresses the space requirement of POMDP algorithms while maintaining well-defined optimality guarantees.

Current methods to find optimal fixed-size controllers rely solely on local information [5, 6], thus can give solutions that are unboundedly below the global optimum. We present a formulation that defines the optimal fixed-size controller, and we employ standard nonlinearly constrained optimization techniques in order to find POMDP policies using this formulation. The quadratic constraints and linear objective function of our formulation belong to a special class of optimization problems for which many robust and efficient algorithms have been developed. While these techniques only guarantee locally optimal solutions, the new formulation facilitates a more efficient search of the solution space and produces high-quality results. Moreover, the algorithms employ more advanced techniques than those previously used [5, 6] to avoid convergence at certain suboptimal points. Using these locally optimal techniques, we demonstrate improved controller quality for a set of large POMDPs while maintaining similar running times when compared with a current state-of-the art approach. This suggests that by using our QCLP, small, high-valued controllers can be efficiently found for a large assortment of POMDPs.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. BACKGROUND

A POMDP can be defined as a tuple $\langle S, A, P, R, \Omega, O\rangle$, with S, a finite set of states with designated initial state distribution b_0, A, a finite set of actions, $P(s'|s,a)$, the state transition model, $R(s,a)$, the reward model, Ω, a finite set of observations, and $O(o|s',a)$, the observation model. In this paper, we consider the case in which the process unfolds over an infinite sequence of stages. At each stage the agent selects an action, which yields an immediate reward, and receives an observation. The agent must choose an action based on the history of observations seen. Because the state is not directly observed, it may be beneficial for the agent to remember the observation history. The agent's goal is to maximize the expected discounted sum of rewards received. To maintain finite sums, a discount rate, $0 \leq \gamma < 1$ is used.

Finite-state controllers are an elegant way of representing POMDP policies using a finite amount of memory. The controller can formally be defined by the tuple $\langle Q, \psi, \eta\rangle$, where Q is the finite set of controller nodes, $\psi : Q \rightarrow \Delta A$ is the action selection model for each node, and $\eta : Q \times A \times O \rightarrow \Delta Q$ represents the node transition model for each node, given an action was taken and an observation seen. We allow for stochastic transitions and action selection, as this can help to make up for limited memory [8]. The value of a node q at state s, given action selection and node transition probabilities, is provided by: $V(q,s) = \sum_a P(a|q)\big[R(s,a) + \gamma \sum_{s'} P(s'|s,a) \sum_o O(o|s',a) \sum_{q'} P(q'|q,a,o)V(q',s')\big]$ This equation is referred to as the Bellman equation.

Two previous POMDP solution techniques have sought to optimize fixed-size controllers. Poupart and Boutilier [6] have developed a method called bounded policy iteration (BPI). This approach alternates between policy improvement and evaluation, iterating through the nodes in the controller and using a linear program to examine the value of probabilistically taking an action and then transitioning into the old controller. If an improvement can be found, the action selection and node transition probabilities are updated

For variables: $x(q', a, q, o)$ and $y(q, s)$
Maximize $\sum_s b_0(s)y(q_0, s)$
Given the Bellman constraints:
$$\forall q, s \; y(q,s) = \sum_a \left[\left(\sum_{q'} x(q', a, q, o) \right) R(s,a) + \gamma \sum_{s'} P(s'|s,a) \sum_o O(o|s',a) \sum_{q'} x(q', a, q, o) y(q', s') \right]$$
And probability constraints:
$$\forall q, o \; \sum_{q',a} x(q', a, q, o) = 1, \; \forall q, o, a \; \sum_{q'} x(q', a, q, o) = \sum_{q'} x(q', a, q, o_k) \text{ and } \forall q', a, q, o \; x(q', a, q, o) \geq 0$$

Table 1: The QCLP for finding the optimal fixed-size controller. Variable $x(q', a, q, o)$ represents $P(q', a|q, o)$, $y(q, s)$ represents $V(q, s)$, q_0 is the initial controller node and o_k is an arbitrary fixed observation.

accordingly. This approach guarantees to at least maintain the value of a provided controller, but it is not likely to find a concise optimal controller. Also, while heuristics can be used to incorporate start state knowledge [7], they may be difficult to adjust and are not be applicable in all domains. Meuleau et al. [5] use gradient ascent based on a cross-product MDP created from the controller and the POMDP. Due to the complex and incomplete gradient calculation, this method can be time consuming and error prone.

3. OPTIMAL FIXED-SIZE CONTROLLERS

The linear program used by BPI may allow for controller improvement, but can easily get stuck in local maxima. While Poupart and Boutilier suggest heuristics for becoming unstuck, a nonlinear approach can allow these local solutions to be avoided. Also, since BPI requires that value is increased for all states, possible improvements are not always made. We believe that many POMDPs have small optimal controllers or can be approximated concisely. Optimization techniques permit these controllers to be found while maintaining moderate resource usage.

Unlike BPI and Meuleau et al.'s gradient ascent, our formulation defines the optimal controller for a given size. This is done by considering the values of each node and state pair variables. To ensure that these values are correct given the action selection and node transition probabilities, quadratic constraints must be added. This results in a quadratically constrained linear program (QCLP). QCLPs may contain quadratic terms in the constraints, but have a linear objective function. They are a subclass of general nonlinear programs that has structure which algorithms can exploit. Although it is often difficult to solve a QCLP exactly, many robust and efficient algorithms can be applied. Our QCLP has a simple gradient calculation and an intuitive representation that matches well with common optimization models.

Contrary to BPI, which alternates between policy improvement and evaluation, our QCLP improves and evaluates the controller in one phase. As seen in Table 1, the value of an initial node is maximized at an initial state distribution using parameters for the action selection and node transition probabilities at each node, $P(q', a|q, o)$, and the values of each node in each state, $V(q, s)$. To ensure that the values are correct given the action and node transition probabilities, nonlinear constraints representing the corresponding Bellman equations must be added to the optimization. The second and last constraints in the table ensure that the variables represent proper probabilities, and the third constraint guarantees that action selection does not depend on the observation that has not yet been seen.

THEOREM 1. *An optimal solution of the QCLP results in an optimal stochastic controller for the given size and initial state distribution.*

PROOF. The optimality of the controller follows from the Bellman equation constraints and maximization of a given node at the initial state distribution. The Bellman equation constraints restrict the value variables to valid amounts based on the chosen probabilities, while the maximum value is found for the initial node and state. Hence, this produces an optimal controller. □

4. EXPERIMENTS

For this paper, we used a freely available nonlinearly constrained optimization solver *snopt* [3] on the NEOS server. BPI was implemented in a manner very similar to the original techniques in [6]. Meuleau et al.'s gradient ascent was also implemented, but produced significantly worse results in substantially more time than the other techniques.

In our experiments, we used BPI and snopt to optimize a range of fixed-size controllers. Each method was initialized with the same ten random deterministic controllers and we report mean values and times. While the experiments were conducted on different computers, we expect solution times to only differ by a small constant factor in our experiments.

4.1 Hallway benchmark

The hallway domain introduced by Littman, Cassandra and Kaelbling [4], is a frequently used benchmark for POMDP algorithms. A robot must navigate a grid world with 57 states, 21 observations and 5 actions.

The results for this domain are shown in Table 2 and Figure 1. We see that for all controller sizes the mean value produced by the QCLP is more than twice that of BPI. Also, the time taken to produce these controllers remains similar, but the difference increases as controller size grows. It can be noted that much higher values can be produced by the

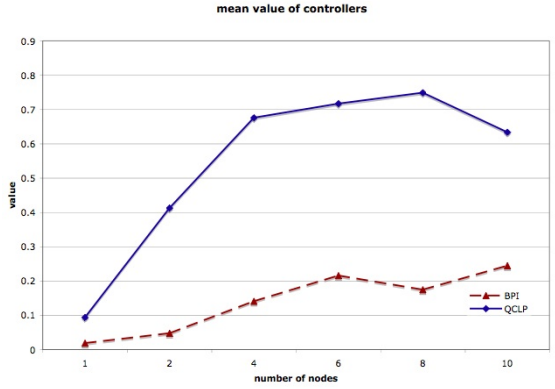


Figure 1: Hallway domain: the mean values using BPI and the QCLP for increasing controller size

# nodes	$QCLP_{hall}$	BPI_{hall}	$QCLP_{mach}$	BPI_{mach}
1	$<$ 1 min	$<$ 1 min	$<$ 1 min	1.3 mins
2	$<$ 1min	$<$ 1 min	$<$ 1 min	4.6 mins
4	$<$ 1 min	$<$ 1 min	7.9 mins	14.1 mins
6	1.4 mins	1.6 mins	42.4 mins	25.5 mins
8	6.9 mins	2.9 mins	57.5 mins	42.9 mins
10	9.1 mins	4 mins	130.8 mins	62.8 mins

Table 2: Mean running times for the QCLP and BPI on the hallway and machine problems

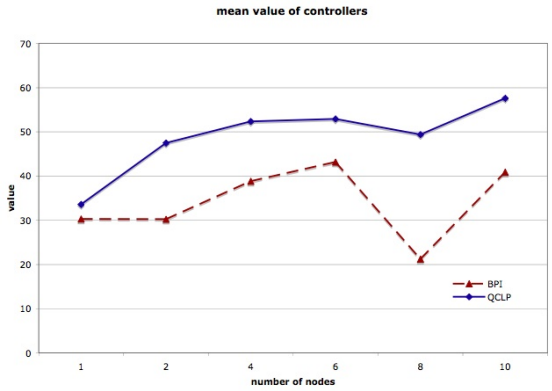


Figure 2: Machine domain: the mean values using BPI and the QCLP for increasing controller size

QCLP with markedly smaller controllers, thus reducing the time required.

Poupart and Boutilier also report results for BPI on a 1500 node controller. In their implementation, an escape technique that attempts to avoid local maxima was also used. After 69 hours, BPI produced a controller with a value of 0.51. Our QCLP formulation generates a controller with 65% higher value in 0.22% of the time with 0.67% of the representation size.

4.2 Machine maintenance

In order to test performance on a larger problem, we consider a machine maintenance domain with 256 states, 4 actions and 16 observations. For more details, see [2].

For this problem, the optimal policy for the underlying MDP is known to be 66.236. We see in Table 2 and Figure 2 that very small controllers produce solutions which are close to this upper bound. The techniques using the QCLP show higher mean values for all controller sizes than those produced by BPI. While the differences are less dramatic than the previous example, they are no less consistent. And although we again notice a disparity in running time as the number of nodes increases, we also notice that the mean value of the 10 node controller produced by BPI has lesser value and longer running time than the 2 node controller for the QCLP. This suggests that smaller controllers could replace those produced by BPI.

5. CONCLUSIONS

We introduced a new approach for solving POMDPs using a nonlinear programming formulation of the problem, which defines an optimal fixed-size stochastic controller. Near-optimal controllers can be found using standard nonlinear optimization techniques. This provides a promising new way for solving POMDPs and approximating optimal solutions with concise controllers. We showed that by using nonlinear optimization algorithms, we can produce higher valued controllers than those found by BPI, the existing state-of-the-art approach. We demonstrated empirically a consistent improvement over a range of controller sizes. We also showed that higher quality solutions can be found with significantly smaller controllers and similar running times. These results suggest that optimal or near-optimal controllers could be found for large POMDPs, making our approach very useful in a range of practical application domains.

In the future, we plan to examine the relationship between the nonlinear formulation and the performance of various optimization algorithms. Different representations may better match current nonlinearly constrained optimization methods and may thus produce better solutions. We also plan to study the applicability of the QCLP approach to problems involving multiple agents modeled as decentralized POMDPs. The BPI technique has already been generalized successfully to the multi-agent case [1]. In future work, we expect to be able to demonstrate similar performance gains in optimizing fixed-size controllers for a set of agents in large multi-agent domains.

6. ACKNOWLEDGMENTS

Support for this work was provided in part by the National Science Foundation under Grant No. IIS-0535061 and by the Air Force Office of Scientific Research under Agreement No. FA9550-05-1-0254.

7. REFERENCES

[1] D. S. Bernstein, E. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized POMDPs. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, 2005.

[2] A. R. Cassandra. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, Brown University, Providence, RI, 1998.

[3] P. E. Gill, W. Murray, and M. Saunders. Snopt: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47:99–131, 2005.

[4] M. L. Littman and A. R. C. L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. Technical Report CS-95-11, Brown University, Department of Computer Science, Providence, RI, 1995.

[5] N. Meuleau, K.-E. Kim, L. P. Kaelbling, and A. R. Cassandra. Solving POMDPs by searching the space of finite policies. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 417–426, Stockholm, Sweden, 1999.

[6] P. Poupart and C. Boutilier. Bounded finite state controllers. In *Advances in Neural Information Processing Systems*, 16, Vancouver, BC, 2003.

[7] P. Poupart and C. Boutilier. VDCBPI: an approximate scalable algorithm for large scale POMDPs. In *Advances in Neural Information Processing Systems*, 17, pages 1081–1088, Vancouver, BC, 2004.

[8] S. Singh, T. Jaakkola, and M. Jordan. Learning without state-estimation in partially observable Markovian decision processes. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 284–292, 1994.

Argumentation and negotiation

An Argumentation based Approach for Practical Reasoning

Iyad Rahwan,[1,2] Leila Amgoud[3]

[1]Institute of Informatics
British University in Dubai
POBox 502216, Dubai, UAE
irahwan@acm.org

[2](Fellow) School of Informatics
University of Edinburgh
Edinburgh, UK

[3]IRIT
118, route de Narbonne
31062, Toulouse, France
amgoud@irit.fr

ABSTRACT

We build on recent work on argumentation frameworks for generating desires and plans. We provide a rich instantiation of Dung's abstract argumentation framework for (i) generating consistent desires; and (ii) generating consistent plans for achieving these desires. This is done through three distinct argumentation frameworks: one (now standard) for arguing about beliefs, one for arguing about what desires the agent should adopt, and one for arguing about what plans to intend in order to achieve the agent's desires. More specifically, we refine and extend existing approaches by providing means for comparing arguments based on decision-theoretic notions (cf. utility). Thus, the *worth* of desires and the *cost* of resources are integrated into the argumentation frameworks and taken into account when comparing arguments.

Keywords

Argumentation in agent systems, formal models of agency

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*intelligent agents*

General Terms

Design, Languages, Theory

1. INTRODUCTION

Various frameworks have been proposed for formalising and mechanising the reasoning of autonomous software agents based on *mental attitudes* such as beliefs, desires and intentions (BDI). These range from theoretical models of mental attitudes using modal logics [13], to operational agent architectures such as AgentSpeak [5] and 3APL [8]. A central feature of reasoning with mental attitudes is that *conflict* may arise between various attitudes.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for pro t or commercial advantage and that copies bear this notice and the full citation on the rst page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior speci c permission and/or a fee.
AAMAS'06 May 8 12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1›59593›303›4/06/0005 ...$5.00.

Argumentation is a promising approach for reasoning with inconsistent information, based on the construction and the comparison of arguments [6]. The basic idea is that it should be possible to say more about the certainty of a particular fact than just assessing a probabilistic certainty degree in the interval $[0, 1]$. In particular, it should be possible to assess the reasons (i.e. *arguments*) why a fact holds, and to combine and compare these arguments in order to reach a conclusion. The process of *argumentation* may be viewed as a kind of reasoning about arguments (considering attacks and conflicts among them, comparing their strengths etc.) in order to determine the most *acceptable* of them. Various argument-based frameworks have been developed in defeasible reasoning [12] for generating and evaluating arguments.

Classicaly, argumentation has been mainly concerned with *theoretical reasoning*: reasoning about *propositional* attitudes such as knowledge and belief. Recently, a number of attempts have been made to use argumentation to capture *practical reasoning*: reasoning about what to do. This requires capturing arguments about non-propositional attitudes, such as desires and goals. Some argument-based frameworks for practical reasoning are instantiations of Dung's abstract framework [6] (e.g. [1, 3, 9]). Others are operational and grounded in logic programming (e.g. [10, 14]).

In this paper, we build on recent work on argumentation frameworks for generating desires and plans [1, 3, 9]. We provide a rich, argumentation-based framework for (i) generating consistent desires; and (ii) generating consistent plans for achieving these desires. This is done through three distinct argumentation frameworks: one (now standard) for arguing about beliefs, one for arguing about what desires the agent should adopt, and one for arguing about what plans to intend in order to achieve the agent's desires. More specifically, we refine and extend existing approaches by providing means for comparing arguments based on decision-theoretic notions (cf. utility). Thus, the *worth* of desires and the *cost* of resources are integrated into the argumentation frameworks and taken into account when comparing arguments.

The paper is organised as follows. After some formal preliminaries in the next section, we present our three integrated argumentation frameworks in Section 3. We discuss related work in Section 4 and conclude in Section 5.

2. PRELIMINARIES

In this section we start by presenting the logical language which will be used throughout this paper, as well as the different mental states of the agents (their bases).

Let $\mathcal{L}$ be a propositional language, $\vdash$ stands for classical

inference and $\equiv$ for logical equivalence. From $\mathcal{L}$ we can distinguish the three following sets of formulas:

- The set $\mathcal{D}$ which gathers all possible desires of agents.
- The set $\mathcal{K}$ which represents the knowledge.
- The set RES which contains all the available resources in a system.

From the above sets, two kinds of rules can be defined: *desire-generation* rules and *planning* rules.

DEFINITION 1. **(Desire-Generation Rules)** *A* desire-generation rule *(or a* desire rule*) is an expression of the form*

$$\varphi_1 \wedge \cdots \wedge \varphi_n \wedge \psi_1 \wedge \cdots \wedge \psi_m \Rightarrow \psi$$

where $\forall\ \varphi_i \in \mathcal{K}$ *and* $\forall\ \psi_i,\ \psi \in \mathcal{D}$.

The meaning of the rule is "if the agent *believes* $\varphi_1, \ldots, \varphi_n$ and *desires* $\psi_1, \ldots, \psi_m$, then the agent will *desire* ψ as well". And let $\texttt{head}(\varphi_1 \wedge \cdots \wedge \varphi_n \wedge \psi_1 \wedge \cdots \wedge \psi_m \Rightarrow \psi) = \psi$.

Let's now define the notion of *planning rule*, which is the basic building block for specifying plans.

DEFINITION 2. **(Planning Rules)** *A* planning *rule is an expression of the form*

$$\varphi_1 \wedge \cdots \wedge \varphi_n \wedge r_1 \cdots \wedge r_m \rightarrowtail \varphi$$

where $\forall\ \varphi_i \in \mathcal{D},\ \varphi \in \mathcal{D}$ *and* $\forall r_i \in RES$.

A planning rule expresses that if $\varphi_1, \ldots, \varphi_n$ are achieved and the resources $r_1, \ldots, r_m$ are used then φ is achieved.[1]

Let DGR and PR be the set of all possible desire generation rules and planning rules, respectively. Each agent is equipped with four bases: a base $\mathcal{B}_b$ containing its *basic beliefs*, a base $\mathcal{B}_d$ containing its desire-generation rules, a base $\mathcal{B}_p$ containing its planning rules and finally a base $\mathcal{R}$ which will gather all the resources possessed by that agent. Beliefs can be uncertain, desires may not have equal priority and resources may have different costs.

DEFINITION 3. **(Agent's bases)** *An agent is equipped with four bases* $\langle \mathcal{B}_b, \mathcal{B}_d, \mathcal{B}_p, \mathcal{R} \rangle$*:*

- $\mathcal{B}_b = \{(\beta_i, b_i) : \beta_i \in \mathcal{K}, b_i \in [0,1], i = 1, \ldots, n\}$. *Pair* (β_i, b_i) *means belief* β_i *is certain at least to degree* b_i.[2]
- $\mathcal{B}_d = \{(dgr_i, w_i) : dgr_i \in DGR, w_i \in \mathbb{R}, i = 1, \ldots, m\}$. *Symbol* w_i *denotes the* worth *of the desire* $\texttt{head}(dgr)$. *Let* $Worth(\psi) = w_i$.
- $\mathcal{B}_p = \{pr_i : pr_i \in PR, i = 1, \ldots, l\}$.
- $\mathcal{R} = \{(r_i, c_i), i = 1, \ldots, n\}$ *where* $r_i \in RES$ *and* $c_i \in \mathbb{R}$ *is the cost of consuming* r_i. *Let* $Cost(r_i) = c_i$ *be a function which returns the cost of a given resource.*

In what follows, $\mathcal{B}_b^*$, $\mathcal{B}_d^*$, $\mathcal{B}_p^*$, $\mathcal{R}^*$ will denote the sets of formulas when the weights are ignored. Using desire-generation rules, we can characterise *potential desires.*[3]

[1]Note that the implications defined in desire-generation rules and planning rules are not material. So for example, from $\neg y$ and $x \rightarrowtail y$, we cannot deduce $\neg x$.

[2]The certainty degree can be seen as a necessity measure of possibility theory.

[3]Amgoud and Kaci [3] call them "potential initial goals."

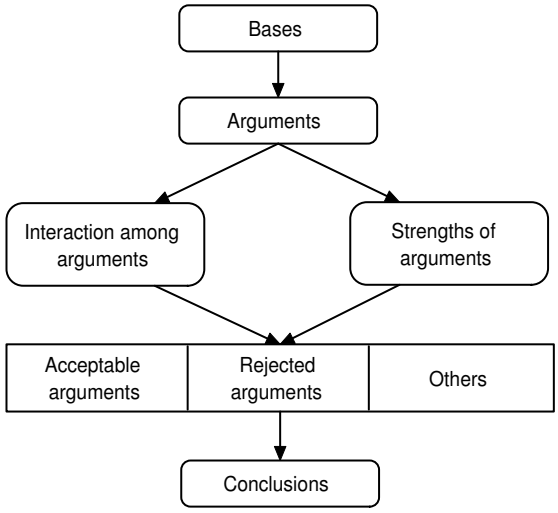


Figure 1: General view of argument-based decision making

DEFINITION 4. **(Potential Desire)** *The set of* potential desires *of an agent is* $\mathcal{PD} = \{\psi : \exists \varphi_1 \wedge \cdots \wedge \varphi_n \wedge \psi_1 \wedge \cdots \wedge \psi_m \Rightarrow \psi \in \mathcal{B}_d^*\}$.

These are "potential" desires because the agent does not know yet whether the antecedents (i.e. bodies) of the corresponding rules are true.

3. ARGUMENTATION FRAMEWORKS

The conceptual sketch of an argumentation framework is illustrated in Figure 1. It is essential to distinguish between arguing over beliefs and arguing over goals or desires. A proposition is believed because it is *true* and *relevant*. Desires, on the other hand, are adopted because they are *justified* and *achievable*. A desire is justified because the world is in a particular state that warrants its adoption. For example, one might desire to go for a walk because she believes it is a sunny day and may drop that desire if it started raining. A desire is *achievable*, on the other hand, if the agent has a plan that achieves that desire.

As a consequence of the different nature of beliefs and desires, they are supported by two different types of arguments. These arguments need to be treated differently, taking into account the different way they relate to one another. For example, a belief argument can be attacked by arguing that it is not consistent with observation, or because there is a reason to believe the contrary. Arguments for desires, on the other hand, could be attacked by demonstrating that the justification of that desire does not hold, or that the plan intended for achieving it is itself not achievable.

To deal with the different nature of the arguments involved, we present three distinct argumentation frameworks: one for reasoning about beliefs, another for arguing about what desires are justified and should be pursued, and a third for arguing about the best plan to intend in order to achieve these desires. The first framework is based on existing literature on argumentation over beliefs, originally proposed by Dung [6] and later extended by Amgoud and Cayrol [2]. For arguing about desires and plans, we draw on and extend

work on argumentation-based desire-generation and planning [1, 3, 9].

3.1 Arguing over beliefs

Using beliefs, an agent can construct *belief arguments*, which have a deductive form. Indeed, from a set of beliefs, another belief is deduced as follows:

DEFINITION 5. **(Belief Argument)** *A* belief argument *A is a pair $A = \langle H, h\rangle$ such that:*

1. *$H \subseteq \mathcal{B}_b^*$;*
2. *H is consistent;*
3. *$H \vdash h$;*
4. *H is minimal (for set $\subseteq$) among the sets satisfying conditions 1, 2, 3.*

The support *of the argument is denoted by $\texttt{SUPP}(A) = H$. The* conclusion *of the argument is denoted by $\texttt{CONC}(A) = h$. $\mathcal{A}_b$ stands for the set of all possible belief arguments that can be generated from a belief base $\mathcal{B}_b$.*

In [2, 11], it has been argued that arguments may have forces of various strengths, and consequently different definitions of the force of an argument have been proposed. Generally, the force of an argument can rely on the information from which it is constructed. Belief arguments involve only one kind of information: *the beliefs*. Thus, the arguments using more certain beliefs are found stronger than arguments using less certain beliefs. A certainty level is then associated with each argument. That level corresponds to the less entrenched belief used in the argument. This definition is also used in belief revision [7].

DEFINITION 6. **(Certainty level)** *Let $A = \langle H, h\rangle \in \mathcal{A}_b$. The* certainty level *of A is $Level(A) = min\{a_i : \varphi_i \in H$ and $(\varphi_i, a_i) \in \mathcal{B}_b\}$.*

The different forces of arguments make it possible to compare pairs of arguments. Indeed, the higher the certainty level of an argument is, the stronger that argument is. Formally:

DEFINITION 7. **(Comparing arguments)** *Let $A_1, A_2 \in \mathcal{A}_b$. The argument A_1 is* preferred *to A_2, denoted $A_1 \succeq_b A_2$, if and only if $Level(A_1) \geq Level(A_2)$.*

Preference relations between belief arguments are used not only to compare arguments in order to determine the "best" ones, but also in order to refine the notion of *acceptability* of arguments. Since a belief base may be inconsistent, then arguments may be conflicting.

DEFINITION 8. **(Conflicts between Belief Arguments)** *Let $A_1 = \langle H_1, h_1\rangle, A_2 = \langle H_2, h_2\rangle \in \mathcal{A}_b$.*

- *A_1* undercuts *A_2 if $\exists h_2' \in H_2$ such that $h_1 \equiv \neg h_2'$.*
- *A_1* attacks$_b$ *A_2 iff A_1 undercuts A_2 and not ($A_2 \succeq_b A_1$).*

Having defined the basic concepts, we are now ready to define the argumentation system for handling belief arguments.

DEFINITION 9. **(Belief Argumentation framework)** *An* argumentation framework *AF_b for handling belief arguments is a pair $AF_b = \langle \mathcal{A}_b, Attack_b\rangle$ where $\mathcal{A}_b$ is the set of belief arguments and $attack_b$ is the defeasibility relation between arguments in $\mathcal{A}_b$.*

Since arguments are conflicting, it is important to know what are the "good" ones, generally called *acceptable*. Beliefs supported by such arguments will be inferred from the base $\mathcal{B}_b$. Before defining the notion of acceptable arguments, let's first introduce a crucial notion of defence.

DEFINITION 10. **(Defence)** *Let $S \subseteq \mathcal{A}_b$ and $A_1 \in \mathcal{A}_b$. S* defends *A_1 iff for every belief argument A_2 where A_2 $attacks_b$ A_1, there is some argument $A_3 \in S$ such that A_3 $attacks_b$ A_2.*

An argument is acceptable either if it is not attacked, or if it is defended by acceptable arguments.

DEFINITION 11. **(Acceptable Belief Argument)** *A belief argument $A \in \mathcal{A}_b$ is* acceptable *with respect to a set of arguments $S \subseteq \mathcal{A}_b$ if either:*

- *$\nexists A' \in S$ such that A' $attacks_b$ A; or*
- *$\forall A' \in S$ such that A' $attacks_b$ A, we have an acceptable argument $A'' \in S$ such that A'' $attacks_b$ A'.*

This recursive definition enables us to characterise the set of acceptable arguments using a fixed-point definition.

PROPOSITION 1. *Let $AF_b = \langle \mathcal{A}_b, Attack_b\rangle$ be an argumentation framework. And let $\mathcal{F}$ be a function such that $\mathcal{F}(S) = \{A \in \mathcal{A}_b : S$ defends $A\}$. The set $Acc(\mathcal{A}_b)$ of* acceptable belief arguments *is defined as: $Acc(\mathcal{A}_b) = \bigcup \mathcal{F}_{i\geq 0}(\emptyset)$*

PROOF. Due to the use of propositional language and finite bases, the argumentation system is finitary, i.e each argument is attacked by a finite number of arguments. Since the argumentation system is finitary then the function $\mathcal{F}$ is continuous. Consequently, the least fixpoint of $\mathcal{F}$ is $\bigcup \mathcal{F}_{i\geq 0}(\emptyset)$. □

The set $Acc(\mathcal{A}_b)$ contains non-attacked arguments as well as arguments defended directly or indirectly by non-attacked ones.

3.2 Arguing over desires

Amgoud and Kaci have introduced *explanatory arguments* as a means for generating desires from beliefs [3]. We extend this framework in this section and refine it in order to resolve some problematic features caused by the fact that they combine belief argumentation with desire argumentation in a single framework. Moreover, we consider more general desire generation rules in the sense that a desire may not only be generated from beliefs as in [3], but it can also be generated from other desires.

In what follows, the functions $\texttt{BELIEFS}(A)$, $\texttt{DESIRES}(A)$ and $\texttt{CONC}(A)$ return respectively, for a given argument A, the beliefs used in A, the desires supported by A and the conclusion of the argument A.

DEFINITION 12. **(Explanatory Argument)** *Let $\langle \mathcal{B}_b, \mathcal{B}_d\rangle$ two bases.*

- *If* $\exists(\Rightarrow \phi) \in \mathcal{B}_d^*$ *then* $\Rightarrow \phi$ *is an* explanatory argument *(A) with:*
  $\texttt{BELIEFS}(A) = \emptyset$
  $\texttt{DESIRES}(A) = \{\phi\}$
  $\texttt{CONC}(A) = \phi$

- *If* $B_1, \ldots, B_n$ *are belief arguments, and* $E_1, \ldots, E_m$ *are explanatory arguments, and* $\exists$ $\texttt{CONC}(B_1) \wedge \ldots \wedge \texttt{CONC}(B_n) \wedge \texttt{CONC}(E_1) \wedge \ldots \wedge \texttt{CONC}(E_m) \Rightarrow \psi \in \mathcal{B}_d^*$ *then* $B_1, \ldots B_n, E_1, \ldots E_m \Rightarrow \psi$ *is an* explanatory argument *(A) with:*[4]
  $\texttt{BELIEFS}(A) = \texttt{SUPP}(B_1) \cup \ldots \cup \texttt{SUPP}(B_n) \cup \texttt{BELIEFS}(E_1) \cup \ldots \cup \texttt{BELIEFS}(E_m)$
  $\texttt{DESIRES}(A) = \texttt{DESIRES}(E_1) \cup \ldots \cup \texttt{DESIRES}(E_m) \cup \{\psi\}$
  $\texttt{CONC}(A) = \psi$

$TOP(A) = \texttt{CONC}(B_1) \wedge \ldots \texttt{CONC}(B_n) \wedge \texttt{CONC}(E_1) \wedge \ldots \wedge \texttt{CONC}(E_m) \Rightarrow \psi$ *is the* TOP rule *of the argument.*
Let $\mathcal{A}_d$ *denote the set of all explanatory arguments that can be generated from* $\langle \mathcal{B}_b, \mathcal{B}_d \rangle$*, and* $\mathcal{A} = \mathcal{A}_d \cup \mathcal{A}_b$*.*

EXAMPLE 1. *Let* $waic \in \mathcal{K}$, $aic \in \mathcal{D}$*; waic denotes "there is a relevant workshop at the Sydney AI conference;" aic denotes "attend the Sydney AI conference." Suppose we have:*
$\mathcal{B}_b = \{(waic, 0.8)\}$
$\mathcal{B}_d = \{(waic \Rightarrow aic, 6)\}$
$\mathcal{B}_p = \emptyset$
$\mathcal{R} = \emptyset$
The agent can construct the explanatory argument A_1 *in favour of its desire to attend the Sydney AI conference:*
B_1*:* $\langle \{waic\}, waic \rangle$
A_1*:* $B_1 \Rightarrow aic$
with $\texttt{BELIEFS}(A_1) = \{waic\}$, $\texttt{DESIRES}(A_1) = \{aic\}$, $\texttt{CONC}(A_1) = \{aic\}$.

Note that the above example involves a desire-generation rule that contains beliefs only in its body. The following extended example shows how a desire can follow from another, already generated desire.

EXAMPLE 2. *Extending example 1, let: keynote denote "interesting key note speech"; attendkey denote "attend the key note speech". Suppose we have the following additional desire-generation rule, which states that if there is an interesting keynote speech at a conference I already desire to attend, then I would also desire to attend that speech:* $(keynote \wedge aic \Rightarrow attendkey, 8)$*. Suppose also that the agent believes that there is an interesting key note speech. Thus, we have the following new bases:*
$\mathcal{B}_b = \{(waic, 0.8), (keynote, 0.7)\}$
$\mathcal{B}_d = \{(waic \Rightarrow aic, 6), (keynote \wedge aic \Rightarrow attendkey, 8)\}$
$\mathcal{B}_p = \emptyset$
$\mathcal{R} = \emptyset$.
The agent can construct the explanatory argument A_2 *for the desire to attend the keynote speech:* B_1*:* $\langle \{waic\}, waic \rangle$
B_2*:* $\langle \{keynote\}, keynote \rangle$
A_1*:* $B_1 \Rightarrow aic$
A_2*:* $B_2, A_1 \Rightarrow attendkey$
with $\texttt{BELIEFS}(A_1) = \{waic\}$, $\texttt{BELIEFS}(A_2) = \{waic, keynote\}$, $\texttt{DESIRES}(A_1) = \{aic\}$, $\texttt{DESIRES}(A_2) = \{aic, attendkey\}$, $\texttt{CONC}(A_1) = \{aic\}$ *and* $\texttt{CONC}(A_2) = \{attendkey\}$.

[4]Note that B_i and E_i are comma-separated argument labels, not a conjunction of formulae (as in desire generation rules).

As with belief arguments, explanatory arguments may have different forces. However, since explanatory arguments involve two kinds of information: *beliefs* and *desires*, their strengths depend on both the quality of beliefs (using the notion of certainty level) and the importance of the supported desire. Formally:

DEFINITION 13. **(The force of explanatory arguments)** *Let* $A \in \mathcal{A}_d$ *be an explanatory argument. The* force *of A is* $Force(A) = <Level(A), Weight(A)>$ *where:*

- $Level(A) = \min\{a_i : \varphi_i \in \texttt{BELIEFS}(A)$ *and* $(\varphi_i, a_i) \in \mathcal{B}_b\}$*. If* $\texttt{BELIEFS}(A) = \emptyset$ *then* $Level(A) = 1$*;*
- $Weight(A) = w_i$ *such that* $(TOP(A), w_i) \in \mathcal{B}_d$*.*

In order to avoid any kind of wishful thinking, belief arguments are supposed to take precedence over explanatory ones. Formally:

DEFINITION 14. **(Comparing mixed arguments)** $\forall A_1 \in \mathcal{A}_b$ *and* $\forall A_2 \in \mathcal{A}_d$*, it holds that* A_1 *is* preferred *to* A_2*, denoted* $A_1 \succeq_d A_2$*.*

Concerning explanatory arguments, one may prefer an argument which will, for sure, justify an important desire. This suggests the use of a conjunctive combination of the certainty level of the argument and its weight. However, a simple conjunctive combination is open to discussion since it gives an equal weight to the importance of the desire and to the certainty of the set of beliefs that establishes that the desire takes place. Indeed, since beliefs verify the validity and the feasibility of desires, it is important that beliefs take precedence over the desires. This is translated by the fact that the certainty level of the argument is more important than the priority of the desire. Formally:

DEFINITION 15. **(Comparing explanatory arguments)** *Let* $A_1, A_2 \in \mathcal{A}_d$*.* A_1 *is* preferred *to* A_2*, denoted by* $A_1 \succeq_d A_2$*, iff*

- $Level(A_1) > Level(A_2)$*, or*
- $Level(A_1) = Level(A_2)$ *and* $Weight(A_1) > Weight(A_2)$*.*

An explanatory argument for some desire can be defeated either by a belief argument (which undermines the truth of the underlying belief justification), or by another explanatory argument (which undermines one of the existing desires the new desire is based on). Figure 2 summaries this notion of attack.

DEFINITION 16. **(Attack among Explanatory and Belief Arguments)**
Let $A_1, A_2 \in \mathcal{A}_d$ *and* $A_3 \in \mathcal{A}_b$*.*

- A_3 b-undercuts A_2 *iff* $\exists h' \in \texttt{BELIEFS}(A_2)$ *such that* $\texttt{CONC}(A_3) = \equiv \neg h'$*;*
- A_1 d-undercuts A_2 *iff* $\exists h' \in \texttt{DESIRES}(A_2)$ *such that* $\texttt{CONC}(A_1) \equiv \neg h'$*;*
- *An argument* $A' \in \mathcal{A}$ attacks_d $A_2 \in \mathcal{A}_d$ *iff* A' *b-undercuts or d-undercuts* A_2 *and not* $(A_2 \succeq_d A')$*.*

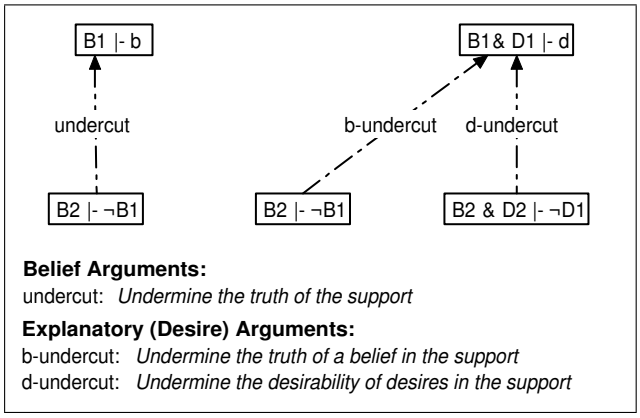


Figure 2: Summary of attacks involving belief and explanatory arguments

The following example illustrates the above concepts.

EXAMPLE 3. *(Builds on example 1) The agent finds out that the workshop has been cancelled (wcancel). That agent does not desire to go to the AI conference if it is not of international standing (int). Unfortunately the Sydney AI conference is not a good one. So the new bases are:*
$\mathcal{B}_b = \{(waic, 0.8), (wcancel, 1), (wcancel \rightarrow \neg waic, 0.8), (\neg int, 1)\}$
$\mathcal{B}_d = \{(waic \Rightarrow aic, 6), (\neg int \Rightarrow \neg aic, 9)\}$
$\mathcal{B}_p = \emptyset$
$\mathcal{R} = \emptyset$.
The following arguments can be built:
B_1: $\langle\{waic\}, waic\rangle$
B_2: $\langle\{wcancel, wcancel \rightarrow \neg waic\}, \neg waic\rangle$
B_3: $\langle\{\neg int\}, \neg int\rangle$
A_1: $B_1 \Rightarrow aic$
A_2: $B_3 \Rightarrow \neg aic$
It is clear that the argument B_2 b-undercuts the argument A_1 since $waic \in$ BELIEFS(A_1) and CONC$(B_2) = \neg waic$. *The argument A_2 d-undercuts the argument A_1 since* CONC$(A_2) = \neg aic$ *and $aic \in$* DESIRES(A_1).

Now that we have defined the notions of argument and defeasibility relation, we are ready to define the argumentation framework that should return the justified/valid desires.

DEFINITION 17. **(Argumentation framework)** *An* argumentation framework *AF_d for handling explanatory arguments is a tuple $AF_d = \langle \mathcal{A}_b, \mathcal{A}_d, Attack_b, Attack_d\rangle$ where $\mathcal{A}_b$ is the set of belief arguments, $\mathcal{A}_d$ the set of explanatory arguments, and $attack_d$ is the defeasibility relation between arguments in $\mathcal{A}$ and $attack_b$ is the defeasibility relation between arguments in $\mathcal{A}_b$.*

The definition of acceptable explanatory arguments is based on the notion of defence. Unlike belief arguments, an explanatory argument can be defended by either a belief argument or an explanatory argument. Formally:

DEFINITION 18. **(Defence among Explanatory and Belief Arguments)**
Let $S \subseteq \mathcal{A}$ and $A \in \mathcal{A}$. S defends *A iff $\forall A' \in \mathcal{A}$ where A' $attacks_b$ (or $attacks_d$) A, there is some argument $A'' \in S$ which $attacks_b$ (or $attacks_d$) A'.*

$\mathcal{F}'$ is a function such that $\mathcal{F}'(S) = \{A \in \mathcal{A}$ such that S defends $A\}$.

One can show easily that the function $\mathcal{F}$ is monotonic. Thus, it admits a least fixpoint. This last captures the acceptable arguments of AF_d.

PROPOSITION 2. *Let $AF_d = \langle \mathcal{A}_b, \mathcal{A}_d, Attack_b, Attack_d\rangle$ be an argumentation framework. The set $Acc(\mathcal{A}_d)$ of* acceptable explanatory arguments *is defined as*

$$Acc(\mathcal{A}_d) = (\bigcup \mathcal{F}'_{i\geq 0}(\emptyset)) \cap \mathcal{A}_d$$

PROOF. Due to the use of propositional language and finite bases, the argumentation system is finitary, i.e each argument is attacked by a finite number of arguments. Since the argumentation system is finitary then the function $\mathcal{F}'$ is continuous. Consequently, the least fixpoint of $\mathcal{F}'$ is $\bigcup \mathcal{F}'_{i\geq 0}(\emptyset)$. □

One can show that the above argumentation framework captures the results of the first framework which handles belief arguments.

PROPOSITION 3. *Let $AF_d = \langle \mathcal{A}_b, \mathcal{A}_d, Attack_b, Attack_d\rangle$ be an argumentation framework. $\bigcup \mathcal{F}'_{i\geq 0}(\emptyset) = Acc(\mathcal{A}_b) \cup Acc(\mathcal{A}_d)$*

PROOF. This follows directly from the definitions of $\mathcal{F}$ and $\mathcal{F}'$, and the fact that belief arguments are not attacked by explanatory arguments since we suppose that belief arguments are preferred to explanatory ones. □

DEFINITION 19. **(Justified desire)** *A desire ψ is* justified *iff $\exists$ $A \in \mathcal{A}_d$ such that* CONC$(A) = \psi$, *and $A \in Acc(\mathcal{A}_d)$.*

Desires supported by acceptable explanatory arguments are justified and hence the agent will pursue them (if they are achievable).

3.3 Arguing over plans

In the previous section, we have presented a framework for arguing about desires and producing a set of *justified* desires. In what follows we will show, among these justified desires, which ones will be pursued and with which plan.
The basic building block of a plan is the notion of "partial plan," which corresponds to a planning rule.

DEFINITION 20. **(Partial Plan)** *A* partial plan *is a pair $[H, \varphi]$ where*

- *$\varphi \in \mathcal{R}$ and $H = \emptyset$, or*
- *$\varphi \in \mathcal{D}$ and $H = \{\varphi_1, \ldots, \varphi_n, r_1 \ldots, r_m\}$ such that $\exists \varphi_1 \wedge \cdots \wedge \varphi_n \wedge r_1 \cdots \wedge r_m \rightarrowtail \varphi \in \mathcal{B}_p$.*

A partial plan $[H, \varphi]$ is elementary *iff $H = \emptyset$.*

DEFINITION 21. **(Instrumental Argument, or Complete Plan)** *An* instrumental argument *is a pair $\langle G, d\rangle$ such that $d \in \mathcal{D}$, and G is a finite tree such that:*

- *the root of the tree is a partial plan $[H, d]$;*
- *a node $[\{\varphi_1, \ldots, \varphi_n, r_1 \ldots, r_m\}, h']$ has exactly $n + m$ children $[H'_1, \varphi_1], \ldots [H'_n, \varphi_n]$, $[\emptyset, r_1], \ldots [\emptyset, r_m]$ where each $[H'_i, \varphi_i]$, $[\emptyset, r_k]$ is a partial plan;*
- *the leaves of the tree are elementary partial plans.*

Nodes(G) *is a function which returns the set of all partial plans of tree* G, *Des*(G) *is a function which returns the set of desires that plan* G *achieves, and Resources*(G) *is a function which returns the set of all resources needed to execute* G.

Let $\mathcal{A}_p$ *denotes the set of all instrumental arguments that can be built from agent's bases.*

An instrumental argument may achieve one or several desires of different worths with a certain cost. So the strength of that argument is the "benefit" or "utility" which is the difference between the worths of the desires and the cost of the plan. Formally:

DEFINITION 22. **(Strength of Instrumental Arguments)** *Let* $A = \langle G, g\rangle$ *be an instrumental argument. The* utility *of* A *is*

$$\texttt{Utility}(A) = \sum_{d_i \in Des(G)} Worth(d_i) - \sum_{r_j \in Resources(G)} Cost(r_j).$$

In [3], the strength of an instrumental argument is defined only on the basis of the weight of the corresponding desire. That definition does not account for the cost of executing the plan.

EXAMPLE 4. *A customer requires a car hire (a resource) in order to go to Sydney (a goal), which in turn achieves the agent's wish to attend an Artificial Intelligence conference (a desire). The customer desires to attend the AI conference because he believes it includes a workshop related to his research (a belief that justifies the desire). Let:*
aic = "attend the Sydney AI conference";
syd = "go to Sydney";
reg = "pay conference registration";
rent = "rent a car";
ford = "get a particular car of make Ford";
pay\$100 = "pay \$100";
pay\$200 = "pay \$200";[5]
We can now specify the following, for the buyer agent B *and seller agent* S:

1. $\mathcal{B}_b^B = \{(waic, 1)\}$
2. $\mathcal{B}_d^B = \{(waic \Rightarrow aic, 6)\}$
3. $\mathcal{B}_p^B = \begin{cases} syd \wedge reg \rightarrowtail aic \\ rent \rightarrowtail syd \\ ford \wedge pay\$200 \rightarrowtail rent \\ pay\$100 \rightarrowtail reg \end{cases}$
4. $RES = \{pay\$100, pay\$200, ford\}$
5. $\mathcal{R}^B = \{pay\$100, pay\$200\}$
6. $\mathcal{R}^S = \{ford\}$

Figure 3 shows an instrumental argument, for attending the Sydney AI conference, that agent B *can construct using the above information. Note that this plan involves the execution of action ford by agent* S, *because* B *does not have "ford" as one of its resources. Without getting the car from* S, B *cannot make it to Sydney using this plan.*

[5]Realistically, one requires a more elaborate treatment of actions, e.g. the agent must also be able to pay \$300, or pay \$50 six times. For simplicity, we suffice with these illustrative unique actions.

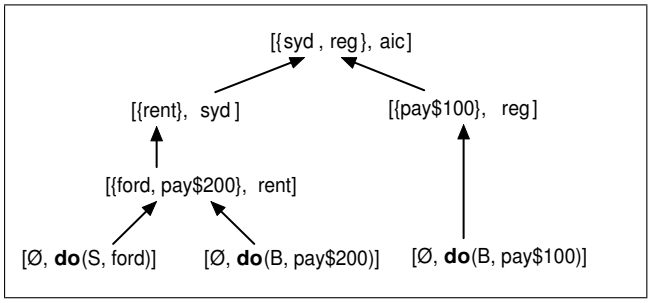


Figure 3: Complete plan for example 4

In [1], it has been shown that there are four great families of conflicts between partial plans. In fact, two partial plans $[H_1, h_1]$ and $[H_2, h_2]$ may be conflicting for one of the following reasons:

- *desire-desire* conflict, *ie* $\{h_1\} \cup \{h_2\} \vdash \bot$
- *plan-plan* conflict, *ie* $H_1 \cup H_2 \vdash \bot$.
- *consequence-consequence* conflict, *ie* the consequences of achieving the two desires h_1 and h_2 are conflicting.
- *plan-consequence* conflict, *ie* the plan H_1 conflicts with the consequences of achieving h_2.

The above conflicts are captured when defining the notion of *conflict-free* sets of instrumental arguments.

DEFINITION 23. **(Conflict-free sets of instrumental arguments)** *Let* $S \subseteq \mathcal{A}_p$. S *is* conflict-free, *with respect to the agent's beliefs* $\mathcal{B}_b^*$, *iff* $\nexists\ \mathcal{B}' \subseteq \mathcal{B}_b^*$ *such that:*

1. $\mathcal{B}'$ *is consistent, and*
2. $\bigcup_{\langle G,d\rangle \in S}[\bigcup_{[H,h]\in Nodes(G)}(H \cup \{h\})] \cup \mathcal{B}' \vdash \bot$

As with belief and explanatory arguments, we now present the notion of an *acceptable set of instrumental arguments.*

DEFINITION 24. **(Acceptable Set of Instrumental Arguments)** *Let* $S \subseteq \mathcal{A}_p$. S *is* acceptable *iff:*

- S *is conflict-free.*
- S *is maximal for set inclusion among the sets verifying the above condition.*

Let $S_1, \ldots, S_n$ *be the different acceptable sets of instrumental arguments.*

DEFINITION 25. **(Achievable desire)** *Let* $S_1, \ldots, S_n$ *be the different acceptable sets of instrumental arguments. A desire* ψ *is* achievable *iff* $\exists S' \in \{S_1, \ldots, S_n\}$, *such that* $\langle G, \psi\rangle \in S'$

DEFINITION 26. **(Utility of Set of Instrumental Arguments)** *For an acceptable set of instrumental arguments* $S = \{\langle G_1, d_1\rangle, \ldots, \langle G_m, d_m\rangle\}$, *the set of all desires achieved by* S *and all resources consumed by* S *as follows:*
$DE(S) = \{g_l : g_l \in Des(G_k), l = 1, \ldots, h, k = 1, \ldots, m\}$
$RE(S) = \{r_l : r_l \in Res(G_k), l = 1, \ldots, h, k = 1, \ldots, m\}$
The utility of a set of arguments S *is:*

$$\texttt{Utility}(S) = \sum_{g_i \in DE(S)} Worth(g_i) - \sum_{r_j \in Resources(S)} Cost(r_j).$$

We can now construct a complete pre-ordering on the set $\{S_1, \ldots, S_n\}$ of acceptable sets of instrumental arguments. The basic idea is to prefer the set with a maximal total utility: a maximal set of consistent plans.

DEFINITION 27. **(Preferred set)**
Let $S_1, \ldots, S_n$ be the acceptable sets of instrumental arguments. S_i is preferred *to S_j iff* $\texttt{Utility}(S_i) \geq \texttt{Utility}(S_j)$

Note that the above definition allows for cases where a set with a single desire/plan pair is preferred to another set with two or more desire/plan pairs (because the utility achieved by this desire is higher than the other two). This is more flexible than the frameworks of Amgoud and of Hustijn and van der Torre [1, 9], where sets with maximal *number* of desires are privileged, with no regard to their priority or the cost of different plans.

In order to be pursued, a desire should be both justified (i.e supported by an acceptable explanatory argument) and also achievable. Such desires will form the intentions of the agent.

DEFINITION 28. **(Intention set)**
Let $T \subseteq \mathcal{PD}$. T is an intention set iff:

1. *$\forall d_i \in T$, d_i is justified and achievable.*
2. *$\exists S_l \in \{S_1, \ldots, S_n\}$ such that $\forall d_i \in T$, $\exists \langle G_i, d_i \rangle \in S_l$.*
3. *$\forall\ S_k \neq S_l$ with S_k satisfying condition 2, then S_l is preferred to S_k.*
4. *T is maximal for set inclusion among the subsets of $\mathcal{PD}$ satisfying the above conditions.*

The second condition ensures that the desires are achievable together. If there is more than one intention set, a single one must be selected (e.g. at random) to become the agent's *intention*. The chosen set is denoted by $\mathcal{I}$. Finally, the *intended resources*, denoted $\mathcal{IR} \subseteq RES$ denote the resources needed by plans in S_l for achieving $\mathcal{I}$. The example below, depicted in Figure 4, puts the above concepts together.

EXAMPLE 5. *(Extends example 4) Suppose the buyer also would like to go on holiday to New Zealand and must reason with a limited budget. Let:*
nz = "take a holiday in New Zealand";
flynz = "fly to New Zealand";
hotel = "book a hotel accommodation";
friend = "stay at a friend's place";
call = "call a friend";
Suppose the agent has the following new desire generation knowledge base: $\mathcal{B}_d^B = \{(waic \Rightarrow aic, 0.6), \Rightarrow nz, 0.5)\}$ and that desires aic and nz are justified.

Finally, suppose costs are assigned as follows: $Cost(pay\$200) = 0.2$, $Cost(pay\$100) = 0.1$, $Cost(pay\$200) = 0.2$, $Cost(call) = 0$, $Cost(ford) = 0$).[6]

Suppose the buyer has two instrumental arguments for going to New Zealand: one requires booking a hotel (and paying \$200), while the other involves calling a friend to arrange a stay at his place. There are no conflicts between the arguments A_1, A_2 and A_3. Thus, there exists a unique acceptable set of instrumental arguments $\{A_1, A_2, A_3\}$. Since the desires aic and nz are supposed justifies, then there is a unique intention set $I = \{aic, nz\}$.

[6]The cost of "*ford*" to the buyer is zero because this resource is possessed by the seller and hence would only incur a cost to the seller.

4. RELATED WORKS

Recently, a number of attempts have been made to use formal models of argumentation as a basis for practical reasoning. Some of these models (e.g. [1, 3, 9]) are instantiations of the *abstract* argumentation framework of Dung [6], and our work is a contribution to this approach. Other approaches are based on an encoding of argumentative reasoning in logic programs (e.g. [10, 14]) or on completely new theories of practical reasoning and persuasion (e.g. [4, 15]).

Amgoud [1] presented an argumentation framework for generating consistent plans from a given set of desires and planning rules. This was later extended with argumentation frameworks that generate the desires themselves (see below).

Amgoud and Kaci [3] have a notion of "conditional rule," which is meant to generate desires from beliefs. Our desire generation rules are more general. In particular, we allow the generation of desires not only from beliefs, but also on the basis of other desires. Hence, our desire generation rules are more general.

Another problem arises because Amgoud and Kaci's definition does not distinguish between desires and beliefs in the antecedent and consequent of these rules. This may lead to incorrect inferences where an agent may conclude beliefs on the basis of yet-unachieved desires, hence exhibiting a form of wishful thinking. Our approach resolves this by distinguishing between beliefs and desires in the rule antecedents, allowing desires only in the consequent, and refining the notion of attack among explanatory arguments accordingly.

Hulstijn and van der Torre [9], on the other hand, have a notion of "desire rule," which contains only desires in the consequent. But their approach is still problematic. It requires that the selected goals[7] are supported by goal trees[8] which contain both desire rules and belief rules that are deductively consistent. This consistent deductive closure again does not distinguish between desire literals and belief literals (see Proposition 2 in [9]). This means that one cannot both believe $\neg p$ and desire p. In our framework, on the other hand, the distinction enables us to have an acceptable belief argument for believing $\neg p$ and, at the same time, an acceptable explanatory argument for desiring p.

Another advantage of our framework is that it derives preferences among explanatory and instrumental arguments using both worth and cost measures. This contrasts with Amgoud's and Hulstijn and van der Torre's frameworks, which privilege extensions with maximal number of desires without regard to desire priorities and resource cost. And while [3] does incorporate the weight of desires when calculating the strength of an instrumental argument, the cost of executing plans is not taken into account.

5. CONCLUSIONS

We presented a formal model for reasoning about desires (generating desires and plans for achieving them) based on argumentation theory. We adapted the notions of attack and preference among arguments in order to capture the differences in arguing about beliefs, desires and plans. We incorporated both the worth of desires and cost of resources in order to produce intentions that maximise utility.

One of the main advantages of our framework is that, being grounded in argumentation, it lends itself naturally

[7]Similar to our justified desires
[8]Similar to our explanatory arguments.

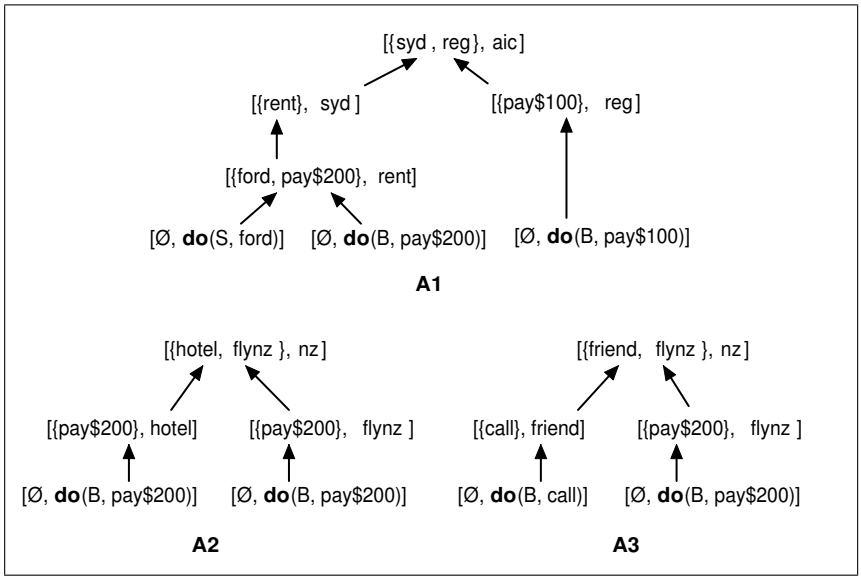


Figure 4: Plans for example 5

to facilitating dialogues about desires and plans. Indeed, we are currently extending our framework with dialogue game protocols in order to facilitate negotiation and persuasion among agents. Another interesting area of future work is investigating the relationship between our framework and axiomatic approaches to BDI agents.

6. ACKNOWLEDGMENTS

The authors thank Liz Sonenberg, Frank Dignum and Rogier van Eijk for discussions surrounding the paper topic.

7. REFERENCES

[1] L. Amgoud. A formal framework for handling conflicting desires. In T. D. Nielsen and N. L. Zhang, editors, *Proc. ECSQARU*, volume 2711 of *LNCS*, pages 552–563. Springer, Germany, 2003.

[2] L. Amgoud and C. Cayrol. A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence*, 34(1–3):197–215, 2002.

[3] L. Amgoud and S. Kaci. On the generation of bipolar goals in argumentation-based negotiation. In I. Rahwan et al, editor, *Proc. 1st Int. Workshop on Argumentation in Multi-Agent Systems (ArgMAS)*, volume 3366 of *LNCS*. Springer, Germany, 2005.

[4] K. Atkinson, T. Bench-Capon, and P. McBurney. Justifying practical reasoning. In C. R. F. Grasso and G. Carenini, editors, *Proc. Workshop on Computational Models of Natural Argument (CMNA)*, pages 87–90, 2004.

[5] R. H. Bordini and J. F. Hübner. Jason: A Java-based AgentSpeak interpreter used with saci for multi-agent distribution over the net, 2005. http://jason.sourceforge.net/.

[6] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.

[7] P. Gărdenfors, editor. *Belief Revision*. Cambridge University Press, 1992.

[8] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Meyer. Agent programming in 3apl. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

[9] J. Hulstijn and L. van der Torre. Combining goal generation and planning in an argumentation framework. In A. Hunter and J. Lang, editors, *Proc. Workshop on Argument, Dialogue and Decision, at NMR*, Whistler, Canada, June 2004.

[10] A. Kakas and P. Moraitis. Argumentation based decision making for autonomous agents. In *Proc. 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 883–890, Melbourne, Australia, 2003.

[11] H. Prakken and G. Sartor. Argument-based logic programming with defeasible priorities. *Journal of Applied Non-classical Logics*, 7:25–75, 1997.

[12] H. Prakken and G. Vreeswijk. Logics for defeasible argumentation. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 4, pages 219–318. Kluwer, Netherlands, 2002.

[13] A. S. Rao and M. P. Georgeff. Decision procedures for BDI logics. *Journal of Logic and Computation*, 8(3):293–342, June 1998.

[14] G. R. Simari, A. J. Garcia, and M. Capobianco. Actions, planning and defeasible reasoning. In *Proc. 10th International Workshop on Non-Monotonic Reasoning*, pages 377–384, 2004.

[15] Y. Tang and S. Parsons. Argumentation-based dialogues for deliberation. In F. Dignum et al, editor, *Proc. AAMAS, Utrecht, The Netherlands*, pages 552–559, New York NY, USA, 2005. ACM Press.

Predicting partner's behaviour in agent negotiation

Jakub Brzostowski *
Faculty of Information and Communication Technologies
Swinburne University of Technology
Hawthorn, Victoria 3122, Australia
jbrzostowski@it.swin.edu.au

Ryszard Kowalczyk
Faculty of Information and Communication Technologies
Swinburne University of Technology
Hawthorn, Victoria 3122, Australia
rkowalczyk@it.swin.edu.au

ABSTRACT

We propose an improved approach for modeling behaviours of negotiation partners and predictive decision-making based on this modelling. Our prediction is based only on the history of the offers during the current negotiation. The mechanism estimates an influence of different factors contributing to partner's behaviour during negotiation and uses this information to construct a prediction about agent's future behaviour. The optimal sequence of offers is determined according to the prediction. The approach is tested in simple scenarios and the results comparing our approach to random strategy selection are illustrated.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Learning, Problem Solving

General Terms

Algorithms, Theory

Keywords

negotiation and conflict handling in agent systems

1. INTRODUCTION

Negotiation is a process of searching a space of potential agreements in order to find a solution that satisfies the preferences and requirements of all interested parties [5]. The problem of negotiation has been studied from various perspectives in many fields. These fields include: decision and game theory, management and social sciences, artificial intelligence and agent technology [9]. The essential problems of the negotiating parties are the limited or uncertain knowledge and conflicting preferences. The agent technology provides tools for solving this kind of problems. However, decision-making paradigms of the negotiation agents still need to be developed and improved. A successful agent needs to learn and adapt to complex behaviour of its negotiation partner and changing environment. An approach proposed by Faratin [4] allows for limited adaptation to partners behaviour by using so called behaviour-dependant tactics. These tactics imitate opponent's behaviour and can be mixed with other types of tactic such as time-dependant. However, it is difficult to decide which type of tactics to use and to what extent in terms of weights mixing various factors such as: time and behaviour imitation. Therefore, some approaches based on machine learning have been proposed to predict the negotiation partner's behaviour.

Most of the learning approaches supporting the negotiation such as Bayesian learning [12], Q-learning [3], case-based-reasoning [1] and evolutionary computation [11][10] focus on learning from previous negotiations. Some work [7][6] studies the negotiation scenarios by taking into account the parameters of negotiation such as: deadline and reservation value but assuming a simplified type of negotiation strategies, namely only time-dependant. The work proposing learning from current negotiation is quite rare. One of the approaches was proposed by Hou [8]. In this paper a nonlinear regression was proposed to model the negotiation opponent's behaviour based only on the previous offers. However, the approach copes only with single tactics but negotiation agent may encounter an opponent with more complex behaviour involving a mixture of different tactics.

In this paper we propose to treat the negotiation as an optimization problem because too small concessions may discourage the partner from making significant concessions and to large concession will obviously lead to a low level of utility at the end of an encounter. Our predictive decision-making mechanism allows for high adaptation to negotiation partner's behaviour based only on the history of offers in the current negotiation.

It extends the predictive decision-making mechanism described in [2]. The approach is based on difference method and is therefore computationally efficient. In the future work we will propose other prediction approach based on nonlinear multivariate regression analysis. Our extended approach allows for additional gain of efficiency compared to the previous paper [2] because instead of a brute force search of the optimal sequence of actions we propose a more efficient search based on a solution of optimization equation that reduces significantly the time of modelling.

The second section recalls an idea of estimation of different factors influencing negotiation partner's behaviour. The

*The first author of this paper is a student

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8–12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

third section presents a method of constructing a prediction. In the fourth section an idea of assessing the whole sequence of offers is described. In the fifth section we describe a way of calculating values of concession. Sixth section presents the results of experiments. Conclusions and future work are presented in the 6 section.

2. THE ESTIMATION OF DEGREES OF VARIOUS FACTORS DEPENDENCY

Typically, there are two main factors that influence the negotiation agent behaviour: time and imitation. The agents behaviour depends to some extent on time and it depends to some extent on imitation. In order to be able to assess the degrees of this dependency appropriate criteria will be introduced in a form of functions mapping the sequences of previous actions of both parties in the current encounter into interval $[0, 1]$. These two criteria should state to what extent our partner responds to its time constraint and to what extent does it respond to our actions. We propose to define these criteria as necessary conditions for using pure tactics by our negotiation partner (either behaviour-dependant or time-dependant). In order to do it we have to know what is characteristic for an agent using time-dependant tactic (necessary condition) or behaviour-dependant tactic. This feature will be used as the factor dependency criterion.

2.1 Time-dependency

The time-dependant tactic may be modelled by the use of polynomial or exponential function. As mentioned before we have to determine the characteristic feature of these type of functions. The whole family of polynomial or exponential functions has a constant sign of all derivatives and this feature is treated as a necessary condition for being time-dependant. The given information is the sequence of previous offers of both parties. Based on it we want to determine the extent to which the sequence of our partner's offers is generated by a particular decision function corresponding to a pure tactic. In other words we want to determine how good the partner's behaviour fits to a particular decision function. Assuming that a decision function used to generate a sequence of offers had a constant sign of all its derivatives the differences of all orders calculated for the given sequence must also have a constant sign. This is because of a correspondence of the difference of n-th order and the n-th derivative. Having a sequence of partner offers: b_i of a length m the differences of different orders can be calculated as follows:

$$\begin{aligned}
\Delta^1 b_i &= b_{i+1} - b_i \qquad i \in \{1, 2, \ldots, m-1\} \\
\Delta^2 b_i &= \Delta b_{i+1} - \Delta b_i \qquad i \in \{1, 2, \ldots, m-2\} \\
\Delta^3 b_i &= \Delta^2 b_{i+1} - \Delta^2 b_i \qquad i \in \{1, 2, \ldots, m-3\} \\
&\ldots \\
\Delta^{m-1} b_i &= \Delta^{m-2} b_{i+1} - \Delta^{m-2} b_i \qquad i \in \{1\}
\end{aligned}$$

If all the differences for each particular order k have the same sign then the necessary condition of being time-dependant is fully satisifed:

$$\forall k \in \{1, 2, \ldots, m-1\} \quad \forall i, j \in \{1, 2, \ldots, m-k\} \quad sgn(\Delta^k b_i) = sgn(\Delta^k b_j)$$

In order to evaluate a degree to which time-dependency is satisfied we formulate a criterion. For the difference of each order k we introduce a subcriterion D^k. The subcriterion must be defined in such a way that it gives full level of satisfaction if all the differences are either negative or positive. If the means of the positive and negative values counterbalance each other then the criterion should give the full level of violation what corresponds to the lowest level of satisfaction. In other words the full violation means that the contributions of positive and negative values are the same. Partial degree of criterion satisfaction will be obtained if we observe a dominance of one sign over other and the higher is this dominance the higher the degree of satisfaction. Therefore, we introduce the partial degree of subcriterion satisfaction D^k of the difference of an order k as an aggregate of the means of the positive values of differences and the negative values of the differences as follows:

$$D_k = \frac{|\frac{\sigma_p^k}{m_p^k} + \frac{\sigma_n^k}{m_n^k}|}{Max\{|\frac{\sigma_p^k}{m_p^k}|, |\frac{\sigma_n^k}{m_n^k}|\}} \tag{1}$$

where $\sigma_p^k = \sum_{\Delta_k b_i > 0} \Delta_k b_i$, $\sigma_n^k = \sum_{\Delta_k b_i < 0} \Delta_k b_i$. m_p^k and m_n^k are the amounts of the positive and negative values of $\Delta^k b_i$, respectively. If the absolute values of the averages $|\frac{\sigma_p^k}{m_p^k}|$, $|\frac{\sigma_n^k}{m_n^k}|$ are close then the value of D^k is small, what means that the criterion is highly violated. The values of all given D^k are then aggregated to the overall value of criterion satisfaction. This value is obtained by the use of weighted sum where the weights are proportional to the extent of contribution of the differences $\Delta^k b_i$:

$$D_t = \sum_{k \leq m-1} w_k D_k$$

where w_k is calculated as follows:

$$w_k = \frac{Max\{|\sigma_p^k|, |\sigma_n^k|\}}{\sum_{j=1}^{m-1} Max\{|\sigma_p^j|, |\sigma_n^j|\}}$$

2.2 Behaviour-dependency

The responsiveness to partner's behaviour may be realized by behaviour-dependant tactic. We propose to use the sequence of relative first order difference as a basis for measuring the extent to which our partner is imitating our behaviour. Having the sequences of the last m concessions of our agent s_i and the sequence of the last m concessions b_i of the negotiation partner we can calculate the sequence $r_i = \frac{\Delta^1 b_i}{\Delta^1 s_i}$ describing the change of behaviour of the modelled agent in relation to the change of our agent's behaviour. We define two subcreteria allowing for the assessment of the imitative-behaviour of our negotiation partner. First subcreterion assesses how positive are the responses of the modelled agent to our offers (the values of r_i are higher then one). We aggregate all values of r_i by the use of weighted sum:

$$r = \sum_{i=1}^{m} w_i r_i$$

The weights w_i may increase with the value of index i which means that the later offers reflect better the current state of our partner's behaviour and therefore, the higher the values of indices i the more important is the quotient r_i. If the

calculated value of r is greater than 1 then our negotiation partner is responding with higher concessions than ours and therefore it may be called relatively cooperative. The value of r equal to 1 means that the agent makes the same concessions as ours and can therefore be called neutral. The values of r smaller than 1 tell us that our partner is not relatively cooperative because its concessions are smaller than ours. The first subcreterion may be defined as a function $h : r \mapsto [0, 1]$ mapping the value of r into a degree of responsiveness in the following way:

$$D_1 = h(r) = Max[1, Min[0, 0.5r]]$$

The neutral value of r ($r = 1$) is assigned the value of 0.5 by function h what in terms of fuzzy logic means that the extent of condition satisfaction and dissatisfaction is the same. If the value of r is greater than 1 then the degree of subcreterion satisfaction is higher than 0.5 (more true than false) and in the opposite case ($r < 1$) the degree of satisfaction is lower than 0.5 (more false than true).

Additional subcreterion can asses the degree of relational trend in the agent's behaviour in terms of monotonicity of sequence r_i. The degree of monotinicity D_2 may be measured as a sign consistency for the differences Δr_i what can be done analogously as in previous subsection (formula 1). We calculate the average D_b of the obtained values of D_1 and D_2 and treat this value as a degree of the overall responsiveness:

$$D_b = \frac{D_1 + D_2}{2}$$

3. THE AGENT'S BEHAVIOUR PREDICTION BASED ON DIFFERENCE METHOD

As mentioned in previous sections there are two main factors contributing to the overall behaviour of negotiation agent: time and imitation. We propose to construct separate predictions of these two factors first. Then prediction can be build by mixing the time-depending and the behaviour-depending predictions together using the weights obtained in the previous section. Our prediction constitutes a function mapping a history of the previous m offers of both negotiating parties H_m and a potential offer s_{m+1} of our agent into a value of the predicted response $\hat{b}_{m+1}$ of our partner:

$$f : (H_m, s_{m+1}) \rightarrow \hat{b}_{m+1}$$

As mentioned in Introduction in the future work we will conisder other type of prediction based on the multivariate regression analysis. The two predictions, namely time-depending f_t and imitation-depending f_b that will be combined together to the overall prediction f map the history of the current encounter to the value of the predicted offer in the following way:

$$\hat{b}^t_{m+1} = f_t(H_m) \quad (2)$$

$$\hat{b}^b_{m+1} = f_b(H_m, s_{m+1}) \quad (3)$$

As stated before these mappings are combined to obtain the overall prediction as follows:

$$\begin{aligned} \hat{b}_{m+1} &= f(H_m, s_{m+1}) = \\ &= \frac{D_t}{D_t + D_b} \times \hat{b}^t_{m+1} + \frac{D_b}{D_t + D_b} \times \hat{b}^b_{m+1} \end{aligned} \quad (4)$$

The predictor f may be applied multiple l times in a recursive way. In order to reuse the predictor second time we store the time-depending prediction $\hat{H}^t_{m+1} = H^t_m \cup \{s_{m+1}, \hat{b}^t_{m+1}\}$ ($H^t_m = H_m$). We also store separately the overall prediction $\hat{H}_{m+1} = H_m \cup \{s_{m+1}, \hat{b}_{m+1}\}$. Based on this information the prediction for the second step can be obtained as follows:

$$\hat{b}^t_{m+2} = f_t(\hat{H}^t_{m+1}) \quad (5)$$

$$\hat{b}^b_{m+2} = f_b(\hat{H}_{m+1}, s_{m+2}) \quad (6)$$

where s_{m+2} is a potential offer of our agent in the next step of negotiation. The overall second step prediction is calculated as follows:

$$\begin{aligned} \hat{b}_{m+2} &= f(\hat{H}_{m+1}, s_{m+2}) = \\ &= w_t \times \hat{b}^t_{m+2} + w_b \times \hat{b}^b_{m+2} \end{aligned} \quad (7)$$

where w_t and w_b are the normalized values of D_t and D_b:

$$w_t = \frac{D_t}{D_t + D_b} \qquad w_b = \frac{D_b}{D_t + D_b}$$

As stated before we can repeat this procedure multiple l times.

3.1 Time-depending prediction

The time-dependant behaviour may be realized by the use of decision functions from the polynomial or exponential family of functions. We used this information in the previous section to determine the degree of time-dependency and in this section we want to determine a level of the particular time-dependant tactic in order to predict this type of behaviour. We propose to measure a trend in agent's behaviour in terms of a degree of concavity or convexity of the concession curve. In other words we want to determine what a kind of tactics (conceder, linear or boulware) our partner is using, and how strong is the tactic in terms of a value of the parameter β. Our time-depending prediction is based on a difference method that requires to determine the maximal order of differences that should be taken into account. In a case of linear and conceder tactics it is sufficient to consider only the first order difference because for the curve generated by the linear tactic all higher order differences are equal to zero and the curve generated by the conceder tactic is close to linear after a number of offers. The most crucial is the successful prediction of the boulware tactic because the most drastic changes of the behaviour of an agent using this kind of tactic occur when it is approaching its deadline. All the differences up to k-th order having the same sign are taken into consideration as follows:

$$sgn(\Delta^1 b_i) = sgn(\Delta^2 b_i) = \cdots = sgn(\Delta^k b_i) \neq sgn(\Delta^{k+1} b_i)$$

The more differences are consistent in a sign the stronger boulware tactic on the side of opponent agent was applied. We calculate the time-dependent prediction for the next step as follows (Figure 1):

$$\hat{b}_{m+1} = f_t(H_m) = b_m + \sum_{j=1}^{k} \Delta^j b_{m-j} \quad (8)$$

where $\hat{b}_{m+1}$ is the prediction of the $m+1$ offer and k is an index of the last difference consistent in sign. It can be proved that the difference based method is a good approximation of any function from the polynomial family of functions. The

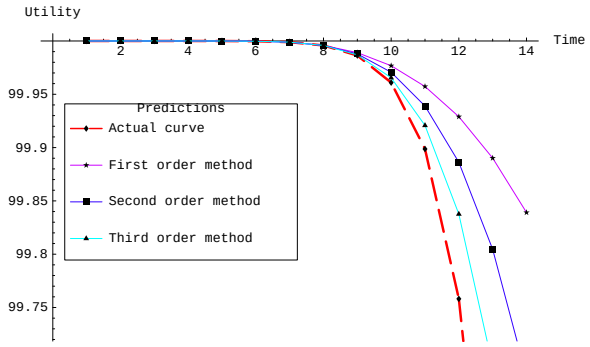


Figure 1: Examples of different order predictions for boulware tactic

recursive application of the formula (8) allows for the prediction in the next steps.

3.2 Imitation-depending prediction

The imitation-depending prediction constitutes a function that maps the history H_m of offers of the current encounter and a potential future offer s_{m+1} of our agent into the prediction about our opponent's next move $\hat{b}_{m+1}$:

$$\hat{b}_{m+1} = f_b(H_m, s_{m+1})$$

The purpose of the function f_b is to predict how the negotiation partner will respond to the potential offer s_{m+1} of our agent. In terms of control theory the history of the negotiation H_m up to an offer m describes the current state of the encounter. The variable s_{m+1} is a potential control with which we may influence partners behaviour. The updated history $\hat{H}_{m+1}$ after the step of control s_{m+1} is the next state of negotiation. In order to construct the function f_b we apply the Taylor's formula of first order as follows:

$$f_b(H_m, s_{m+1}) \quad = \quad b_m + \frac{\Delta b_{m-1}}{\Delta s_{m-1}}(s_{m+1} - s_m)$$

where the value of a relational difference $\frac{\Delta b_{m-1}}{\Delta s_{m-1}}$ is treated as an approximation of the first order derivative of the function f_b in a point s_m.

4. SEMI-STRATEGIC REASONING

The predictor described in previous sections can be regarded as a transition function that maps a state (history of previous offers) and a potential control (our potential offer) into the next state. In these terms the negotiation may be treated as a multi-stage control process and our task is to determine the sequence of optimal controls (our potential future offers) $s_{m+1}, s_{m+2}, \ldots, s_{m+l}$ knowing the sequence of our opponent's respones $\hat{b}_{m+1}, \hat{b}_{m+2}, \ldots, \hat{b}_{m+l}$. The multi-stage control process constitutes the combination of prediction with decision-making about the next offers. The mechanism described in [2] involves discretization of the space of each potential offer in each negotiation step. All possible sequences of offers in the discrete space are considered and the optimal sequence is determined. However, such an approach is a kind of brute force, and therefore is quite time consuming. We propose to determine a sequence of the optimal

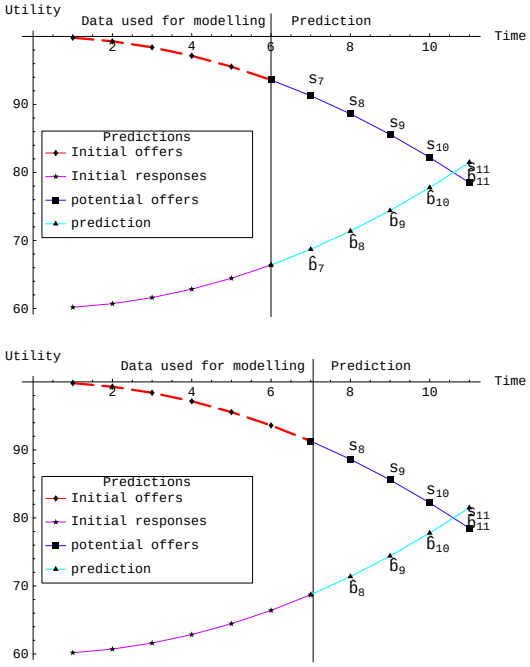


Figure 2: An example of prediction as a response to sequence of potential offers in steps 7 and 8

controls in a different way. In order to avoid the inefficiency of searching the whole space of solutions we propose to predict a small amount of k opponent's steps forward. Then we assume that we make the same average concession up to the terminal state l and determine this concession by solving optimization equation (9). In other words we want to determine the sequence of a form: $s_{m+1}, s_{m+2}, \ldots, s_{m+k}, s_{m+k} + c, s_{m+k} + 2c, \ldots, s_{m+k} + (l-k)c$. In order to determine the concession c of the second part of the sequence we solve the following optimization problem:

$$c : min\{|\hat{b}_{m+l} - (s_{m+k} + (l-k)c)|\} \qquad (9)$$

It states that we have to find a value of c for which we meet with the opponent (ia. reach an agreement) in terminal state $m+l$. After the counterpart has responded, the whole mechanism can be reused for further decision making in the next steps of negotiation.

5. CONCESSION DETERMINATION

The previous section described a way we asses a sequence of offers. Assuming that we determined the optimal sequence of concessions we have to determine the value of the next offer based on this information. It turns out that various sequences of offers may result (in terms of prediction) in similar final utility. Therefore, the decision-making process should take into account all sequences with various first concessions satisfying the optimization constraint (9). Assume we have all the optimal sequences $\{s^j\}$ with the various first

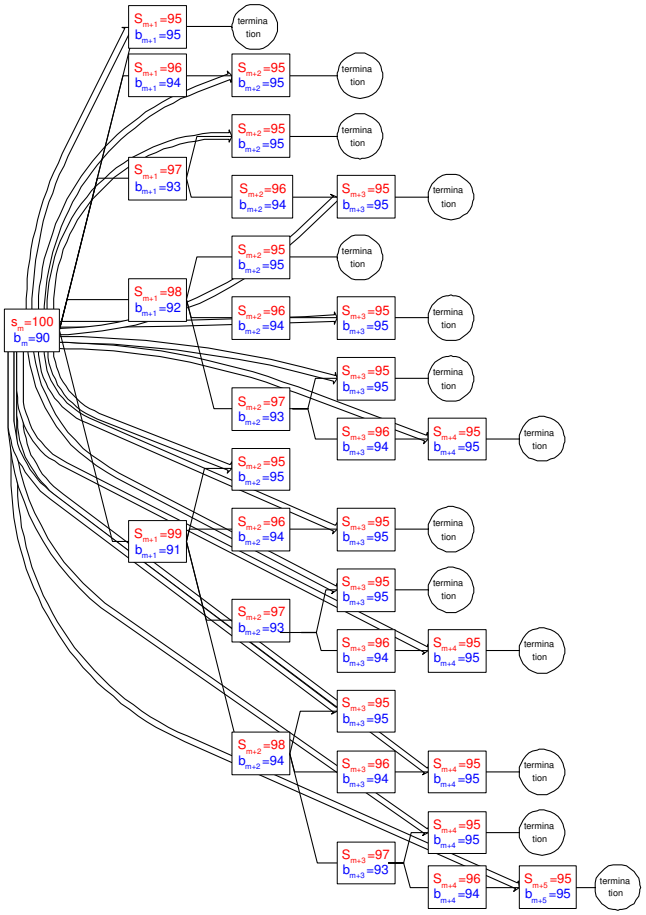


Figure 3: Simplified example of negotiation prediction. The thick arrows indicate the optimal sequences with various first concessions

concessions:

$$\begin{array}{ccc} s^1_{m+1} & , & s^1_{m+2}, \ldots, s^1_{m+l} \\ s^2_{m+1} & , & s^2_{m+2}, \ldots, s^2_{m+l} \\ \ldots & , & \ldots \\ s^n_{m+1} & , & s^n_{m+2}, \ldots, s^n_{m+l} \end{array}$$

To determine the value of a concession based on the predicted sequence we use two values: the first concession in the sequence $s^j_{m+1} - s^j_m$ and the average of all concessions in the sequence $\frac{s^j_{m+l} - s^j_m}{l}$. The less responsive an agent is the less important the first concession is. It means that in the case of time-dependant behaviour the opponent agent does not respond to our behaviour and it is not important what concession we make in what rounds. What is important is the average concession in all rounds. Therefore, the average concession should be taken into account in a case of time-dependant behaviour and the first concession should be taken into account in a case of the responsive agent. Because the extents to which the two factors influence agent's behaviour are fuzzy we have to aggregate the two types of concession taking into consideration the level of imitative behaviour and the level of time-depending behaviour. The experiments presented in the next section show that the following type of aggregation is appropriate:

$$\begin{aligned} offer^j &= s_m + sgn(s^j_{m+1} - s^j_m)(|s^j_{m+1} - s^j_m|)^{w_t} \times \\ &\times sgn\left(\frac{s^j_{m+l} - s^j_m}{l}\right)\left|\frac{s^j_{m+l} - s^j_m}{l}\right|^{w_b} \end{aligned}$$

The offers $offer^j$ are then averaged over all the optimal sequences with the various first concessions in order to calculate the final value of an offer:

$$offer = offer^1 w_1 + offer^2 w_2 + \cdots + offer^n w_n$$

where the weights w_j are proportional to the utility that can be gained by applying j-th sequence of actions $\{s^j\}$ according to our prediction.

6. RESULTS

The enhanced mechanism has been tested in a simulated experimental negotiations with more advanced settings than in [4] (the same settings as in [2]). On a side of our negotiation agent the predictive mechanism is used and the negotiation partner uses the static strategies being a mixture of a time-dependant tactic and a behaviour-dependant tactic with various weights assigned to these two tactics. In order to make experiments feasible we choose seven time-dependant tactics and divide them into three subsets:

- Conceder: $C = \{\beta \quad | \quad \beta \in \{2, 5, 8\}\}$
- Linear: $L = \{\beta \quad | \quad \beta \in \{1\}\}$
- Boulware: $B = \{\beta \quad | \quad \beta \in \{0, 3, 0.5, 0.7\}\}$

We consider two behaviour-dependant tactics:

- Absolute Tit-For-Tat: a: $\delta \in \{1\}$ and $R(M) = 0$ (without a random factor)
- Relative Tit-For-Tat: r: $\delta \in \{1\}$

The space of weights assigned to the tactics consists of 9 values obtained through discretization. Analogously as for the set of β values we decompose the set of all weights into three subsets:

- Small: $S = \{0.1, 0.2, 0.3\}$
- Medium: $M = \{0.4, 0.5, 0.6\}$
- Large: $L = \{0.7, 0.8, 0.9\}$

The cartesian product of the sets of the time-dependant tactics, the behaviour-dependant tactics and the weights results in a set of 126 static strategies:

$$ST = (C \cup L \cup B) \times \{a, r\} \times (S \cup M \cup L)$$

The whole set of the static strategies ST is divided into 18 groups of strategies in order to illustrate the performance of our approach:

$$\begin{aligned} ST = &(C \times \{a\} \times S) \cup (L \times \{a\} \times S) \cup (B \times \{a\} \times S) \cup \\ &\cup (C \times \{a\} \times M) \cup (L \times \{a\} \times M) \cup (B \times \{a\} \times M) \cup \\ &\cup (C \times \{a\} \times L) \cup (L \times \{a\} \times L) \cup (B \times \{a\} \times L) \cup \\ &\cup (C \times \{r\} \times S) \cup (L \times \{r\} \times S) \cup (B \times \{r\} \times S) \cup \\ &\cup (C \times \{r\} \times M) \cup (L \times \{r\} \times M) \cup (B \times \{r\} \times M) \cup \\ &\cup (C \times \{r\} \times L) \cup (L \times \{r\} \times L) \cup (B \times \{r\} \times L) \end{aligned} \quad (10)$$

Each group will be shortly denoted using only the first letters. For example:

$$CAS = C \times \{a\} \times S$$

A simple scenario involving a buyer agent and a seller agent

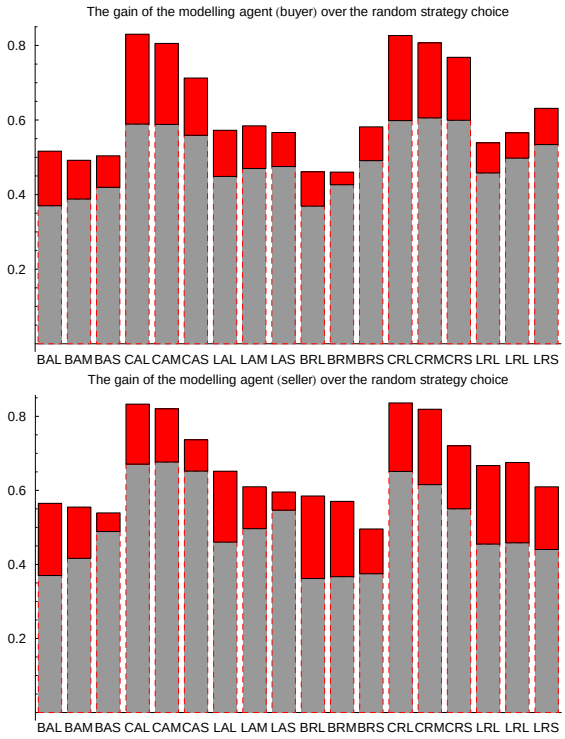


Figure 4: The performance of prediction mechanism applied for the buyer and the seller agents

is presented below. We consider following settings for the buyer: $min^b \in \{10\}$, $max^b \in \{50\}$, $k^b \in \{0\}$, and the consistent deadline for both players $t^b_{max} = 15$. The agreement ranges of both parties are assumed to be fully overlapping ($\Phi = 0$). Analogously as in [4], $max^b - min^b = max^s = min^s$ and $k^s = k^b$. $min^s = min^b + \Phi * (max^b - min^b)$, $max^s = min^s + (max^b - min^b)$. In the experiments we apply the prediction mechanism up to 9 steps forward, ie in the case of deadline $t^b_{max} = 15$ it makes prediction after the first six offers have been generated by the use of time-dependant tactic. This provides also initial data for the modelling. For the initial offers generation we use medium tough tactic ($\beta = 0.5$) with additional perturbations in order to better explore the initial responses of negotiation partner. We carry out the simulations without a communication cost and with low communication cost ($comm \in \{0.01\}$) separately (cost $\tau(t) = tanh(t * comm)$). More advanced scenarios considering different behaviour-dependant tactics and different values of communication cost will be described in further work. The performance of our enhanced modelling approach is illustrated in Figures (4) and (5). The grey parts of bars in the chart represent an average value of utility of a negotiation agent using all the strategies from particular subset of static strategies (for instance: CSA) against all the strategies from the set ST. This corresponds to playing with a random strategy from a particular subset against a random strategy selected from the whole ST according to an uni-

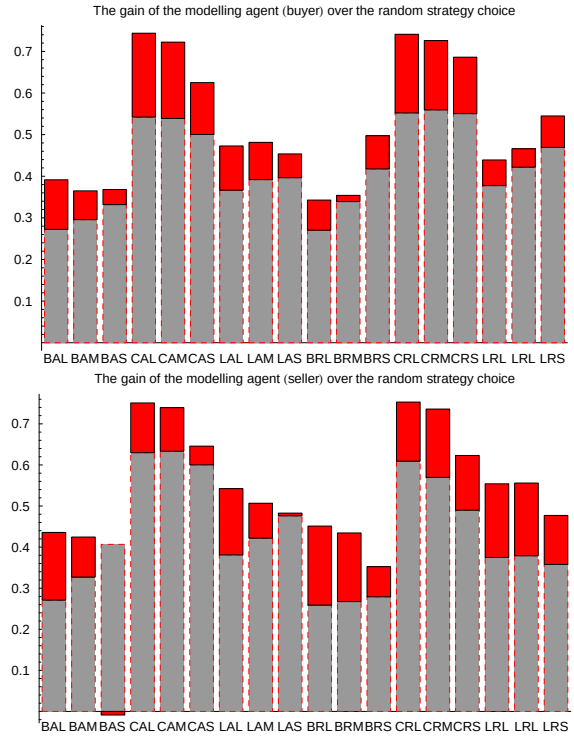


Figure 5: The performance of prediction mechanism taking into account the communication cost applied for the buyer and the seller agents

form probability distribution. The red (dark) surplus in the bar chart illustrates a gain in terms of the utility obtained by the use of our approach instead of the random static strategy choice. In cases of a gain the red (dark) surplus is above the grey part of a bar and in a case of loss it is under the grey part. The bar charts (Figure 4) clearly show that the gain of our predictive mechanism is significant, i.e. on average in comparison with the classical approach without prediction. Although, the reasoning was simplified in comparison to [2] the proposed mechanism still outperforms the random strategy choice and in terms of the gained utility the results are similar to [2]. It should be noted that in the cases of strategies with high contribution of behaviour-dependant tactics (BAS, BRS, etc.) the utility gain is smaller compared to other strategies which is caused by a specific nature of behaviour-dependant tactics. If an opponent uses the time-dependant tactic we can take advantage of it to finish negotiation earlier with a good level of utility. However, very high values of utilities are not possible because the imitation of our behaviour on the side of our partner means that roughly its concessions can not be much bigger than ours. As far as the communication cost is concerned the results are better than in [2]. The figure 5 shows that the utility gain is significant and positive in most of the cases. Another thing worth noticing is that the results are not symmetric for the buyer and for the seller because of the use of Relative Tit For Tat tactic that does not work symmetrically for both roles of an agent. The reason for it is that the value of quotient that is used in this tyoe of tactic is usually higher for the buyer than for the seller. Compared to [2] we obtained high gain in efficiency of the approach.

The simulation for the same scenario took just 20% of the time of the simulation presented in [2].

7. CONCLUSIONS AND FUTURE WORK

The paper proposes an improved approach for reasoning from the history of offers of current negotiation and decision-making based on this reasoning. The predictive mechanism allows an agent to adapt to negotiation partner's behaviour that results in better agreements. The presented results of simulated experiments show that our agent equipped with the enhanced predictive mechanism outperforms the classical approach by achieving substantial gains in terms of utility. In the future the regression analysis will be applied for partner's behaviour prediction and the approach will be tested in more complex scenarios including not consistent deadlines of negotiation parties, partial degrees of overlap of the ranges and different values of communication cost.

8. REFERENCES

[1] Jakub Brzostowski and Ryszard Kowalczyk. On possibilistic case-based reasoning for selecting partners in multi-agent negotiation. In Geoff Webb and Xinghuo Yu, editors, *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence*, volume 3339 of *Lecture Notes in Artificial Intelligence*, pages 694–705, Cairns, Australia, 2004. Springer.

[2] Jakub Brzostowski and Ryszard Kowalczyk. Modelling partner's behaviour in agent negotiation. In *Proceedings of the 18th Australian Joint Conference on Artificial Intelligence*, Lecture Notes in Artificial Intelligence, Sydney, Australia, 2005. Springer (in press).

[3] H. L. Cardoso and E. Oliveira. Using and evaluating adaptive agents for electronic commerce negotiation. In *IBERAMIA-SBIA '00: Proceedings of the International Joint Conference, 7th Ibero-American Conference on AI*, pages 96–105, London, UK, 2000. Springer-Verlag.

[4] P. Faratin. *Automated Service Negotiation Between Autonomous Computational Agents*. PhD thesis, University of Londodn, 2000.

[5] P. Faratin, C. Sierra, and N. R. Jennings. *Negotiation among groups of autonomous computational agents*. University of Londond, 1998.

[6] S. Fatima, M. Wooldridge, and N. R. Jennings. Comparing equilibria for game theoretic and evolutionary bargaining models. In *5th International Workshop on Agent-Mediated E-Commerce*, pages 70–77, 2003.

[7] S. S. Fatima, M. Wooldridge, and N. R. Jennings. Optimal negotiation strategies for agents with incomplete information. In *ATAL '01: Revised Papers from the 8th International Workshop on Intelligent Agents VIII*, pages 377–392, London, UK, 2002. Springer-Verlag.

[8] C. Hou. Modelling agents behaviour in automated negotiation. Technical Report KMI-TR-144, Knowledge Media Institute, The open University, Milton Keynes, UK, May 2004.

[9] Guoming Lai, Cuihong Li, Katia Sycara, and Joseph Andrew Giampapa. Literature review on multi-attribute negotiations. Technical Report CMU-RI-TR-04-66, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 2004.

[10] N. Matos, C. Sierra, and N. Jennings. Determining successful negotiation strategies: An evolutionary approach. In *ICMAS '98: Proceedings of the 3rd International Conference on Multi Agent Systems*, page 182, Washington, DC, USA, 1998. IEEE Computer Society.

[11] J. Oliver. A machine learning approach to automated negotiation and prospects for electronic commerce, 1997.

[12] D. Zeng and K. Sycara. Bayesian learning in negotiation. In Sandip Sen, editor, *Working Notes for the AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems*, pages 99–104, Stanford University, CA, USA, 1996.

Tractable Negotiation in Tree-structured Domains

Yann Chevaleyre[1], Ulle Endriss[2] and Nicolas Maudet[1]
[1]LAMSADE, Université Paris-Dauphine, France
[2]ILLC, University of Amsterdam, The Netherlands
[1]{chevaley,maudet}@lamsade.dauphine.fr, [2]ulle@illc.uva.nl

ABSTRACT

Multiagent resource allocation is a timely and exciting area of research at the interface of Computer Science and Economics. One of the main challenges in this area is the high complexity of negotiation. In particular, the complexity of the task of identifying rational deals, *i.e.* deals that are beneficial for all participants, often hinders the successful transfer of theoretical results to practical applications. To address this issue, we propose several protocols designed to tame the complexity of negotiation by exploiting structural properties of the utility functions used by agents to model their preferences over alternative bundles of resources. In particular, we consider domains where utility functions are *k-additive* (that is, synergies between different resources are restricted to bundles of at most k items) and *tree-structured* in the sense that the bundles for which there are synergies do not overlap. We show how protocols exploiting these properties can allow for drastically simplified negotiation processes.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*; J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Economics, Theory

Keywords

Multiagent resource allocation, Negotiation

1. INTRODUCTION

Multiagent resource allocation is a timely and exciting area of research at the interface of Computer Science and Economics. Applications range from electronic commerce [19], over logistics [12], to the exploitation of earth observation satellites [13]. In this paper, we are going to be concerned with a framework where agents negotiate over a finite number of indivisible resources in a distributed manner [9, 18]. Agents are assumed to only negotiate deals that are beneficial for each of the parties involved (which may be more than two). We are interested in negotiation histories that lead to allocations that are optimal from a social point of view; and we consider an allocation optimal if it maximises the sum of utilities of the agents in the system, *i.e.* if it maximises (utilitarian) social welfare [1].

Unfortunately, the complexity associated with the task of identifying rational deals, *i.e.* deals that are beneficial for all participants, is typically very high and often hinders the successful transfer of theoretical results to practical applications. Common approaches to tame this type of complexity are to restrict the range of utility functions agents can use to model their preferences and to exploit the structure of utility functions during negotiation [4, 15, 17]. This is also the approach we are going to follow in this paper. Concretely, we are going to discuss so-called *tree-structured domains*, where the utility functions of agents are subject to certain structural restrictions. We use a compact representation of utility functions based on the notion of *k-additivity*, which restricts the synergies between different resources to bundles of at most k items each [11]. Tree-structured domains are sets of k-additive utility functions where the bundles for which there are synergies do not overlap.

We then propose two protocols that restrict negotiation to deals involving only the smallest bundles first, and then incrementally bigger bundles. However, these guided negotiation processes are not guaranteed to converge to an allocation with maximal social welfare, *i.e.* the protocols need to be enhanced further. The first protocol we propose is made *intrusive* in the sense that it modifies the structure of agents' utility functions. The second protocol has a centralised flavour and introduces a *system agent* that can *compensate* for a temporary loss in utility. This combination of *guidance* and *social compensation* allows us to restore the convergence properties of the negotiation framework.

Paper Overview

In the remainder of this introduction we briefly review the multilateral negotiation framework we take as the basis for our work [9, 18]. We then exemplify the high complexity of the framework, in Section 2, by analysing the number of deals that agents need to consider at each stage during a negotiation process. The class of tree-structured utilities is introduced in Section 3. Section 4 then presents a number of protocols exploiting these structural properties of util-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

ity functions, and shows how they reduce the complexity of the negotiation process. Our theoretical results are complemented by a brief experimental analysis of the protocols. Section 5 concludes with a discussion of the benefits and drawbacks of the approach advocated in this paper.

Multilateral Negotiation over Indivisible Resources

We briefly review the negotiation framework originally introduced by Sandholm [18], albeit in the context of task allocation problems (see [9] for a detailed introduction). Let $\mathcal{A}$ be a finite set of *agents* and let $\mathcal{R}$ be a finite set of indivisible *resources* that cannot be shared. An *allocation* A is a partitioning of the resources in $\mathcal{R}$ amongst the agents in $\mathcal{A}$. Each agent $i \in \mathcal{A}$ is equipped with a *utility function* $u_i : 2^{\mathcal{R}} \rightarrow \mathbb{R}$. We write $u_i(A) = u_i(A(i))$ for the utility assigned by agent i to the bundle it receives in allocation A. A *deal* may involve the reallocation of any number of resources between any number of agents. We therefore specify deals $\delta = (A, A')$ simply as pairs of allocations (with $A \neq A'$). A deal may be accompanied by a monetary *side payment*. We write p_i for the amount paid by agent i (negative in case i receives money). The sum of side payments associated with any given deal has to add up to 0. A deal $\delta = (A, A')$ with side payments p is called *individually rational* iff $u_i(A') - u_i(A) - p_i > 0$ for each agent i involved in the deal (and $p_i = 0$ otherwise). Agents are assumed to only negotiate such individually rational deals.

To assess the overall quality of an allocation, we use the concept of (utilitarian) *social welfare* [1]: $sw(A) = \sum_{i \in \mathcal{A}} u_i(A)$. It is well-known that a deal is individually rational iff it increases social welfare [9]. It is also known that any sequence of individually rational deals will eventually result in an allocation with maximal social welfare [18]. However, this *convergence result* only applies in case agents can negotiate deals involving any number of agents and resources. A recent result has identified the class of *k-deals* (involving at most k resources each) as sufficient to guarantee convergence to an optimal allocation in case all agents use *additively k-separable* utility functions to model their preferences, *i.e.* in case $\mathcal{R}$ can be divided into several preferentially independent bundles of at most k items each [4].

2. NUMBER OF BOUNDED DEALS

Results that identify conditions under which the convergence to a socially optimal allocation remains possible when agents only negotiate deals involving at most k items each significantly reduce the complexity of negotiation (at least in cases where k is small). In particular, the restriction to k-deals significantly reduces the number of deals to be considered at any given stage. In this sense, the quoted result on additively k-separable domains [4] is a first step towards a realistic multilateral negotiation system. However, even in such favourable cases it may be infeasible for agents to identify a k-deal that is acceptable to all parties involved. In this section, we are going to demonstrate this problem by analysing the number of bounded deals that could be implemented in a given allocation.

Recall that $|\mathcal{A}|$ is the number of agents and $|\mathcal{R}|$ is the number of resources in the system. This means that there are $|\mathcal{A}|^{|\mathcal{R}|}$ possible allocations overall (for each resource, we can choose which agent to give it to). Hence, in any given situation there are $|\mathcal{A}|^{|\mathcal{R}|} - 1$ potential deals to consider (we could move to any alternative allocation). This number can, of course, be somewhat reduced, for instance by discounting any previous allocations, but the remaining number of potential deals that need to be checked for their mutual acceptability would still be prohibitively high.

Bounded Number of Resources

The situation improves significantly if we restrict ourselves to bounded deals involving at most k distinct resources. There are $\binom{|\mathcal{R}|}{k}$ ways of selecting k items from the full set $\mathcal{R}$. Then each of the chosen k items can be moved to any of the $|\mathcal{A}|$ agents in the system. If the agent receiving a given item is the same agent where that item resided before the deal, then we take this as an encoding for the item not being moved at all, *i.e.* we are also capturing those deals that involve fewer than k resources. Only one of these cases needs to be excluded, namely the case where not a single of the k chosen resources gets moved (as this would not count as a valid deal). Hence, we end up with the following overall number of possible k-deals for any given allocation A:

$$\binom{|\mathcal{R}|}{k} \cdot (|\mathcal{A}|^k - 1) \tag{1}$$

This is polynomial in both the number of agents and the number of resources, while it has been exponential in the number of resources for deals without structural restrictions. In the case of $k = 1$, for instance, we only have to consider $|\mathcal{R}| \cdot (|\mathcal{A}| - 1)$ potential deals from any given allocation.

Bounded Number of Agents

In case we restrict the *number of agents* involved in a deal, the number of possible deals is not constant anymore, but varies depending on the current distribution of goods. To see this, we first compute the number of deals that are possible within a subgroup $S \subseteq \mathcal{A}$ of agents for a given allocation A. For any given agent $i \in S$, and any resource $r \in A(i)$ currently held by i, i can choose to which agent within the group S the item r should go (or whether it should remain with i). This makes $|S|^{|A(i)|}$ choices for agent i. The same is true for every agent in S (and in the end we still need to subtract 1 to account for the case were not a single resource is moving). We end up with the following formula:

$$|S|^{\sum_{i \in S} |A(i)|} - 1 \tag{2}$$

This number is constant as long as we do not change the overall number of resources held by the agents in S. For a group of two agents $S = \{i, j\}$, for instance, the above formula simplifies to $2^{|A(i)|+|A(j)|} - 1$. Hence, the number of *bilateral* deals (involving only two agents) that are implementable given allocation A can be computed as follows:

$$\sum_{i \neq j \in \mathcal{A}} (2^{|A(i)|+|A(j)|} - 1) \tag{3}$$

Now this number is not constant anymore; it depends on the distribution of resources in allocation A. Take a scenario with 3 agents and 9 resources. If each agent holds 3 resources, then the above formula shows that there are $3 \cdot (2^{3+3} - 1) = 189$ potential bilateral deals. If, on the other hand, we start with an allocation where one of the agents holds all 9 resources to begin with, then there are $2 \cdot (2^{9+0} - 1) + (2^{0+0} - 1) = 1022$ bilateral deals. Indeed, formula (3) shows that the number of possible deals increases as

resources become less evenly distributed amongst the agents in the system. It also shows that, even if we only allow for bilateral deals, the number of deals to be considered is exponential in $|\mathcal{R}|$, the overall number of resources, in case the distribution of resources is highly unbalanced. If the distribution of resources is reasonably well balanced, on the other hand, then the number of possible bilateral deals is "only" exponential in $\frac{|\mathcal{R}|}{|\mathcal{A}|}$, which may be manageable in cases where the number of resources is of a similar order of magnitude as the number of agents.

In conclusion, our discussion in this section has given some insight into the factors that determine the number of deals to be considered in each negotiation round and we have seen that the restriction to deals involving only a limited number of resources can result in a significant reduction in complexity. If we restrict deals to those between a bounded number of agents, the positive effect is considerably less significant. In either case, the number of deals to be considered typically remains very high and makes the design of general strategies for multilateral negotiation particularly challenging.

3. TREE-STRUCTURED DOMAINS

In this section, we are going to introduce a restriction on utility functions which we call *tree-structured* utilities. This is a natural restriction for many applications, which will allow us to drastically reduce the search space without losing the convergence properties.

The k-additive Form

An important parameter in the specification of any multiagent resource allocation problem is the way we represent the preferences of individual agents over alternative bundles of resources. A natural and compact language is the so-called *k-additive form* of representing utility functions. A utility function u_i is k-additive iff there exists a set of *coefficients* $\{\alpha_i^T \mid T \subseteq \mathcal{R}\}$ such that $\alpha_i^T = 0$ whenever $|T| > k$, and the following equality holds for all bundles $R \subseteq \mathcal{R}$:

$$u_i(R) = \sum_{T \subseteq R} \alpha_i^T \quad (4)$$

That is, agent i enjoys an increase in utility of α_i^T when it owns all the items in T together, *i.e.* α_i^T represents the synergetic value of this bundle. If a utility function is defined in terms of such coefficients, we say that it is given in k-additive form. When describing examples, we are going to use a simplified notation. For instance, $u_i = 5.r_1 + 3.r_2.r_3$ represents a 2-additive utility function with two non-zero coefficients: $\alpha_i^{\{r_1\}} = 5$ and $\alpha_i^{\{r_2,r_3\}} = 3$.

The k-additive form goes back to the work of Grabisch [11] in fuzzy measure theory and has recently been advocated as a useful preference representation language in the context of both combinatorial auctions [5] and distributed negotiation schemes [3]. Similar preference languages have been used in Artificial Intelligence for some time [2]; what is special about the k-additive form is that it parametrises the degree of synergies in a natural manner. If k can be chosen freely, then *any* utility function can be written in k-additive form, and the k-additive form is typically significantly more *concise* than simply enumerating the utilities for all bundles [3]. For ease of presentation, we are going to assume that the coefficient for the empty bundle $\alpha_i^{\{\}} = 0$ for all agents $i \in \mathcal{A}$ (but the applicability of our results is not affected by this assumption).

Terms and Trees

To define tree-structured domains, we require utilities to be represented in k-additive form. The set of *terms* of a utility function u_i is the set of bundles T with a non-zero coefficient α_i^T in the k-additive form. For instance, the utility function $u_i = 16.r_2 + 5.r_1.r_2.r_4 - 2.r_1.r_3$ uses the terms r_2, $r_1.r_2.r_4$, and $r_1.r_3$. Let $\{u_1, \ldots, u_n\}$ be the set of utility functions of the agents in the system. $\mathcal{T}$ will denote the set of all terms appearing in the representation of any of the utility functions $u_1, \ldots, u_n$, and α_i^T will denote the coefficient of term T in u_i. Finally, $\mathcal{T}^l$ denotes the set of terms in $\mathcal{T}$ consisting of exactly l resources, while $\mathcal{T}^{\leq l}$ denotes the set of terms in $\mathcal{T}$ with at most l resources each.

Intuitively, tree-structured utilities are k-additive functions in which there are no "overlapping" terms.

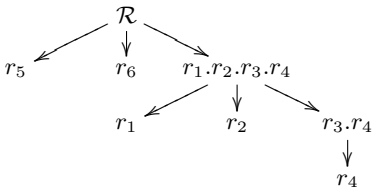
DEFINITION 1. *A set of utility functions $\{u_1, \ldots, u_n\}$ is called tree-structured iff, when represented in k-additive form, it is the case that for all terms $T_1, T_2 \in \mathcal{T}$ we have either $T_1 \subseteq T_2$, or $T_1 \supseteq T_2$, or $T_1 \cap T_2 = \{\}$.*

It is also helpful to note that a set of utility functions is tree-structured iff the terms of $\mathcal{T}$ can be represented by a tree, in which $\mathcal{R}$ is the root, and each term is a node. Branches of the tree represent the $\subset$ relation. The following example illustrates this representation.

EXAMPLE 1. *Consider the set $\{u_1, u_2, u_3\}$ of utility functions defined as follows:*

$$\begin{aligned} u_1 &= r_2 + 3.r_5 \\ u_2 &= 3.r_1 + 10.r_1.r_2.r_3.r_4 + 8.r_5 + 4.r_6 \\ u_3 &= r_6 - r_4 + 8.r_3.r_4 \end{aligned}$$

These functions are 4-additive as well as tree-structured. The set of terms $\mathcal{T} = \{r_1, r_2, r_4, r_5, r_6, r_3.r_4, r_1.r_2.r_3.r_4\}$ can also be represented by the following tree:



The idea of tree-structured domains has also been used by Rothkopf *et al.* [17] in the context of combinatorial auctions. In their work, bids are restricted to bundles that belong to a given tree structure. They show that this restriction makes the winner determination problem tractable. We are going to exploit this reduced level of complexity as well, although by designing a negotiation protocol rather than an optimisation algorithm.

We note here that any set of k-additive tree-structured utilities is also additively k-separable. Hence, the convergence towards an allocation with maximal social welfare in case agents only negotiate individually rational k-deals is guaranteed by the aforementioned result [4].

$\mathcal{T}$-deals

Still, as argued in Section 2, even finding a rational k-deal can often be computationally infeasible. Our aim is therefore to exploit this new restriction on utility functions to

come up with a new type of deal that is less complex, while still ensuring the same convergence properties.

DEFINITION 2. *Let $\mathcal{T}$ be a set of terms. A $\mathcal{T}$-deal is a deal involving the reallocation of the resources of a single term in $\mathcal{T}$ from any number of senders to a single receiver.*

For our example above, for instance, moving all of r_1, r_2, r_3 and r_4 from whichever agent currently holds them to, say, agent 1 would be a valid $\mathcal{T}$-deal. Only moving r_3, on the other hand, would not constitute a $\mathcal{T}$-deal (because r_3 does not occur as a term in the representation of the utility function any of our three agents).

How many $\mathcal{T}$-deals are possible from a given allocation? The exact number will depend on the particular allocation in question, but $|\mathcal{T}| \cdot |\mathcal{A}|$ will certainly be an upper bound: for any term in $\mathcal{T}$ we can choose which of the agents in $\mathcal{A}$ should receive that term in its entirety. How many terms can we have in $\mathcal{T}$ in the worst case? Firstly, the number of non-overlapping terms of size k is bound by $\lceil \frac{|\mathcal{R}|}{k} \rceil$. Secondly, given a set of k resources, we can use these to construct a subtree with at most $2k-1$ nodes (to see this, consider that splitting each term/node into two balanced subnodes will give a tree exhibiting the maximal number of nodes). Hence, we end up with the following upper bound for the number of terms in $\mathcal{T}$ for any set of k-additive tree-structured utility functions:

$$|\mathcal{T}| \quad \leq \quad (2k-1) \cdot \left\lceil \frac{|\mathcal{R}|}{k} \right\rceil \tag{5}$$

Hence, even if we do not put any restrictions on k (*i.e.* in case $k = |\mathcal{R}|$), we end up with an upper bound for the number of possible $\mathcal{T}$-deals of $(2 \cdot |\mathcal{R}| - 1) \cdot |\mathcal{A}|$. This number is of the same order of magnitude as the number of 1-deals available from any given allocation, and significantly lower than the number of possible k-deals for any $k > 1$. In other words, the complexity of finding rational $\mathcal{T}$-deals is also very low compared to that of finding rational k-deals.

However, as illustrated by our next example, simply restricting negotiation to individually rational $\mathcal{T}$-deals will not be sufficient to guarantee that an allocation with maximal social welfare will be reached eventually in all cases.

EXAMPLE 2. *Consider the following utility functions:*

$$\begin{aligned} u_1 &= 10.r_1 \\ u_2 &= 10.r_2 \\ u_3 &= 11.r_1.r_2 \end{aligned}$$

Furthermore, let the initial allocation be the allocation assigning all items to agent 3. Then the (only) optimal allocation gives r_1 to agent 1 and r_2 to agent 2. Clearly, there are no rational $\mathcal{T}$-deals allowing the system to reach this optimal allocation. For any of these deals, the utility of agent 3 would drop from 11 to 0, while the receiving agent (either 1 or 2) would only gain 10 utility points. Hence, it would not be possible to arrange for suitable side payments making such a deal beneficial for everyone involved.

In order to restore the desired convergence properties of the framework, we will need to investigate more complex protocols restricting the negotiation process and exploiting, through $\mathcal{T}$-deals, the properties of tree-structured utilities.

4. TREE-CLIMBING PROTOCOLS

The basic idea of *tree-climbing protocols* is to allow $\mathcal{T}$-deals involving only the smallest bundles first, then to incrementally allow bigger bundles, until all possible $\mathcal{T}$-deals have been tried. The *Naive Tree-Climbing Protocol* is a simple implementation of this principle.

Algorithm 1 Naive Tree-Climbing Protocol

Require: n agents with tree-structured utilities $u_1, \ldots, u_n$ (with $\mathcal{T}$ the set of all terms)
1: $l \leftarrow 1$.
2: **repeat**
3: Restrict allowed deals to $\mathcal{T}^l$-deals.
4: Let agents negotiate individually rational $\mathcal{T}^l$-deals.
5: **if** no more deal can be conducted **then**
6: $l \leftarrow l + 1$
7: **end if**
8: **until** $l > |\mathcal{R}|$

Clearly, this protocol is more restrictive than the liberal "protocol" consisting of allowing any $\mathcal{T}$-deals, *i.e.* it cannot guarantee an optimal outcome either (see Example 2). By inspecting more carefully the scenario described in Example 2, however, we find clues indicating possible improvements of this protocol: During the first step of the protocol (where $l = 1$, and only $\mathcal{T}^1$-deals are allowed), the term $r_1.r_2$ in u_3 prevents the resources from moving away from agent 3. If, during this first step, terms containing more than one resource were to be *removed* from the utility functions, then the resources would be able to move towards agents 1 and 2.

4.1 An Intrusive Tree-Climbing Protocol

Based on this idea, we design a new protocol which, during round l, removes from all utility functions any terms involving more than l resources. This protocol is called the *Intrusive Tree-Climbing Protocol* (ITCP). It is intrusive in the sense that it modifies the utility functions of each agent during each round of the protocol. In the specification of the ITCP, $u_i^{\leq l}$ denotes the utility function u_i with all terms of size greater than l having been removed. For instance, if $u_i = 12.r_1 + 3.r_1.r_2 + r_1.r_3 - 2.r_1.r_2.r_4$, then $u_i^{\leq 1} = 12.r_1$, $u_i^{\leq 2} = 12.r_1 + 3.r_1.r_2 + r_1.r_3$, and $u_i^{\leq 3} = u_i$.

Algorithm 2 Intrusive Tree-Climbing Protocol (ITCP)

Require: n agents with tree-structured utilities $u_1, \ldots, u_n$ (with $\mathcal{T}$ the set of all terms)
1: $l \leftarrow 1$.
2: **repeat**
3: Restrict allowed deals to $\mathcal{T}^l$-deals,
4: Enforce all agents i to change their utilities to $u_i^{\leq l}$.
5: Let agents negotiate individually rational $\mathcal{T}^l$-deals.
6: **if** no more deal can be conducted **then**
7: $l \leftarrow l + 1$
8: **end if**
9: **until** $l > |\mathcal{R}|$

Convergence

As we shall see, the ITCP guarantees convergence to an allocation with maximal social welfare, provided the utility functions of all agents are tree-structured with *positive* coefficients. We are first going to show that each round of the

protocol will lead to an allocation deemed optimal wrt. the truncated utility functions of agents populating the system. The social welfare of an allocation A wrt. truncated utility functions in round l is denoted by $sw^{\leq l}(A) = \sum_{i \in \mathcal{A}} u_i^{\leq l}(A)$.

LEMMA 1. *For any set of tree-structured utilities with positive coefficients, if at the end of round l of the ITCP an optimal allocation wrt. $sw^{\leq l}$ has been reached, then at the end of round $l+1$ an optimal allocation wrt. $sw^{\leq l+1}$ will be reached.*

PROOF. Assume after round l the allocation A^l has been reached and that allocation is optimal wrt. $sw^{\leq l}$. Let A^{l+1} be (any of) the allocation(s) that are optimal wrt. $sw^{\leq l+1}$. Because of the tree structure, we can consider the optimal allocation of resources belonging to distinct terms in $\mathcal{T}^{l+1}$ separately. Let $T \in \mathcal{T}^{l+1}$. We can distinguish three cases: (1) If T is owned by a single agent in A^{l+1}, then we are done, because this can be achieved using a $\mathcal{T}^{l+1}$-deal. (2) If not, and if T is also not owned by a single agent in A^l, then we are done, because the optimal allocation will be the same as in A^l. (3) If it were the case that T is owned by a single agent in A^l but not in A^{l+1}, then that would pose a problem, because the necessary reallocation would not be a $\mathcal{T}^{l+1}$-deal. However, this situation is impossible due to our restriction to positive coefficients. □

We are now in a position to prove that the ITCP does indeed guarantee the intended convergence property.

THEOREM 1. *For any set of tree-structured utilities with positive coefficients, following the ITCP will result in an allocation of resources with maximal social welfare.*

PROOF. The proof is by induction over the number of rounds. In the first round, all agents have additive utility functions, and it has been shown elsewhere that negotiation with individually rational deals involving only a single resource each will converge to an optimal allocation in additive domains [9]. Lemma 1 shows that the property of having maximal social welfare wrt. truncated utility functions persists from one round to the next. Hence, after the final round, the allocation reached will have maximal social welfare wrt. the full utility functions. □

Negative Coefficients

Observe that our proof of Lemma 1 shows why we need to restrict ourselves to k-additive functions with *positive* coefficients. If there are negative coefficients, then also deals that "destroy" an existing term can increase utility. This is also demonstrated by the following example.

EXAMPLE 3. *Consider the following utilities:*

$$u_1 = 2.r_1.r_2 + 10.r_3 - 100.r_1.r_2.r_3$$
$$u_2 = 5.r_3$$

After round 2 of the ITCP, agent 1 would own all three resources, and the social welfare relative to terms of length ≤ 2 would be 12. In round 3, the only individually rational deal involving a term with exactly 3 resources would be to move $r_1.r_2.r_3$ from agent 1 to agent 2 (increasing social welfare from -88 to 5). But then the deal of moving r_3 back to agent 1 would increase social welfare even further, namely from 5 to 10 (which is the maximal social welfare attainable for this resource allocation problem). However, this deal would not be admissible in the ITCP during round 3.

A potential way around this problem would be to widen the class of $\mathcal{T}$-deals, and to also allow for deals that involve a term from $\mathcal{T}$ being removed from a *single sender* (while our actual definition of $\mathcal{T}$-deal foresees a single receiver). With that extension, the ITCP could be proved to converge to an allocation with maximal social welfare also for tree-structured utilities with arbitrary coefficients, although it would also lose much of its appeal by increasing the range of available deals at each stage significantly.

Note that in the work of Rothkopf *et al.* [17] this problem does not occur, because the standard bidding language for combinatorial auctions (the so-called "OR language") can only express superadditive preferences in the first place [14].

Communication Complexity

The *communication complexity* of a negotiation scheme assesses the length of negotiation processes and the amount of information that agents exchange. The "first level" of communication complexity is concerned with the *number of deals* required to reach an optimal allocation [8]. Here we can distinguish two types of upper bounds, namely the length of the longest and that of the shortest path to the optimum. For the ITCP we obtain the following bounds:

- The ITCP requires agents to implement a maximum of $|\mathcal{T}| \cdot (|\mathcal{A}| - 1)$ deals (*longest path*), after which the optimal allocation will be reached.
- In the ITCP, agents can always reach an optimal allocation by negotiating at most $|\mathcal{T}|$ deals (*shortest path*).

The shortest path applies if agents always implement the "ideal" deal for every term in $\mathcal{T}$. The longest path applies if utility functions are such that we can move each term to every agent before it finally reaches its eventual owner.

Discussion

A possible interpretation of the ITCP would be that it works with a sequence of progressively more fine-grained *approximations* to the real utility functions in such a way that structurally simple deals become individually rational (wrt. these approximations). Of course, these are approximations wrt. the *representation* of utility functions (in k-additive form), but not necessarily wrt. the values that utility functions assign to bundles of resources.

These two dimensions may, however, coincide in many practical cases. If we assume that the synergetic effect of a bundle of resources reduces as the cardinality of that bundle increases, then the sequence of auxiliary utility functions used in the ITCP will indeed be progressively more accurate in terms of the values they assign to bundles as negotiation develops. In other words, if the coefficients in the k-additive form are high for single resources, and if they get smaller as the size of the respective term increases, then the utility functions used in the ITCP become more accurate in each round. This is a very reasonable assumption to make: the larger a bundle of resources is, the more difficult would it be for an agent to estimate the additional benefit incurred by owning all the resources in that bundle *together* (beyond the benefit incurred by the relevant subsets). Indeed, the very same (cognitive) argument is one of the main reasons why k-additive utilities with low values of k are not only computationally attractive but also highly relevant in practice.

Nevertheless, the ITCP *does* violate the postulate of full individual rationality for all stages of a negotiation process.

Also, because of its "intrusive" aspect, this protocol is not completely satisfactory: it cannot be applied when agents are not under the control of the designer or when they are not willing to temporarily manipulate their own preferences in order to implement the protocol correctly. Clearly, such a protocol would be of little use when agents are humans, or simply when privacy issues prevent the designer from directly accessing the agents' preferences.

4.2 A Non-Intrusive Protocol

To overcome the drawbacks of our intrusive protocol, we design a third protocol in which utilities are not modified, but where a special payment function influences the rationality considerations of each agent. In addition, the system itself also participates in the side payments, by giving money to or taking money from the negotiating agents. For this purpose, we introduce a so-called *system agent*. The trick here is that these side payments have exactly the same effect as truncating the utility functions, but without being intrusive (as shown in Theorem 2). Thus, this protocol could be used in real applications, even with human agents as long as utilities have been elicited.

This new protocol is called the *Omniscient ϵ-Altruistic Tree-Climbing Protocol* (OATCP). In this protocol the system actively mediates the negotiation process. The protocol is called *omniscient*, because the system needs to know the utility functions of all agents. It is called *ϵ-altruistic*, because the system must be prepared to share its money with the agents in order to reach an allocation with maximal social welfare, and because the amount of money it shares depends on a parameter ϵ. For the specification of the OATCP, we define $\Delta^{\leq l}_{(A,A')} = sw^{\leq l}(A') - sw^{\leq l}(A)$. This is the increase in social welfare for the deal $\delta = (A, A')$ wrt. truncated utilities considering only terms of size $\leq l$.

Algorithm 3 Omniscient-Altruistic Tree-Cl. Pr. (OATCP)

Require: n agents with tree-structured utilities $u_1, \ldots, u_n$ and a parameter $\epsilon \in (0, 1]$ (with $\mathcal{T}$ the set of all terms)

1: All agents transmit their utility functions to the *system agent*.
2: $l \leftarrow 1$.
3: **repeat**
4: Restrict allowed deals to $\mathcal{T}^l$-deals.
The side payment for agent i for a $\mathcal{T}^l$-deal $\delta = (A, A')$ involving n' agents is computed as follows:
$p_i = u_i(A') - u_i(A) - \frac{\epsilon}{n'} \cdot \Delta^{\leq l}_{(A',A)}$.
For the system agent, $p_{sys} = -\sum p_i$.
5: Let agents negotiate individually rational deals.
6: **if** no more deal can be conducted **then**
7: $l \leftarrow l + 1$
8: **end if**
9: **until** $l >\mid \mathcal{R} \mid$

Role of the System Agent

Note that in the OATCP the *system agent* could be viewed as a kind of bank giving or receiving money from agents. The amount of money shared with other agents depends on the difference of satisfaction enjoyed by the whole society before and after the deal and on the value of the parameter ϵ.

- When ϵ is close to 1, the system agent in not guaranteed to earn any money in the long run. In that case, the system agent is said to be *altruistic*, because the other agents will benefit from this by earning a lot.
- When ϵ is close to 0, then during each deal, the system agent takes from the other agents as much money as possible. Thus, the negotiating agents will earn the minimum amount of money necessary to reach the optimal allocation, and the system agent will earn a lot more (not altruistic at all in that case).

The protocol could be extended in such a way that the money earned by the system agent would later be redistributed. One option would be to simply divide the money equally amongst all agents, but how to best set up this redistribution in view of both fairness considerations and strategic issues is a question that warrants further investigation.

Convergence

The following results show that this protocol also converges to an allocation with maximal social welfare, just as the ITCP does. We first establish that a deal is "rational" with the special payment function (*i.e.* utility gains outweigh payments for each of the agents except the system agent) iff it increases social welfare wrt. truncated utilities.

LEMMA 2. *During round l of the OATCP, a given deal $\delta = (A, A')$ is rational iff $sw^{\leq l}(A) < sw^{\leq l}(A')$.*

PROOF. '$\Rightarrow$': Suppose the deal is rational. That is, for each agent i involved, $u_i(A') - u_i(A) - p_i > 0$. Thus,

$$sw^{\leq l}(A') - sw^{\leq l}(A) \;=\; \frac{1}{\epsilon} \cdot \sum (u_i(A') - u_i(A) - p_i) \;>\; 0$$

'$\Leftarrow$': Now suppose $sw^{\leq l}(A') > sw^{\leq l}(A)$. Then for each agent i involved in the deal, the rationality criterion is met:

$$u_i(A') - u_i(A) - p_i \;=\; \frac{\epsilon}{n'} \cdot \Delta^{\leq l}_{(A',A)} \;>\; 0$$

This completes our proof of Lemma 2. □

This directly leads to the convergence result.

THEOREM 2. *For any set of tree-structured utilities with positive coefficients, following the OATCP will result in an allocation of resources with maximal social welfare.*

PROOF. We show that the deals accepted in this protocol are the same as those accepted in the ITCP protocol. Clearly, during round l, a deal $\delta = (A, A')$ is accepted in the ITCP iff $sw^{\leq l}(A) < sw^{\leq l}(A')$. Lemma 2 shows that the OATCP behaves in exactly the same way. The claim then follows from Theorem 1. □

Given this operational equivalence between the two protocols, the communication complexity of the OATCP (*i.e.* the number of deals required to reach an optimal allocation following the protocol) is the same as for the ITCP.

Money Earned by the System Agent

A critical point of the OATCP is that it requires an external provider of money. However, the following result shows that the system agent will actually *earn* money over the entire duration of the protocol, as long as we choose a sufficiently small value for ϵ.

THEOREM 3. *If ϵ is sufficiently small, then the system agent will have earned a positive amount of money by the end of any OATCP negotiation.*

PROOF. Let us first compute how much is earned by the system agent each time a deal is conducted. Suppose at step l, a deal $\delta = (A, A')$ is conducted by n' agents. The system agent makes the following payment:

$$\begin{aligned} p_{sys} &= -\sum \left(u_i(A') - u_i(A) - \frac{\epsilon}{n'} \cdot \Delta^{\leq l}_{(A',A)} \right) \\ &= -(sw(A') - sw(A)) + \epsilon \cdot \Delta^{\leq l}_{(A',A)} \end{aligned}$$

Now let d be the number of deals conducted during the whole of the negotiation process. Furthermore, let A_{init} be the initial allocation and let A_{fin} be the final allocation If p^{total}_{sys} denotes the total amount of money the system agent pays out during the whole negotiation process, then $-p^{total}_{sys}$ is the money earned by the system. We get:

$$\begin{aligned} -p^{total}_{sys} &= \sum_{\delta=(A,A')} \left(sw(A') - sw(A) - \epsilon \cdot \Delta^{\leq l}_{(A',A)} \right) \\ &\geq sw(A_{fin}) - sw(A_{init}) - \epsilon \cdot d \cdot \max_{A,A',l} \Delta^{\leq l}_{(A',A)} \end{aligned}$$

Clearly, we can see that $-p^{total}_{sys}$ converges to the overall social surplus as ϵ decreases:

$$\lim_{\epsilon \to 0}(-p^{total}_{sys}) = sw(A_{fin}) - sw(A_{init})$$

Hence, as long ϵ is chosen small enough, the system agent can always earn during negotiation. □

The central idea in the OATCP is that the system agent (*i.e.* the mechanism) can give a "loan" to agents to allow them to accept otherwise disadvantageous deals, but that these agents repay this loan later on when they arrange deals that are overly beneficial for themselves. This general idea would also be applicable in other scenarios than we have considered here, where agents do not necessarily have tree-structured utility functions. As long a the final allocation has higher utilitarian social welfare than the initial allocation and as long as the system agent has sufficient funds to temporarily sustain allocations with very low social welfare, *any* sequence of deals can be made individually rational from the viewpoint of the negotiating agents and would still result in an eventual surplus for the system agent.

4.3 Experimental Analysis

We have shown that both the ITCP and the OATCP perform optimally in case all agents model their preferences using tree-structured utilities with positive coefficients. While this is a reasonably wide class of utility functions that are relevant to real application domains, it is also clear that it will not always be possible to meet these constraints. Still, the ideas implemented in our protocols seem to have a general appeal and may also be useful (albeit not to the degree of guaranteeing optimal outcomes in all cases) in domains that are not tree-structured.

To test this intuition, we have carried out a number of experiments with agents that do not have tree-structured utilities. Figure 1 shows an example. The graph relates the performance of the ITCP/OATCP and of a protocol only implementing 1-deals to the maximal social welfare attainable. In each run, we have generated the utility functions of 4 agents over 10 resources in k-additive form by randomly choosing 8 terms and coefficients each. The size of terms is drawn from $\{1..k_{max}\}$ and the size of coefficients is drawn from {1..100} (uniform distribution). The horizontal axis shows the value of k_{max} and the vertical shows the social welfare of the final allocation (averaged over 20 runs). The experiment shows that the ITCP/OATCP clearly outperforms the 1-deal protocol for larger values of k_{max}, and that it typically manages to reach an allocation with over 95% of the maximal social welfare. We have run similar experiments with other distributions (not reported here for lack of space). The ITCP/OATCP generally performs well and is preferable to a 1-deal protocol whenever the agents' utilities include a significant number of terms of size greater than 1.

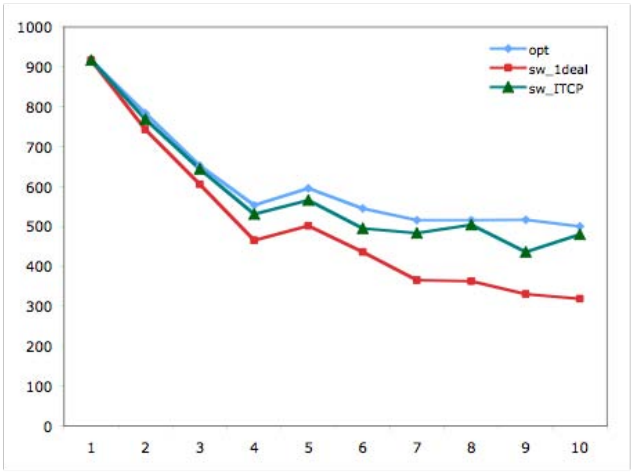


Figure 1: Performance of different protocols

5. CONCLUSION

We have further analysed a negotiation framework previously studied by several authors [7, 8, 9, 18]. While most work on negotiation in multiagent systems has addressed either bilateral negotiation [16] or auctions [6], this framework is multilateral, *i.e.* deals may involve any number of agents and any number of resources. The requirement for full multilateral negotiation stems from the fact that agents are assumed to be both rational and myopic. Recent results have identified situations in which deals involving only a limited number of resources each are required to reach allocations that are optimal from a social point of view [4].

However, the task of identifying rational deals, *i.e.* deals that are beneficial for all participants remains highly complex. In this paper, we have shown that if all agents model their preferences by means of *tree-structured* utility functions, then finding an allocation with maximal social welfare does become feasible by implementing adequate protocols. Two protocols, clearly designed to exploit the specific structure of these functions, have been proposed in this paper. The crucial contribution in our proposal, we believe, is the *combination of two ideas:*

- The central idea in the ITCP has been to provide a protocol that *guides* negotiation. In each round, there are only a relatively small number of options as to which deal to choose (and the complexity of these deals increases stepwise, from round to round). The particular choices made in the definition of the ITCP mean that deals are not too far from truly rational deals (certainly if we make some additional assumptions on the nature of utility functions, as discussed earlier).

- The additional idea in the OATCP has been to introduce a system agent that can *compensate* for a temporary loss in social welfare (provided that such a loss is due to a deal sanctioned by the guidelines implemented in the ITCP). It thereby allows for a wider range of deal sequences leading to the optimal allocation, including sequences of structurally simple deals.

This combination of *guidance* and *social compensation* makes our approach work. While our formal results are restricted to tree-structured domains, we believe that the central ideas are general enough to be applied to other scenarios as well. This view is supported by our experimental results with utility functions that are not tree-structured.

Our proposal may be considered a hybrid between distributed and centralised approaches to resource allocation. On the one hand, allocations emerge in a distributed manner as a result of a sequence of local negotiation steps. On the other hand, the concept of the system agent introduces a centralised element. One of the main arguments usually given in favour of distributed approaches over centralised ones (such as combinatorial auctions), is that it may not always be possible to find an agent that could act as the central authority (the auctioneer) executing the allocation procedure.[1] To some degree, the same reservations may be made against our proposal. However, we believe that there is at least one important difference. In a combinatorial auction, the auctioneer is endowed with the (significant) computational burden of deciding the final allocation (the so-called winner determination problem [6]). In our approach, on the other hand, there are no computationally hard tasks associated with the role of the system agent. Of course, in the context of the tree-structured domains studied here, the centralised task is not prohibitively hard in itself. Still, our work shows that it can also be distributed safely amongst agents by employing our protocols. Importantly, our experiments suggest that the ideas underlying the design of these protocols also have a more general appeal.

A potential problem with our approach is that agents could stop participating in negotiation before the optimal allocation has been reached. Indeed, it may be rational for them to do so, because our protocols allow certain agents to temporarily enjoy higher utility than they would enjoy in the socially optimal allocation. In this sense, the framework loses the "anytime character" of the original system [18], where every new allocation would be guaranteed to have higher social welfare than its predecessor. This anytime character is particularly important because of the prohibitively high complexity of the general framework [3, 7, 8]. If optimal allocations cannot be guaranteed to be found within a reasonable amount of time, then the original framework does at least guarantee some improvement over the initial allocation when time runs out. In our proposal, the anytime character of the framework is far less important, because it dramatically reduces the complexity of negotiation and thereby makes convergence to an optimal allocation within a reasonable amount of time a realistic option. Nevertheless, the issue of incentive compatibility in the OATCP protocol remains an interesting topic for future work.

[1]Such arguments for distributed negotiation schemes are similar to those often given to motivate the interest in *distributed constraint optimisation problems* [10]. It would be interesting to explore whether algorithms developed in this area could be applied to our negotiation setting as well.

6. REFERENCES

[1] K. J. Arrow, A. K. Sen, and K. Suzumura, editors. *Handbook of Social Choice and Welfare*. North-Holland, 2002.

[2] U. Chajewska and D. Koller. Utilities as random variables: Density estimation and structure discovery. In *Proc. UAI-2000*. Morgan Kaufmann, 2000.

[3] Y. Chevaleyre, U. Endriss, S. Estivie, and N. Maudet. Multiagent resource allocation with k-additive utility functions. In *Proc. DIMACS-LAMSADE Workshop on Computer Science and Decision Theory*, 2004.

[4] Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. Negotiating over small bundles of resources. In *Proc. AAMAS-2005*. ACM Press, 2005.

[5] V. Conitzer, T. W. Sandholm, and P. Santi. Combinatorial auctions with k-wise dependent valuations. In *Proc. AAAI-2005*. AAAI Press, 2005.

[6] P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006.

[7] P. E. Dunne, M. Wooldridge, and M. Laurence. The complexity of contract negotiation. *Artificial Intelligence*, 164(1–2):23–46, 2005.

[8] U. Endriss and N. Maudet. On the communication complexity of multilateral trading: Extended report. *Journal of Autonomous Agents and Multiagent Systems*, 11(1):91–107, 2005.

[9] U. Endriss, N. Maudet, F. Sadri, and F. Toni. Negotiating socially optimal allocations of resources. *Journal of Artificial Intelligence Research*, 2006. To appear.

[10] B. Faltings and M. Yokoo. Introduction: Special issue on distributed constraint satisfaction. *Artificial Intelligence*, 161(1–2):1–5, 2005.

[11] M. Grabisch. k-order additive discrete fuzzy measures and their representation. *Fuzzy Sets and Systems*, 92:167–189, 1997.

[12] P. J. 't Hoen and J. A. La Poutré. A decommitment strategy in a competitive multi-agent transportation setting. In *AMEC V*. Springer-Verlag, 2004.

[13] M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille. Selecting and scheduling observations of agile satellites. *Aerospace Sciences and Technology*, 6:367–381, 2002.

[14] N. Nisan. Bidding languages for combinatorial auctions. In P. Cramton et al., editors, *Combinatorial Auctions*. MIT Press, 2006.

[15] V. Robu, D. Somefun, and J. A. La Poutré. Modelling complex multi-issue negotiations using utility graphs. In *Proc. AAMAS-2005*. ACM Press, 2005.

[16] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter*. MIT Press, 1994.

[17] M. H. Rothkopf, A. Pekeč, and R. M. Harstad. Computationally manageable combinational auctions. *Management Science*, 44(8):1131–1147, 1998.

[18] T. W. Sandholm. Contract types for satisficing task allocation: I Theoretical results. In *Proc. AAAI Spring Symposium: Satisficing Models*, 1998.

[19] P. R. Wurman, W. E. Walsh, and M. P. Wellman. Flexible double auctions for electronic commerce: Theory and implementation. *Decision Support Systems*, 24(1):17–27, 1998.

Predicting People's Bidding Behavior in Negotiation

Ya'akov Gal and Avi Pfeffer
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
{gal,avi}@eecs.harvard.edu

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]:

General Terms

Design, Experimentation

Keywords

Negotiation, decision-making, Opponent modeling

ABSTRACT

This paper presents a statistical learning approach to predicting people's bidding behavior in negotiation. Our study consists of multiple 2-player negotiation scenarios where bids of multi-valued goods can be accepted or rejected. The bidding task is formalized as a selection process in which a proposer player chooses a single bid to offer to a responder player from a set of candidate proposals. Each candidate is associated with features that affect whether or not it is the chosen bid. These features represent social factors that affect people's play. We present and compare several algorithms for predicting the chosen bid and for learning a model from data. Data collection and evaluation of these algorithms is performed on both human and synthetic data sets. Results on both data sets show that an algorithm that reasons about dependencies between the features of candidate proposals is significantly more successful than an algorithm which assumes that candidates are independent. In the synthetic data set, this algorithm achieved near optimal performance. We also study the problem of inferring the features of a proposal given the fact that it was the chosen bid. A baseline importance sampling algorithm is first presented, and then compared with several approximations that attain much better performance.

1. INTRODUCTION

Recent developments in automated decision-making are making it possible for agents to negotiate with each other in environments of increasing complexity, such as supply chains and market-based scheduling [16, 1]. At the same time, computer systems are becoming commonplace in our every day lives, and settings in which computers and people make decisions together are increasingly prevalent [4]. One area in which both of these trends interleave is electronic commerce, where computers acting as autonomous agents, or as proxies for individual people, are changing the way goods and services are traded on the Internet [21].

One of the main challenges for agents that negotiate in strategic environments is reasoning about the bidding strategies of others. For example, in the Supply Chain domain of the Trading Agent Competition [1] (TAC SCM), agents bid competing offers for supplying a series of multi-valued contracts, each satisfying a minimal configuration of computer hardware resources. The agent with the lowest offer is selected to supply each contract after procuring the necessary resources. Agents cannot observe each others' bids, resources or the winning bid configuration. For agents to out-bid their opponents without sacrificing future revenue, they must be able to predict their opponents' bids for current and future contracts.

Although much work has been done on predicting the bidding and pricing behavior of automatic agents in negotiation [16, 20, 19], modeling people's bidding behavior presents several new challenges. First, social factors such as altruism, self-interest and fairness affect people's negotiation strategies [3]. Second, people's behavior is diverse – the extent to which social factors affect behavior varies greatly among people [12]. Third, people's play does not generally adhere to traditional game-theoretic equilibria [13, 3]. Fourth, humans often make mistakes with regard to their reported utility function[2].

To explain these findings, researchers in behavioral economics have suggested that people's preferences in settings that include other decision-makers are affected by others' outcome as well as their own [15]. For example, in particular contexts, some buyers reject proposals that are highly beneficial to the sellers, even though they are beneficial to the buyers, because they see them as unfair. To be able to interact successfully with individuals, artificial agents must be able to learn how these social preferences affect the bidding behavior of different people.

This work presents a machine learning approach for meeting these challenges. Our study uses a series of take-it-or-leave-it negotiation scenarios in which one agent proposes a trade consisting of multiple goods to a responder agent who must accept or reject the trade. In previous work, we showed that a computer player that used a decision theoretic model to reason about the responder's social preferences outperformed other computer players that used standard game theoretic equilibria [10]. This work formalizes the task of bidding from the proposer's perspective without using a decision-theoretic model. It defines a *selection process* where a number of candidate proposals are available to the proposer and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

one of them is selected as the chosen bid. Previously we were only concerned with predicting whether or not the proposer's offer was accepted by the responder. In this work, we need to predict the outcome of a multi-way decision process where there are many possible offers for the proposer to choose from.

Each candidate is associated with features, that affect whether or not the candidate is chosen. In general, these features can represent anything about the decision-making that is salient to agents' models. In this work, these features are players' social preferences, defined in terms of functions over their payoffs.

In this work we address the following three tasks. First, we show how to predict the chosen bid given the features of all the candidate proposals. Second, we present and evaluate two algorithms for learning a selection process from data. One of the algorithms uses supervised learning, where an instance consists of the attributes of each proposal and the classification is whether or not the proposal was the chosen bid. In this approach, the probability of a particular candidate offer being selected does not depend on the quality of the attributes of other candidates. The other algorithm learns from each *pair* of candidate proposals, where one was the chosen bid and one was not. In this way, it learns from individual candidates, but also takes into account the quality of other candidates.

To capture the behavior of different types of people, both algorithms use a mixture model of belief networks, each associated with a separate utility function that is a weighted summation of its social preferences. In this way, each network predicts how likely a particular type of person is to bid any given proposal. The feature weights for each type are updated by using standard gradient descent. Because the type of a person is not observed, we estimate a probability distribution over the type space using a technique that is inspired by the Expectation Maximization algorithm [8].

These algorithms are evaluated on real data that was obtained by observing people negotiate with each other in laboratory settings as well as synthetic data that was designed by the researchers by sampling from pre-generated selection models. We expected the second algorithm to be better suited for learning selection processes because it learns from candidate proposals that are *not* chosen as well the chosen bid. This gives us vital information to use in the learning process. For example, if there are many candidate proposals with feature values that are similar to those of the chosen bid, their likelihood should be higher than that of a proposal whose feature values are *less* similar to the chosen bid. The results confirm our hypothesis, in that they show that the algorithm that learns from pairs of candidates performs much better for both data sets. In particular, the second algorithm achieved near optimal performance on the synthetic data set.

The third task is to infer the features of a proposal given the fact that it was the chosen bid. For example, suppose that the contracts in the TAC SCM domain specified minimal hardware configurations. Given a winning bid and a prior distribution over the possible configuration values, it might be possible for agents to infer that the RAM size feature of the winning bid was significantly larger than the minimal required RAM size. They could then tailor their future bids to include larger RAM components.

As a baseline, we present an importance sampling algorithm that weights the features of a candidate proposal by the probability that the object is the winner. We then discuss several alternative approximation algorithms that are much quicker than the baseline algorithm. As a result, many more samples may be generated in the same amount of time, leading to great increases in accuracy over the baseline.

While this paper focuses on the application of selection processes to negotiation, many domains in which a chosen element is selected out of a set of candidates can be modeled as selection processes. For example, the winner of a racing match can be modeled as the chosen candidate out of a set of participants. The features in this example might be skill, age and years of training. Another example might be choosing a spokesperson for a large company where the skills of the spokesperson are the features. We can thus infer that the chosen individual had good communication skills. Lastly, selection processes can also be of use to behavioral economists who wish to describe and predict how behavior differs across cultures and social contexts. The mixture models we present in this work can be used to capture how societies differ in their negotiation behavior.

2. RELATED WORK

Several works provided models for predicting the probability of acceptance of proposals in negotiation. Most of these studied automatic agents. Pardoe and Stone [19] used data obtained from past trading agent competitions to learn the probability that a customer will accept a given bid price. They estimated a separate probability distribution for each range within the possible bid prices and then converted these probabilities to a density function that was biased towards assigning a higher probability of acceptance to lower bids. Lawrence [14] used a naive Bayes classifier to learn the winning probability of seller bids given features relating to buyer characteristics (type of company, past profitability, etc...) The optimal seller price maximized its expected profit given the winning probabilities. All of these works learned a two-way decision process – whether to accept or reject a bid based on past history. We consider a different problem of choosing between multiple candidates where the likelihood of different candidates depends on the relationship between their features.

Das *et al*. [7] showed that computer agents playing stochastic strategies that are based on observed market data could outperform humans in a continuous double auction. This work compared between the performance of automatic agents with the collective performance of people and did not consider social factors. Our criteria of performance is not whether a computer agent can beat humans on average, but the predictive accuracy of a computer model that uses social preferences to predict *individual* bids of people. Also we use a different domain, in which multi-valued goods can be traded by players.

Because we represent social preferences in agents' utility functions, Our work is related to recent approaches for learning agents' utility functions. Ng and Russell [18] defined the inverse reinforcement learning problem as to infer an agent's utility function by observing its behavior. The main technique was to view the agent's decisions as defining a set of linear constraints on the utility functions. This approach relied explicitly on the assumption that the agent is fully rational, and therefore any decision constrains its utility function. Chajewska *et al.* [6], like Ng and Russell, learned a utility function by observing past decisions, but they also allowed a prior probability distribution over the space of utility functions. They also made the assumption that the agent is rational.

In contrast, we do not assume that a decision maker is rational, for two reasons. First, as stated in Section 1, people may choose actions that are contrary to their perceived utility functions [2]. We wish to model the behavior of real agents, so we need to allow for the possibility of mistakes. Second, choosing an action is only one kind of selection process, and we wish are approach to apply to selection processes in general, including those not resulting from a decision-making problem.

Lastly, Chajewska and Koller [5] used a probabilistic approach to learn the structure of utility functions from a database of partially

elicited utility functions of individuals. This is a different problem from the one we study, as we are not given any explicit information about the utility function, we are only told which action is selected.

3. SELECTION PROCESSES

We begin by presenting notation. There is a set of N data instances $d^{(1)}, \ldots, d^{(N)}$. Each instance $d^{(k)}$ represents one selection process, consisting of a set $m^{(k)}$ of candidate proposals $c_1^{(k)}, \ldots, c_{m^{(k)}}^{(k)}$ and a chosen bid that that is denoted as $c_*^{(k)}$. We usually drop the superscript (k) where unambiguous and write $c_1, \ldots, c_m$ for the set of candidates. Each candidate proposal c_j has a set of n features $\mathbf{x}_j = x_{j:1}, \ldots, x_{j:n}$, where each feature $x_{j:i}$ is a real number. Following our convention, we use $\mathbf{x}_*$ to denote the features of the chosen bid. In general, i will be used to index features, j will be used to index candidates within an instance, and k will be used to index instances.

Our task is to develop a model for selecting the chosen bid c_* from the set of candidate proposals in each instance. For any candidate, we let $S(c_j)$ denote the decision made for candidate proposal c_j according to the model. This decision can be to select or not to select the candidate as the chosen bid. Given the features of all the candidates, we assume that one of several types is used for making each decision. A type represents a particular way of making a selection, and there may be more than one way of deciding a particular selection process. For example, two types of agents in the TAC SCM domain can make different bids for supplying the same hardware configuration because they weigh the configuration features in different way.

One approach for representing this diversity in the model is to define a separate type for every instance. This captures the fact that each instance is a separate negotiation process that is independent from the other instances. In this case the number of types would have to equal the size of the training set, making it impossible to learn. Instead, we say that agents fall into a finite number of types. There is a prior probability distribution over types, denoted by $P(\tau)$. Once a type is selected according to $P(\tau)$ it is used to select the chosen bid. According to our model, the same type is used for making the decision for all candidate proposal in an instance. This is because it is the same agent that picks the chosen bid and at the same time rejects all the other proposals in that instance.

Associated with each type is a set of weights. The weight w_i^τ is the weight associated with feature i according to type τ. Given the candidate features, and the type, we compute a score for each candidate offer, which is a weighted sum of its feature values. Formally, we write u^τ for the scoring function associated with each type τ, and define

$$u_j^\tau = u^\tau(\mathbf{x}_j) = \sum_{i=1}^{n} w_i^\tau x_{j:i}$$

Now, consider first a simple selection process in which there are only two candidate proposals. For a fully rational decision maker, the chosen bid will be the proposal associated with the higher utility. To capture the fact that people sometimes make mistakes with respect to their utility function, we can allow noise to exist in the selection process, by making the probability of the chosen bid depend on the sigmoid function. That is

$$P(S(c) = \text{chosen}|\mathbf{x}, \tau) = \frac{1}{1 + e^{-u^\tau(\mathbf{x})}} \quad (1)$$

This function makes sense, because when the score is large and positive, the probability that a particular proposals is the chosen bid tends towards 1; when it is large and negative this probability tends towards 0. At 0, the proposer is completely indifferent between the two possibilities and has a probability of $\frac{1}{2}$ of choosing either. The closer the score is to 0, the more likely the proposer will make a mistake and choose the proposal with lower expected utility.

However, we need to generalize this to a selection process which involves more than two candidate proposals. To this end, we turn the scores of proposals into probabilities by making the probability that candidate c_j is selected to be the chosen offer be proportional to $e^{u_j^\tau}$. That is

$$P(S(c_j) = \text{chosen}|\mathbf{x}_1, \ldots, \mathbf{x}_m, \tau) = \frac{e^{u^\tau(\mathbf{x}_j)}}{\sum_{j=1}^{m} e^{u^\tau(\mathbf{x}_j)}} \quad (2)$$

This soft-max function is helpful in two ways. First, it is a generalization of the sigmoid model for the simpler model. Indeed, if the decision comes down to two candidates, we have

$$\begin{aligned} P(S(c_1) = \text{chosen}|S(c_1)\text{or}S(c_2) = \text{chosen}, \mathbf{x}_1, \mathbf{x}_2, \tau) &= \frac{e^{u_1^\tau}}{e^{u_1^\tau} + e^{u_2^\tau}} \\ &= \frac{1}{1 + e^{-(u_1^\tau - u_2^\tau)}} \end{aligned} \quad (3)$$

which is just the sigmoid function using the difference between scores of c_1 and c_2.

Second, by using the soft-max function, we make the probability of each candidate offer to depend not only on its own features, but also on the features of the other candidates which were not chosen.

4. LEARNING TO CHOOSE BIDS

The learning task is to simultaneously learn the prior distribution over type $P(\tau)$, as well as the feature weights $\mathbf{w}^\tau$ associated each type. We are given the features of each candidate, and a classification representing whether the candidate proposal was the chosen bid. We define mixture model over types, where each type is associated with a single-layer sigmoid belief network [17], but the type that generates each data instance is not observed. We present two algorithms for providing a mapping from the candidate features to a classification.

The first algorithm, called LEARN-SUPERVISED is shown in Figure 1. It uses supervised learning where each candidate offer is viewed as a separate training instance. The sigmoid function of Equation 1 is used to compute the predicted output $P(S(c_j))$, which is the likelihood that a particular candidate proposal is selected as the chosen bid. To learn the feature weights of the utility function associated with each type we use gradient descent. We make the true output 1 if the candidate was selected as the chosen bid and 0 if candidate was not selected. The error function to minimize for each candidate c_j is the absolute difference between the true output and the predicted output $P(S(c_j))$.

For each type, we compute the posterior probability $P(\tau \mid S(c_j))$ of generating the candidate c_j using Bayes rule. This probability is then used to adjust the learning rate for the gradient descent step. Intuitively, if a type is more likely to have been used for rejecting or selecting a candidate, that candidate should have more influence in learning the feature weights of that type. Similarly, if a type is unlikely to have been used for a candidate, the candidate should have little impact on the features weights. Therefore the learning rate is made proportional to the probability that the type was used to make this decision. Once the learning rate has been determined, a standard gradient descent update is performed.

The second algorithm views each data instance as consisting of *all* the possible candidate proposals, together with the identity of the selected candidate. The algorithm, called LEARN-PAIRS, is presented in Figure 2. It compares the features of the selected candidate c_* with those of every other candidate c_j, and treats the choice

Repeat until convergence of parameters:
  For each instance $d^{(k)}$:
    For each candidate $c_j \in d^{(k)}$:
      For each type τ:
        Let $u_j^\tau = \sum_{i=1}^n w_i^\tau x_{j:i}$
        If $S(c_j)$ = chosen, $P(S(c_j) \mid \tau, \mathbf{x}_j) = \frac{1}{1+e^{-u_j^\tau}}$
        Else $P(S(c_j) \mid \tau, \mathbf{x}_j) = 1 - \frac{1}{1+e^{-u_j^\tau}}$
      For each type τ:
        $P(\tau \mid S(c_j), \mathbf{x}_j) = \frac{P(S(c_j)|\tau,\mathbf{x}_j)P(\tau)}{\sum_\tau P(S(c_j)|\tau,\mathbf{x}_j)P(\tau)}$
        Let learning rate $\alpha_\tau = \alpha * P(\tau \mid S(c_j), \mathbf{x}_j)$
        $p = P(S(c_j) = \text{chosen} \mid \tau, \mathbf{x}_j) = \frac{1}{1+e^{-u_j^\tau}}$
        $p' = p * (1-p)$
        For each feature w_i^τ:
          If $S(c_j) = $ chosen
            $w_i^\tau \leftarrow w_i^\tau + \alpha_\tau * x_{j:i} * (1-p) * p'$
          Else
            $w_i^\tau \leftarrow w_i^\tau + \alpha_\tau * x_{j:i} * (-p) * p'$
  Reestimate $P(\tau) = \frac{\sum_{k=1}^N \sum_{j=1}^{m^{(k)}} P(\tau|S(c_j^{(k)}),\mathbf{x}_j)}{\sum_{k=1}^N m^{(k)}}$

Figure 1: Algorithm LEARN-SUPERVISED

Repeat until convergence of parameters:
  For each instance $d^{(k)}$:
    For each type τ:
      For each candidate $c_j \in d^{(k)}$, let $u_j^\tau = \sum_{i=1}^n w_i^\tau x_{j:i}$
      $P(c_*\text{chosen} \mid \tau, \mathbf{x}_1, \ldots, \mathbf{x}_m) = e^{u_*^\tau} / \sum_{j=1}^m e^{u_j^\tau}$
    For each type τ:
      $P(\tau \mid S(c_*) = \text{chosen}, \mathbf{x}_1, \ldots, \mathbf{x}_m) = \frac{P(S(c_*)=\text{chosen}|\tau,\mathbf{x}_1,\ldots,\mathbf{x}_m)P(\tau)}{\sum_\tau P(S(c_*)=\text{chosen}|\tau,\mathbf{x}_1,\ldots,\mathbf{x}_m)P(\tau)}$
      Let learning rate $\alpha_\tau = \alpha * P(\tau \mid S(c_*) = \text{chosen}, \mathbf{x}_1, \ldots, \mathbf{x}_m)$
      For each $c_j \in d^{(k)}$ where $c_j \neq c_*$:
        For each feature i, let $d_i = x_{*:i} - x_{j:i}$
        Let $u_{diff} = \sum_i w_i^\tau d_i = u_*^\tau - u_j^\tau$
        Let $p = P(S(c_*) = \text{chosen} \mid S(c_*)\text{or } S(c_j) = \text{chosen}, \mathbf{x}_*, \mathbf{x}_j)$
          $= 1/(1+e^{-u_{diff}})$
        $p' = p * (1-p)$
        For each feature w_i^τ:
          $w_i^\tau = w_i^\tau + \alpha_\tau * d_i * (1-p) * p'$
  Reestimate
    $P(\tau) = \frac{1}{N} \sum_{k=1}^N P(\tau \mid S(c_*^{(k)}) = \text{chosen}, \mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_{m^{(k)}}^{(k)})$

Figure 2: Algorithm LEARN-PAIRS

between them as a two-way decision. The utility to use in making the decision is the difference between the individual utilities u_*^τ and u_j^τ. Because the score is a linear function of the features, this difference is equal to the result of applying the scoring function u^τ to the differences between the features. Therefore, the predicted output is simply the probability of c_* being the chosen candidate, given that c_j is not chosen, which can be computed as

$$P(S(c_*) = \text{chosen} \mid c_* \text{ or } S(c_j) = \text{chosen}, \mathbf{x}_*, \mathbf{x}_j) = \frac{1}{1+e^{-(u_*^\tau - u_j^\tau)}}$$

The error function to minimize is always 1 minus this quantity, which is just applying Equation 3 to two decisions. The learning rate for the gradient descent step is computed in a similar fashion to algorithm LEARN-PAIRS.

Both algorithms need to compute the distribution over the type space. Because the actual type that generates each instance is unknown, we compute a new distribution over the type space by aggregating the likelihood of each and every data instance, and then normalizing. This process is similar in fashion to the Expectation Maximization algorithm. After all candidates for all instances have been processed, new values of $P(\tau)$ are estimated by the Bayes rule using the previously computed $P(\tau|S(c_j), \mathbf{x}_j)$.

4.1 Experimental Results

We evaluated the algorithms using data originating from the Colored Trails (CT) framework [11], in which two players must exchange resources in order to achieve their goals. Each game consists of a one-shot deal, in which one of the players, deemed the allocator, must make an offer to the other player, deemed the deliberator, which can in turn accept or reject the offer. The deliberator is not allowed to counter the allocator's offer with another proposal. The score that each player receives if no offer is made is identical to the score each player receives if the offer is rejected by the deliberator. We refer to this event as the *no negotiation alternative*. The score that each player receives if the offer is accepted by the deliberator is referred to as the *proposed outcome* score. The outcome of the game is determined by an offer-response pair, and each player's score in the game depends solely on his or her own performance. Games can differ in the resource players hold and the dependency relationships that hold between the players.

Each instance in our data set consisted of a CT game, in which the candidates were the set of possible proposals for the allocator in the game, and the chosen bid. There were about 30 candidates in each instance. Let NN_A and NN_D be the no-negotiation alternative scores for the allocator and deliberator, and PO_A and PO_D, the proposed outcome scores The properties of each proposal represents four social preferences that were computed as a function of players' scores: the individual benefit to the allocator if the proposal is accepted $PO_D - NN_D$; the joint benefit for both players $(PO_D + PO_A) - (NN_D + NN_A)$; the degree to which the outcome benefits the allocator more than the deliberator $PO_D - PO_A$; and the advantageousness of the trade to the allocator $(PO_D - NN_D) - (PO_A - NN_A)$.

We evaluated the algorithms using real data, generated by people playing CT games in a lab setting, and synthetic data, where each instance was generated by sampling a CT game, generating the set of candidate proposals for the game and choosing a bid according to a predetermined model. There were 169 instances in the real data set, whereas the synthetic data set included 1,000 instances. In both cases, we trained and tested the algorithms separately, using ten-fold cross validation. We evaluated the algorithms on a held-out data set using two metrics. First, "Likelihood" measures the average probability specified by the learned model that the chosen bid wins (the average is the geometric mean over all instances in the test set). Second, "Accuracy" measures the frequency with which the predicted bid, i.e. the candidate proposal with highest probability of winning, was actually the chosen candidate in the test set. With 30 candidates to choose from, we would expect both numbers to be quite small, but the relative sizes of the numbers indicate which algorithm performs better.

Table 1 compares the performance of LEARN-PAIRS to LEARN-SUPERVISED on real data, when using one, two and three types to learn a selection process. It also includes a naive predictor as a baseline that assumes a uniform probability distribution over all candidates for each instance. The relative sizes of the numbers indicate which algorithm performs better.

As shown in the table, while both algorithms performed better than the baseline, the performance of LEARN-PAIRS was consistently better than that of LEARN-SUPERVISED for both evaluation metrics. In addition, both algorithms improved in one metric as the number of types increased.

	Algorithm	Likelihood	Accuracy
	UNIFORM	0.0318	0.0375
One type	LEARN-SUPERVISED	0.0345	0.0562
	LEARN-PAIRS	0.0483	0.0812
Two types	LEARN-SUPERVISED	0.0344	0.0625
	LEARN-PAIRS	0.0591	0.0812
Three types	LEARN-SUPERVISED	0.0341	0.0688
	LEARN-PAIRS	0.0609	0.0812

Table 1: Performance comparison on real data

Table 2 evaluates performance of both algorithms on synthetic data. Here, we ran in an experiment in which a model of 3 types was used to generate the data and 3 types were used by the learning algorithms. We also report the performance of the actual model used to generate the data, which constitutes a gold standard. We ran experiments with many different numbers of types, both for the actual model and the learning algorithm, with comparable results. Results show that the likelihood of LEARN-PAIRS is much better than that of LEARN-SUPERVISED, and is close to the performance represented by the gold standard. Interestingly, both algorithms have the same accuracy as the gold standard. This suggests that simply predicting the chosen bid is an easier task than determining the probability of selecting each candidate as the chosen bid.

One possible explanation about this difference in performance of the algorithms in the CT domain is that LEARN-SUPERVISED ignores the dependencies between candidate proposals. It assumes that the probability a candidate is accepted depends only on its own features and not on the features of all the other candidates. In reality, if the other candidate proposals are exceptionally good, this candidate should be less likely to be the chosen bid, and conversely if they are bad. By ignoring this fact, LEARN-SUPERVISED is missing crucial information relevant to the selection of the chosen bid. Furthermore, the probabilities of acceptance for all candidates in an instance will in general not sum to 1, and a further drawback of LEARN-SUPERVISED is that a different type may be used for the decision for each candidate. In contrast, LEARN-SUPERVISED models the dependencies between candidates, and assumes the same type makes the decision for each instance.

Algorithm	Likelihood	Accuracy
UNIFORM	0.0320	0.0374
LEARN-SUPERVISED	0.0346	0.2212
LEARN-PAIRS	0.0603	0.2212
ACTUAL MODEL	0.0620	0.2212

Table 2: Performance comparison on synthetic data

5. INFERRING THE PROPERTIES OF A CHOSEN BID

In this section we consider the problem of inferring the properties of a proposal from the fact that it was the chosen bid. We assume now that the model is known, and that we have a prior probability distribution over the properties of any candidate proposal. For now, we do not assume that different candidates have the same prior. If a given proposal happens to be the chosen bid, it means that those property values that lead to a higher score are more likely. The task is to obtain the posterior distribution over the properties of a target proposal, given that it is the chosen bid.

As a baseline, we propose a simple importance sampling method to achieve this task. Let the target candidate be c_1. We begin by sampling from the prior distribution for each candidate including the target, to obtain features $\mathbf{x}_1$ for the target and $\mathbf{x}_j$ for each of the other candidates c_j. We also sample the type τ used to make the decision from the probability distribution $P(\tau)$. We compute the score for each candidate $u_j^\tau = \sum_i w_i^\tau x_{j:i}$. We then compute the probability $p = \frac{e^{u_1^\tau}}{\sum_{j=1}^m e^{u_j^\tau}}$ that the target candidate was chosen. We then weight the sample consisting of the features $\mathbf{x}_1$ by the probability p. We thus obtain a set of weighted samples that is an approximation to the posterior distribution over features of the target candidate. This set of samples can be used to obtain expectations of the features we are interested in.

This method, which we call IMPORTANCE-SAMPLE, requires sampling all the candidate proposals. We can do better by making the observation that the probability that a candidate is the chosen bid depend most heavily on its own score. Once the type has been fixed, and the score of the target candidate has been determined, we can attempt to estimate the probability that the target is the chosen bid without sampling the other candidates. This approach has the advantage that because only one candidate proposals needs to be sampled, many more samples can be taken for the same running time. On the other hand, because this approach uses an estimate of the probability that the target candidate is the winner, rather than sampling, it is biased. The degree of the bias depends on the quality of the estimates.

There are several ways to estimate the probability that the target proposal is the chosen bid, given its features. We wish to estimate

$$P(c_1 \text{ wins}|\mathbf{x}_1, \tau) = E_{\mathbf{x}_2,\ldots,\mathbf{x}_m}[\frac{e^{u^\tau(\mathbf{x}_1)}}{e^{u^\tau(\mathbf{x}_1)} + \sum_{j=2}^m e^{u^\tau(\mathbf{x}_j)}}] \quad (4)$$

where $E_{\mathbf{x}_2,\ldots,\mathbf{x}_m}[f]$ denotes the expectation of f with respect to $\mathbf{x}_2, ..., \mathbf{x}_m$. This probability is hard to estimate directly without sampling $\mathbf{x}_2, \ldots, \mathbf{x}_m$ many times for each $\mathbf{x}_1$, which is precisely what we wish to avoid. However, we can hope to approximate it, either with a method that does not sample $\mathbf{x}_2, \ldots, \mathbf{x}_m$ at all, or with a method that only samples $\mathbf{x}_2, \ldots, \mathbf{x}_m$ a fixed number of times, and amortizes that sampling over all samples of $\mathbf{x}_1$. Our first approximation comes from the consideration that in the case where c_1 is unlikely to be the winner, $e^{u^\tau(\mathbf{x}_1)}$ will usually be small compared to $\sum_{j=2}^m e^{u^\tau(\mathbf{x}_j)}$. Therefore the ratio $\frac{P(c_1^{(1)} \text{ wins}|\mathbf{x}_1^{(1)},\tau)}{P(c_1^{(2)} \text{ wins}|\mathbf{x}_1^{(2)},\tau)}$ of the probability the target candidate wins for two different samples $\mathbf{x}_1^{(1)}$ and $\mathbf{x}_1^{(2)}$ will usually be approximately equal to

$$\frac{E_{\mathbf{x}_2,\ldots,\mathbf{x}_m}[\frac{e^{u^\tau(\mathbf{x}_1^{(1)})}}{\sum_{j=2}^m e^{u^\tau(\mathbf{x}_j)}}]}{E_{\mathbf{x}_2,\ldots,\mathbf{x}_m}[\frac{e^{u^\tau(\mathbf{x}_1^{(2)})}}{\sum_{j=2}^m e^{u^\tau(\mathbf{x}_j)}}]} = \frac{e^{u(\mathbf{x}_1^{(1)})}}{e^{u(\mathbf{x}_1^{(2)})}} \quad (5)$$

Since the only thing that is required to get good weighted samples is that the ratios between the weights be correct, we can simply use $e^{u^\tau(\mathbf{x}_1^{(k)})}$ as the weight of the kth sample. The scheme that uses these weights is called SAMPLE-SIMPLE.

However, this is a very crude approximation. In particular, the approximation is inaccurate in the cases where $e^{u^\tau(\mathbf{x}_1)}$ is large. In such cases, the weights will be biased upwards, because we are ignoring the $e^{u^\tau(\mathbf{x}_1)}$ term in the denominator of Equation 4 which

is no longer negligible. A better approximation can be obtained by not ignoring the $e^{u^\tau(\mathbf{x}_1)}$ term in the denominator. Instead, we approximate Equation 4 by

$$\frac{e^{u^\tau(\mathbf{x}_1)}}{e^{u^\tau(\mathbf{x}_1)} + E_{\mathbf{x}_2,\ldots,\mathbf{x}_m}[\sum_{j=2}^{m} e^{u^\tau(\mathbf{x}_j)}]} \quad (6)$$

This scheme, called SAMPLE-EXPECT, uses a preprocessing step in which a number of samples of $\mathbf{x}_2, ...\mathbf{x}_m$ are taken to estimate $E_{\mathbf{x}_2,\ldots,\mathbf{x}_m}[\sum_{j=2}^{m} e^{u^\tau(\mathbf{x}_j)}]$ for each type τ. The estimate for the selected type is then substituted in Formula 6 to obtain the approximate weight for sample $\mathbf{x}_1$.

Of course, simply moving the expectation inwards is likely to produce an inaccurate approximation. In particular, the expectation of Equation 4 is likely to be dominated by cases where $\sum_{j=2}^{m} e^{u^\tau(\mathbf{x}_j)}$ is small, not by the expected value of $\sum_{j=2}^{m} e^{u^\tau(\mathbf{x}_j)}$. This suggests a different approximation scheme, where we use small values of $\sum_{j=2}^{m} e^{u^\tau(\mathbf{x}_j)}$. In this scheme, called SAMPLE-MIN, we take a certain number of samples of $\mathbf{x}_2, ..., \mathbf{x}_m$ in a preprocessing step. From those samples we take the minimum value of $\sum_{j=2}^{m} e^{u^\tau(\mathbf{x}_j)}$ for each type. We use that value for the selected type instead of the expectation in Equation 6 to compute the weight for sample $\mathbf{x}_1$. An interesting question in this scheme is how many samples of $\mathbf{x}_2, ...\mathbf{x}_m$ to take in the preprocessing step. Even if we had infinite time, taking more samples would not necessarily be a good thing, because the minimum may become too small, yielding a less accurate approximation.

Thus far, we have assumed there is a particular target proposal whose features we want to infer. But another kind of inference problem is to infer the features of a generic proposal, given the fact that it was chosen. For example, the runners in a race might include three runners whom we know by name and who have a reputation, as well as fifteen other anonymous, generic runners. Suppose now that we want to infer the posterior probability distribution over the properties of an anonymous candidate. One approach would be simply to choose one of the anonymous candidates to be our target candidate, and proceed using one of the methods described earlier. But this would be unnecessarily wasteful — it would ignore the probability that one of the other anonymous candidates is chosen as the winner. In the extreme case, all the candidates have the same prior distribution over features. In that case the probability that one of the anonymous candidates is selected is 1, and every sample should have a weight of 1.

A better approach, assuming there is a reasonable number of anonymous candidates, is to use a rejection sampling method, which we call SAMPLE-GENERIC . We sample a type from $P(\tau)$, and features for each of the candidates, as in the baseline IMPORTANCE-SAMPLE method. We then select a winning candidate using the model. If the winner is an anonymous candidate with the prior we are interested in, we keep its features as a sample. If not, we reject the sample.

It is worth examining the advantage of this approach more closely. The advantage over IMPORTANCE-SAMPLE is indeed what we have stated, that each sample can be expected to have a much higher weight. The advantage of this approach over the estimated schemes SAMPLE-SIMPLE, SAMPLE-EXPECT and SAMPLE-MIN is not that the samples taken have higher weight, since with those schemes more samples could be taken, enough so that their total weight is equal to that of SAMPLE-GENERIC. Rather, it comes from the fact that SAMPLE-GENERIC does not use a biased approximation of the probability that the target candidate wins.

5.1 Experimental Results

We first tested the algorithms for inferring the properties of a winning candidate using synthetic data, where there is only one feature x, one type, and the scoring function is simply $u = x$. The prior over the feature is a Gaussian.

		Probability that c_1 wins		
		0.001	0.01	0.1
Algorithm	Samples	Error	Error	Error
IMPORTANCE-SAMPLE	100	0.1770	0.1689	0.1454
SAMPLE-SIMPLE	1000	0.0576	0.0610	0.2412
SAMPLE-EXPECT	900	0.0828	0.0586	0.0554
SAMPLE-MIN	900	0.0680	0.0577	0.0806

Figure 3: Relative error rates for the different algorithms on three experiments, varying the probability that c_1 wins

We considered three cases, varying the probability that the target candidate c_1 will be the winning candidate from 0.001 to 0.1. Figure 3 shows the results for an experiment in which the number of samples was chosen to ensure that each algorithm had the same running time. We report the relative error of the estimated expected value of the feature x of c_1 compared to an estimate of ground truth obtained by taking 1,000,000 samples and keeping the ones where c_1 wins. The results are averaged over 10,000 runs. Also shown is the number of samples used by each algorithm.

We see that in the first case, where there is a very low probability of c_1 winning, algorithm SAMPLE-SIMPLE is a very good approximation and has the best performance. On the other hand, in the third case where the probability is quite high, SAMPLE-SIMPLE performs very poorly indeed. These results are as one would expect, because the approximation of Equation 5 assumes that the score of the target candidate is very small relative to the scores of the other candidates. Meanwhile, algorithms SAMPLE-EXPECT and SAMPLE-MIN, which do not make this assumption, significantly outperform IMPORTANCE-SAMPLE in all three cases. SAMPLE-MIN is slightly better for low probabilities of c_1 winning and SAMPLE-EXPECT is better for high probabilities.

Algorithm	Signed Error	Absolute Error
IMPORTANCE-SAMPLE	-0.0069	0.0232
SAMPLE-SIMPLE	0.0242	0.0277
SAMPLE-EXPECT	0.0257	0.0259
SAMPLE-MIN	0.0239	0.0239
SAMPLE-GENERIC	-0.0062	0.0212

Figure 4: Signed and absolute errors for the five algorithms on the Colored Trails model

Finally, we compare the algorithms, and also SAMPLE-GENERIC, on a richer model learned from the Colored Trails data. Each instance now consisted of the candidate proposals for a Colored Trails game. Because all proposals have the same prior, we can use SAMPLE-GENERIC here. Note that now the candidates are correlated with each other — if a game has one high-scoring proposal it is likely to have more. This fact will hurt the algorithms that only sample c_1, because they will not discover the correlations between candidates. The results are shown in Figure 4. The figure shows both the average signed errors, and the average absolute errors, of the estimate of the individual utility feature of the winning proposal, taken over 100 runs. SAMPLE-GENERIC is the best performer. Examining the signed errors, we see that they are very high and positive for all three algorithms that only sample c_1. These three algorithms are consistently overestimating, and are suffering from an upward bias

which causes them to underperform the other two algorithms even though they use many more samples.

6. CONCLUSION AND FUTURE WORK

We presented a model used to predict selection processes, and two learning algorithms for learning the parameters of a selection process from data. We showed that a model which reasons about dependencies between candidates is superior to one that does not. We also presented several methods for inferring the properties of a candidate from the fact that it won, and showed that algorithms that only sample the target candidate perform much better than an algorithm that samples all candidates.

In other domains, it is possible for zero or more than one candidates to be chosen but the choice of one candidate does have a bearing on whether other candidates are chosen. For example, consider hiring a candidate to fill a faculty position. If the search turns up more than one outstanding candidate, more than one offer will be made. If the committee feels that there are no good candidates, no offer will be made. But in general the candidates are in competition with each other, and if an offer is made to one candidate that reduces the probability that one will be made to another candidate. In future, we intend to model these types of processes.

Selection processes can naturally be accommodated in probabilistic models. There is a close relationship between selection processes and probabilistic relational models (PRMs) [9]. In a PRM, an object X may be related to another object Y via a complex attribute. We may have uncertainty as to which other object Y the object X is related to. This is known as reference uncertainty. In a PRM, this uncertainty has been traditionally modeled by specifying an explicit probability distribution over the possible objects Y. The distribution over possible objects traditionally does not depend on the attributes of the possible objects. However, it is natural to make the choice of possible object Y a selection process, which depends on the attributes of the possible objects. For example, we may have a *Job* object. This object will have a complex attribute *Candidate*, which is a multi-valued attribute. The *Job* will be related to a number of *Person* objects via the *Candidate* attribute. In addition, the *Job* object will have an *Occupant* attribute, which is single-valued. We have reference uncertainty with regards to the value of the *Occupant* attribute. Modeling *Occupant* as a selection process, will make the candidates the values of the *Candidate* attribute, and the winner depends on the features of the candidates.

7. ACKNOWLEDGMENTS

We thank Barbara Grosz, Adam Juda and Loizos Michael for helpful comments and discussions. This work was supported in part by NSF grant CNS-0453923. Development and dissemination of the Colored Trails formalism is supported in part by DARPA.

8. REFERENCES

[1] R. Arunachalam, J. Eriksson, N. Finne, S. Janson, and N. Sadeh. The TAC supply chain management game. Technical report, Swedish Institute of Computer Science, 2003.

[2] M. Bazerman. *Judgment in Managerial Decision Making*. Wiley, 2001.

[3] C. Camerer. *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton University Press, 2003.

[4] J. Cavano. Computers as partners: A technology forecast for decision-making in the 21st century. In *Proceedings of the Twenty-Third Annual International Computer Software and Applications Conference*, 2001.

[5] U. Chajewska and D. Koller. Utilities as random variables. In *Proc. 16th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2001.

[6] U. Chajewska, D. Koller, and D. Ormoneit. Learning an agent's utility function by observing behavior. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.

[7] R. Das, J. E. Hanson, J. O. Kephart, and G. Tesauro. Agent-human interactions in the continuous double auction. In *IJCAI*, pages 1169–1187, 2001.

[8] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1), 1977.

[9] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proc. 16th International Joint Conference on Artificial Intelligence (IJCAI)*, 1999.

[10] Y. Gal, A. Pfeffer, F. Marzo, and B. Grosz. Learning social preferences in games. In *Proc. 19th National Conference on Artificial Intelligence (AAAI)*, 2004.

[11] B. Grosz, S. Kraus, S. Talman, and B. Stossel. The influence of social dependencies on decision-making. Initial investigations with a new game. In *Proc. 3rd International Joint Conference on Multi-agent systems (AAMAS)*, 2004.

[12] J. Henrich, R. Boyd, S. Bowles, H. Gintis, E. Fehr, and C. Camerer, editors. *Foundations of Human Sociality: Economic Experiments and Ethnographic Evidence in Fifteen Small-Scale Societies*. Oxford University Press, 2004.

[13] J. Kagel and A. Roth, editors. *The hanbook of experimental economics*. Princeton University Press, 1995.

[14] R. Lawrence. A machine-learning approach to optimal bid pricing. In *Proceedings of the Eighth INFORMS Computing Society Conference on Optimization and Computation in the Network Era, Arizona, 2003. 6*, 2003.

[15] G. Loewenstein, M. Bazerman, and L. Thompson. Social utility and decision making in interpersonal contexts. *Journal of Personality and Social psychology*, (57)(3):426–441, 1989.

[16] J. MacKie-Mason, A. Osepayshivili, D. Reeves, and M. Wellman. Price prediction strategies for market-based scheduling. In *Proc. of 18th International Conference on Automated Planning and Scheduling*, June 2004.

[17] R. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113, 1992.

[18] A. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 2000.

[19] D. Pardoe and P. Stone. Bidding for customer orders in TAC SCM. In P. Faratin and J. Rodriguez-Aguilar, editors, *Agent Mediated Electronic Commerce VI: Theories for and Engineering of Distributed Mechanisms and Systems (AMEC 2004)*, volume 3435 of *Lecture Notes in Artificial Intelligence*, pages 143–157, Berlin, 2005. Springer Verlag.

[20] R. Schapire, P. Stone, D. McAllester, M. Littman, and J. Csirik. Modeling auction price uncertainty using boosting-based conditional density estimation, 2002.

[21] P. R. Wurman, M. P. Wellman, and W. E. Walsh. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In K. P. Sycara and M. Wooldridge, editors, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 301–308, New York, 1998. ACM Press.

Support-based Distributed Search

A new approach for multiagent constraint processing

Peter Harvey
Decision Systems Laboratory
University of Wollongong
NSW, Australia
pah06@uow.edu.au

Chee Fon Chang
Decision Systems Laboratory
University of Wollongong
NSW, Australia
c03@uow.edu.au

Aditya Ghose
Decision Systems Laboratory
University of Wollongong
NSW, Australia
aditya@uow.edu.au

ABSTRACT

Distributed Constraint Satisfaction Problems provide a natural mechanism for multiagent coordination and agreement. To date, algorithms for Distributed Constraint Satisfaction Problems have tended to mirror existing non-distributed global-search or local-search algorithms. Unfortunately, existing distributed global-search algorithms derive from classical backtracking search methods and require a total ordering over agents for completeness. Distributed variants of local-search algorithms (such as distributed breakout) inherit the incompleteness properties of their predecessors, or depend on the creation of new communication links between agents. This paper presents an algorithm in which a global ordering is not required, while avoiding the problems of existing local-search algorithms.

1. INTRODUCTION

Constraint Satisfaction Problems (CSPs) have proven applicable in a wide variety of domains. A CSP is classically defined by a set of variables $\mathcal{V}$, a domain for each variable $\mathcal{D}_v$, and a set of constraints $\mathcal{C}$. A *solution* to a CSP is a complete assignment of values to variables which satisfies every constraint.

A Distributed CSP (DisCSP) is formed when the description and solution procedure of a CSP are separated amongst multiple agents. The distributed environment extends the applicability of CSPs to domains such as distributed scheduling and resource contention. Of particular interest is the use of DisCSPs as models for solving other multiagent problems, with DisCSP algorithms defining a protocol for agent communication. Common examples of such problems include scheduling, task assignment, and limited forms of negotiation where simple decision(s) must be made per agent. A DisCSP can be constructed by representing each of the agents decisions as a variable, with constraints describing any inter-agent relationships.

A DisCSP can be solved by distributed variants of existing global-search or local-search algorithms. However, local-search algorithms [7] are incomplete in both the distributed and non-distributed case. Distributed variants of global-search [1, 3, 5, 6] presented to date make use of a total order over variables.

We argue that any total order impacts the characteristics of backtracking-style search in undesirable ways for use in many multiagent problems. For example, an agent which has a 'higher' rank in the ordering has more 'authority' and is therefore less likely to change its value than a 'lower' ranking agent. In an anytime environment this results in higher-ranked agents being granted more stable answers. While some problems may desire such behaviour our concern lies with those problems which do not.

We also argue that, when using a total order, it is difficult to add constraints between two previously independent DisCSPs. To do so would require a re-computation of the variable ordering and/or an arbitrary decision that one DisCSP ranks higher than the other. If a problem is frequently altered by the addition of groups of variables, as is likely to occur in large DisCSP networks, global re-computation will become increasingly difficult. If variable ordering is instead made arbitrarily (for example, ordering by variable identifier) the problem of stability is exacerbated.

These arguments can be demonstrated in an example large-scale meeting scheduling problem described below.

Example. The universities of Pluto and Saturn each use an automated system for scheduling meetings amongst their own staff. The staff give constraints of the form 'Alice needs to meet with Bob for 2 hours this Wednesday or Thursday' to agents on their own computers. Individual universities contain a large number of staff with generally sparse connections, so a distributed algorithm is used in which agents communicate directly with each other. Agents are assigned comparable identifiers using finely-tuned schemes specific to each university. These identifiers are chosen to permit backtracking in a distributed global-search algorithm within the university.
Despite best efforts at fairness, a static ordering creates problems between research peers. If the ordering states 'Alice ranks higher than Bob' then any trivial change in Alice's meeting times must always be accepted by Bob. Inversely, Bob may request a change to Alice's meetings only after exhausting all possible meeting schedules and detecting infeasibility. This problem is distinct from that of preference orderings, and instead relates to stability (ie. Alice's schedules are more stable than Bob's).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

To worsen matters, Bob from Pluto wishes to arrange a meeting with Carla from Saturn. Their identifiers, while still possibly unique, are not meaningfully comparable for the purpose of a backtracking search. To continue using any existing distributed algorithm, we must be able to compare identifiers between agents operating at Pluto's and Saturn's universities. An example solution is to decide that all Saturn's identifiers are 'greater' than Pluto's identifiers. Unfortunately this would have the same impact on the behaviour of the algorithm as outlined above - meeting schedules for researchers from Pluto would become subservient to those from Saturn. Any changes in meeting times for Saturn's researchers, no matter how trivial, must be accepted by Pluto's researchers.
Furthermore, any decision for resolving the variable order would require intervention by authorities at each university or the use of a heuristic method such as DisAO [1, 3]. While these decisions can be made for pairs or sets of universities, it does not scale well computationally. For example, if Dennis was an independent researcher he must establish 'comparability' with each university and all other researchers. The addition of new researchers frequently raises the possibility of frequent re-computation of variable ordering.

The above example highlights problems that arise from using a total order to establish 'authority' or 'importance' between agents, and maintaining a total order subject to merging of previously independent DisCSPs. The computational disadvantages of a total order were also noted and addressed within the development of Asynchronous Weak-Commitment Search (AWCS) [5, 6]. However, AWCS creates additional links between variables, which we believe is undesirable in large-scale meeting scheduling. The specific difficulties of large-scale distributed meeting scheduling motivated us to develop an algorithm which:

- has no need for 'authority' between variables, effectively avoiding the need for a total order on variables.
- provides fairness in the level of stability for variables.
- does not add links between variables, avoiding the eventual need for 'broadcasting' assignments.
- addresses the risk of cyclic behaviour exhibited by local search algorithms.

Section 2 will present a model of a simple meeting scheduling problem as a DisCSP, and present how *arguments* can form the basis of communication between agents. Section 3 will describe the internal decision processes of each agent to handle arguments in an appropriate manner. Section 4 will present analysis of the algorithm.

2. MODELLING ARGUMENTS

We begin with a simple example demonstrating how a distributed constraint satisfaction problem can be solved through arguments. We will then describe a formal model of this problem with corresponding notation able to represent the dialogue. This transformation serves as inspiration for a new distributed constraint satisfaction algorithm.

Example. Alice, Bob, Carla and Dennis are attending a conference and must organise meeting times for themselves:

- Bob must meet with Carla.
- Bob must meet with Alice before meeting with Carla.
- Dennis must meet with Alice.
- Bob, Carla and Dennis must have a separate group meeting.
- Available times are 1pm, 2pm and 3pm.
- Double-booking for meeting time-slots is not allowed.
- Each person knows only those meetings that they need to attend.
- No person has any 'authority' over any other.
- Communication can only occur between people who already know each other.

To find the solution they state arguments (proposals and rejections) in turn, providing further detail if two arguments are contradictory or if they alter a previous argument:

Alice to Dennis	$\Rightarrow$	I propose a 1pm meeting
Dennis to Carla	$\Rightarrow$	I propose a 2pm group meeting
Dennis to Bob	$\Rightarrow$	I propose a 2pm group meeting
Carla to Bob	$\Rightarrow$	I propose a 1pm meeting
Alice to Bob	$\Rightarrow$	I propose a 2pm meeting
Bob to Alice	$\Rightarrow$	I have a group meeting at 2pm, so I propose a 1pm meeting instead
Bob to Carla	$\Rightarrow$	I completely reject your proposal, so I propose a 3pm meeting instead
Alice to Dennis	$\Rightarrow$	I now have another meeting at 1pm, so I propose a 3pm meeting instead

2.1 Constraint Model

To construct a distributed constraint satisfaction problem we translate the time of attending a meeting for each person into a variable. Equality constraints are used to ensure meeting times are agreed to by all users. For example, a pair of variables a and b may represent the scheduled time of the meeting between Alice and Dennis. The constraint $a = b$ is interpreted as 'the time Alice decides to meet with Dennis must be the same as the time that Dennis decides to meet with Alice'. Inequality constraints ensure that meetings occur at distinct times.

Note that there is significant redundancy in the constraints and variables as presented in Figure 1. This occurs as the constraints upon one person are not automatically known to others. Relaxing this requirement would generate a simpler constraint graph, but would conflict with our aim to solve in a distributed manner. Using this constraint model as an example, we will now define suitable notation for representing the dialogue.

$$\mathcal{V} = \{a, b, c, d, e, f, g, h, i\}$$

$$\mathcal{D} = \{1\text{pm}, 2\text{pm}, 3\text{pm}\}$$

$$\mathcal{C} = \left\{ \begin{array}{ll} a = b & a \neq i \\ b \neq c & c = d \\ c = g & d \neq e \\ d = g & e = f \\ f \neq g & f > h \\ g \neq h & h = i \end{array} \right\}$$

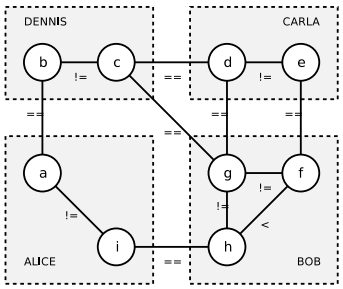


Figure 1: Example of a constraint model and graph

2.2 Argument Model

To represent proposals and rejections as outlined above, we require two distinct message types; isgoods and nogoods. These will be the only message types required to construct our algorithm.

An **isgood** is an ordered partial assignment for a sequence of connected variables, and so represents a 'proposal'. Consider the argument in our example where Bob says to Alice: "I already have a group meeting at 2pm, so I propose a 1pm meeting for us instead". This is a proposal, and so can be written as an ordered partial assignment or 'isgood':

$$\langle (g, 2\text{pm}), (h, 1\text{pm}) \rangle$$

This isgood is read as "variable g took on value 2pm, and so h took on value 1pm". Note that variables in an isgood must be connected to their immediate predecessor, and therefore $\langle (d, 2\text{pm}), (h, 1\text{pm}) \rangle$ is *not* an isgood. Also note that we use the operator $+$ to represent the appending of a variable assignment to an isgood. For example, $\langle (g, 2\text{pm}), (h, 1\text{pm}) \rangle + (i, 1\text{pm}) = \langle (g, 2\text{pm}), (h, 1\text{pm}), (i, 1\text{pm}) \rangle$.

A **nogood** is an unordered partial assignment which is provably not part of a solution, and so represents a 'rejection'. Consider the argument in our example where Bob says to Carla: "I reject your proposal, and I propose a 3pm meeting for us instead" This is a rejection (he must meet Carla before Alice, so 1pm is not a possible meeting time) followed by a proposal, which written in sequence are:

$$\{(e, 1\text{pm})\} \text{ and } \langle (f, 3\text{pm}) \rangle$$

They are read as "variable e cannot take value 1pm" and "variable f took on value 3pm" respectively. As demonstrated in our example, a nogood is usually accompanied by an isgood.

We say that a constraint is **satisfied** by an isgood I if the constraint is explicitly not violated by the assignments in I. Testing whether a constraint is satisfied is therefore only possible if all variables appearing in the constraint also appear in I. Similarly, a nogood is satisfied if it is not a subset of the assignments in I. For example, given an isgood $I = \langle (g, 2\text{pm}), (h, 1\text{pm}) \rangle$, we know:

- $I + (i, 2\text{pm})$ does not satisfy the constraint $h = i$
- $I + (i, 1\text{pm})$ does satisfy the constraint $h = i$
- I does not satisfy the nogood $\{(h, 1\text{pm})\}$
- I does satisfy the nogood $\{(h, 1\text{pm}), (i, 1\text{pm})\}$

Thus, given a set of constraints and a set of nogoods, we say that an assignment (v, d) is **consistent** with respect to an isgood I iff each constraint on v and each nogood is satisfied by $I + (v, d)$.

By the notation $|I|$ we indicate the number of tuples in I. Finally, we will write $I \sqsubseteq I'$ to indicate that I is a **sub-isgood** of I'. A sub-isgood is the tail (or entirety) of another isgood. For example:

$$\begin{aligned} \langle (i, 1\text{pm}) \rangle &\sqsubseteq \langle (h, 1\text{pm}), (i, 1\text{pm}) \rangle \\ &\sqsubseteq \langle (g, 2\text{pm}), (h, 1\text{pm}), (i, 1\text{pm}) \rangle \\ \langle (i, 2\text{pm}) \rangle &\not\sqsubseteq \langle (h, 1\text{pm}), (i, 1\text{pm}) \rangle \\ \langle (h, 2\text{pm}) \rangle &\not\sqsubseteq \langle (h, 1\text{pm}), (i, 1\text{pm}) \rangle \end{aligned}$$

3. SOLVING WITH ARGUMENTS

Using the above notation, and the dialogue of our example as a guide, it is possible to construct a distributed search algorithm in which agents will:

- send and receive proposals (isgoods) and rejections (nogoods)
- convince neighbours to change their value by sending progressively longer proposals
- reject a proposal from a neighbour if it is inconsistent
- justify their variable assignment by the proposal of just one neighbour
- communicate only with agents for which they share a constraint

To achieve this, each agent records the most recent proposals sent/received by neighbouring agents and an unbounded nogood store. Unlike other distributed algorithms, SBDS does not regard all information from neighbours as a consistent 'agent view'. Instead, the isgood received from just one neighbour is chosen as justification for our current assignment and combines to form our 'agent view'. Formally, the information stored by each agent is:

- *sent*(v) - last isgood sent to each neighbour v
- *recv*(v) - last isgood received from each neighbour v
- *nogoods* - set of all nogoods ever received
- *support* - the neighbour chosen for our 'agent view'
- *view* - current agent view (*recv*(*support*) extended by an assignment to our own variable)

Procedure 1 *main* ()

```
1: while true do
2:    for all received nogoods N (in fifo order) do
3:       receive-nogood(N)
4:    for all received isgoods I (in fifo order) do
5:       receive-isgood(I)
6:    select-support()
7:    for all neighbours v do
8:       send-isgood(v)
9:    wait until at least one message in the queue
```

Procedure 2 *receive-isgood* (I)

```
1: let v be the variable which sent I
2: set recv(v) to I
3: if no choice of value is consistent wrt recv(v) then
4:    send-nogood(v)
```

Procedure 3 *receive-nogood* (N)

```
1: if N in nogoods then
2:    break, as this nogood was already known
3: add N to nogoods
4: if no value is consistent then
5:    terminate algorithm
6: for all neighbours v do
7:    if no choice of value is consistent wrt recv(v) then
8:       send-nogood(v)
```

Procedure 4 *select-support* ()

1: *update-view* ()
2: **if** our current value is inconsistent wrt some $recv(v)$ and $|recv(v)| \geq |view|$ **then**
3: **set** *support* to a neighbour u, maximising $|recv(u)|$
4: *update-view* ()

Procedure 5 *update-view* ()

1: **let** $view'$ be $recv(support)$ extended by a consistent assignment to self, and maximal with respect to $\prec$
2: **let** v be the first variable assigned in $view'$
3: **if** $scope(view) \neq scope(view')$ **or** $view \prec view'$ **or** the assignment of v is equal in $view'$ and $recv(v)$ **or** the assignment of v is unequal in $view$ and $recv(v)$ **then**
4: **set** $view$ to $view'$

Procedure 6 *send-nogood* (v)

1: **let** N be an inconsistent subset of $recv(v)$
2: **send** N to v
3: **set** $recv(v)$ to $\langle\,\rangle$
4: **if** $support = v$ **then set** *support* to *self*

Procedure 7 *send-isgood* (v)

1: **if** our current value is consistent wrt $recv(v)$ **and** $sent(v) \sqsubseteq view$ **then**
2: **break**, as a new isgood is not necessary
3: **lock** communication channel with v
4: **if** there are no unprocessed isgoods from v **then**
5: **let** L be $\min(\max(|recv(v)|, |sent(v) + 1|), |view|)$
6: **let** I be an isgood such that $I \sqsubseteq view$ and $|I| = L$
7: **send** I to v
8: **set** $sent(v)$ to I
9: **unlock** communication channel with v

The *main* loop of our algorithm processes all messages before choosing *support* and *view* and sending new isgoods. Incoming isgoods are stored in $recv(v)$ by the *receive-isgood* procedure. If no assignment to our own variable is consistent with respect to the new isgood and current known nogoods, the procedure *send-nogood* is called to derive and send a nogood. Similarly the *receive-nogood* procedure handles an incoming nogood; each $recv(v)$ is re-tested for consistency, and *send-nogood* is called if appropriate.

The *select-support* procedure determines which neighbouring variable will be considered as our support for this iteration. A new support must be chosen if a received isgood from a neighbour is longer than our current *view* and conflicts with our current value.

The *update-view* procedure refreshes the current *view* according to the isgood $recv(support)$. In most cases *update-view* will replace *view* by selecting and appending a consistent assignment for our variable to the tail of $recv(support)$. As our algorithm is asynchronous, and agents can determine their assignments simultaneously, there is the possibility of cyclic behaviour. This is corrected in existing distributed search algorithms by the use of a total order on agents; a change by a lower ranked agent cannot induce a change in a higher ranked agent. In place of a total order on agents, we use orderings for isgoods defined over the same set of variables. These orderings are only used in a situation where a cycle is deemed likely. Being independent they can contain no bias towards any agent or value.

Formally, let $scope(I)$ be the sequence of variables in the isgood I. For example, with $I = \langle(c, 1\text{pm}), (b, 2\text{pm})\rangle$ we have $scope(I) = \langle c, b\rangle$. We assume an ordering $\prec$ is known to all agents and is total for isgoods of the same *unordered* scope.[1] We will not replace a *view* with $view'$ if each of the following is true:

- *view* is not out-of-date (ie. the most recently received assignment for v is the same as that presented in *view*)
- $view'$ is out-of-date (ie. the most recently received assignment for v is different from that presented in $view'$)
- $view'$ may be part of a cycle (ie. the ordered scopes of *view* and $view'$ are identical, and the first assignment is to a neighbour v)
- $view'$ is lower in the ordering (ie. $view' \prec view$)

This scheme causes an agent to postpone changing its value if its new view would be out-of-date, would propagate a cycle, and the old view is regarded as 'superior' by the ordering. As the definition of the ordering is uniform across all agents, any cyclic behaviour will quickly resolve in favour of a single view. Theorem 1 contains a formal statement and proof of this result.

The *send-nogood* procedure generates and sends an appropriate nogood when a received isgood is found to be inconsistent. The nogood may be any inconsistent subset of the assignments in the isgood. An interesting effect of this procedure is that nogoods are formed only from variables in a sequence, rather than 'all neighbours' as occurs in AWCS.

The *send-isgood* procedure constructs the strongest possible isgood to send to agent v while satisfying certain 'minimality' requirements. If there is no need to send an isgood (ie. we have already established agreement with our neighbour) then we need not counter its argument. An isgood is not sent to a neighbour v if there are unprocessed isgoods in the communication channel.

We ensure that individual pairs of agents do not communicate simultaneously by using a mutex on individual communication channels. If an argument from a neighbour v has arrived between the last call to *receive-isgood* and the activation of the lock then we skip sending an argument on this iteration. The result is half-duplex communication with all agents able to operate asynchronously.

To avoid cycles of oscillating agent values in inconsistent problems, we increase the length of successive arguments which are sent. This is achieved by recording the last isgood sent and attempting to increase the length of subsequent isgoods. As any cycle must be finite, eventually the arguments (isgoods) being transmitted will contain the cycle itself. If the cycle is formed from inconsistent values it will generate a nogood and break the cycle; otherwise the cycle-breaking mechanism of *update-view* will take effect.

[1] Cryptographic hash functions can be used to provide a suitably unbiased ordering for sets of assignments written in a canonical form.

From	To	Argument
a	b, i	$\langle(a, 1\text{pm})\rangle$
c	b, d, g	$\langle(c, 1\text{pm})\rangle$
e	d, f	$\langle(e, 1\text{pm})\rangle$
b	a	$\langle(b, 1\text{pm})\rangle$
b	c	$\langle(a, 1\text{pm}), (b, 1\text{pm})\rangle$
d	c, g	$\langle(d, 1\text{pm})\rangle$
d	e	$\langle(c, 1\text{pm}), (d, 1\text{pm})\rangle$
f	e	$\{(e, 1\text{pm})\}$
f	e, g, h	$\langle(f, 3\text{pm})\rangle$
e	f	$\langle(e, 3\text{pm})\rangle$
e	d	$\langle(f, 3\text{pm}), (e, 3\text{pm})\rangle$
c	b	$\langle(c, 2\text{pm})\rangle$
c	d, g	$\langle(b, 1\text{pm}), (c, 2\text{pm})\rangle$
g	c, d, f, h	$\langle(g, 2\text{pm})\rangle$
i	a, h	$\langle(i, 2\text{pm})\rangle$
d	c	$\langle(d, 2\text{pm})\rangle$
d	g	$\langle(c, 2\text{pm}), (d, 2\text{pm})\rangle$
d	e	$\langle(b, 1\text{pm}), (c, 2\text{pm}), (d, 2\text{pm})\rangle$
h	g	$\langle(h, 1\text{pm})\rangle$
h	i	$\langle(g, 2\text{pm}), (h, 1\text{pm})\rangle$
i	h	$\langle(i, 1\text{pm})\rangle$
i	a	$\langle(h, 1\text{pm}), (i, 1\text{pm})\rangle$
a	i	$\langle(a, 3\text{pm})\rangle$
a	b	$\langle(i, 1\text{pm}), (a, 3\text{pm})\rangle$
b	a	$\langle(b, 3\text{pm})\rangle$
b	c	$\langle(a, 3\text{pm}), (b, 3\text{pm})\rangle$
c	d, g	$\langle(b, 3\text{pm}), (c, 2\text{pm})\rangle$

Solved: $a, b, e, f = 3\text{pm}$; $c, d, g = 2\text{pm}$; $h, i = 1\text{pm}$

In the first iteration, a, c and e announce their initial values (all chose 1pm). Note that other agents could communicate at the same time (eg. g and h) but to simplify the explanation we have limited the presentation of arguments.

In the second iteration, b, d and f respond to these proposals. In the first case, b accepts the proposed time of 1pm from a and communicates an agreeable value to a. As c proposed a contradictory value, b provides a longer isgood as a counter-proposal. At the same time, d accepts the proposed time of 1pm from c and communicates that to its neighbours. Finally, f rejects the proposal from e outright with a nogood, and proposes instead a time of 3pm.

In the third iteration, e and c respond to the counter-proposals. Given the longer proposal from f, e must change its current value. When updating its neighbour d, e must provide a longer argument. Similarly, c provides longer proposals to d and g.

In the fourth iteration g and i announce their values for the first time, choosing values supported by c and a respectively.

In the fifth iteration, d accepts the proposed time of 2pm from c. Note the increasing (and varying) lengths of isgoods as d communicates with different neighbours. Similarly, h accepts the proposed time of 2pm from g and chooses its first assignment accordingly.

In the sixth iteration, i changes its *support* from a to h, and changes its value accordingly. This change is then communicated to a. Note that this is the first communication from i to a.

In the seventh iteration, a is forced to choose between b and i as *support*. As a had no previous support it is free to choose either (if it previously had used b as *support* it would have retained it). In this instance, a chooses to use i as *support* and change value accordingly.

The final iterations consist of propagating this change to b (which changes its value), and then to c. Note that c does not change value, but still must communicate the change of b's value to d and g.

Table 1: Example algorithm execution on the problem in Figure 1

4. RESULTS

We have described desirable properties of an algorithm for distributed constraint satisfaction. Specifically, we have attempted to construct an algorithm which:

- has no need for 'authority' between variables, effectively avoiding the need for a total order on variables.
- provides fairness in the level of stability for variables.
- does not add links between variables, avoiding the eventual need for 'broadcasting' assignments.
- addresses the risk of cyclic behaviour exhibited by local search algorithms.

The fact that we do not add links between variables is evident from the algorithm itself. Similarly, we note the absence of any total ordering over the variables, which avoids any notion of 'authority'. Each isgood establishes an order over variables within a local context in the form of a sequence in which assignments were made. This is necessary for the introduction of nogoods in the style of Dynamic Backtracking [2, 1]. However, the combination of these local orders does not necessarily end in the construction of a total order over variables. We have also provided a novel method to address cyclic behaviour which plagues distributed local search algorithms [7]. Each of these results can be observed in the detailed algorithm execution shown in Table 1.

Key to the success of SBDS is the elimination of cyclic behaviour in what is otherwise a local search algorithm. Below we will present formal proof that cyclic behaviour has been eliminated:

LEMMA 1. *Eventually no new nogoods will be generated, and the length of view will become stable (bounded above) for each agent.*

PROOF. Each agent maintains an unbounded store of nogoods, and each isgood is constructed in a way that is consistent with all nogoods known at the time. Therefore, if the algorithm were to never terminate, eventually no new nogoods will be generated.

If no new nogoods are generated, then the method of selection for *support* and *view* ensures that *support* will stabilise. As a result, the length of *view* will also stabilise as it is monotonic increasing over time, but bounded above by either the depth of the 'support tree' or the length of a cycle. □

THEOREM 1. *The algorithm is sound, and will terminate.*

PROOF. The algorithm uses sound nogood derivation techniques and will terminate with 'no solution' only if the empty nogood is derived. Inversely, each agent ensures that its neighbours know its current value, and will continue to communicate if an inconsistency exists. The algorithm will not terminate unless it has a correct answer, and therefore is sound.

By Lemma 1 we know that eventually no new nogoods will be generated and the length of *view* will become stable for each agent. Therefore the *support* for each agent will also become stable, and so value selection for each variable will become dependant only upon information from its *support*. In such a situation the algorithm will only fail to terminate if there exists some directed tour of agents $v_1, \ldots, v_n$ which are 'supporting' each other and oscillating between candidate solutions. However, each candidate solution has the same unordered scope, and so we can utilise the postponement scheme outlined in Section 3. By this scheme, solutions ranked lower by $\prec$ will be removed until the oscillating stops and the algorithm terminates. □

Finally, we provide empirical evidence that no variables change value significantly more often than any other. Figure 2 presents the number of value changes per variable while using our algorithm to solve a randomly constructed feasible problem. Observe that the frequency of value change is fairly evenly distributed amongst variables, with none forced to change significantly more than another. Identical results are seen for other problem sizes and constraint tightness. This particular result also provides further evidence that no significant 'authority' is exerted by one variable over another. If authority was exerted we would expect an exponential (or at least polynomial) increasing curve, as some variables are forced to change by others.

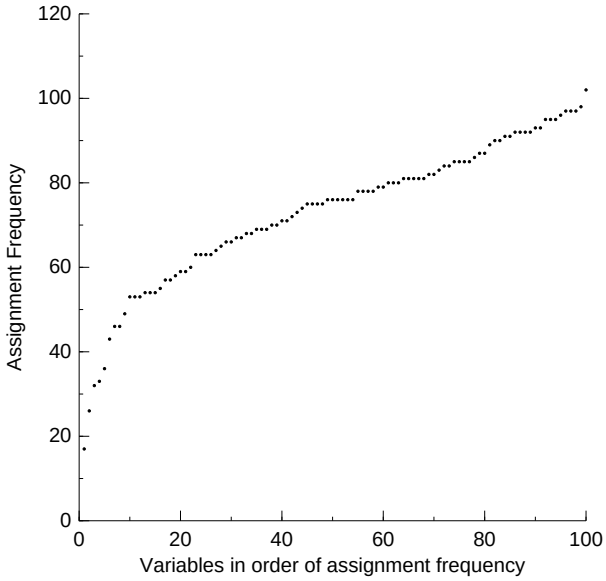


Figure 2: Frequency of assignment changes per variable for a random problem of 100 variables, 300 constraints, domain size of 5, and constraint tightness of 0.325

5. PERFORMANCE

In previous work [4] we used a complicated definition of isgood involving a measure of 'strength' that allowed SBDS to approximate chronological backtracking search. The aim of that measure of strength was to minimise the rate of growth of L in *send-isgood*, and so reduce the amount of information being transmitted in each iteration. However, this measure significantly complicated the proof of completeness. To minimise the rate of growth of L, and to increase performance, we replace this measure with a simple heuristic carefully constructed to maintain completeness.

First note that the completeness of SBDS depends upon lines 1 of *update-view* and line 5 of *send-isgood*. We risk cyclic behaviour if we do not choose a $view'$ which is maximal with respect to $\prec$ or forcibly increase the length sent to neighbours. However, these items incur significant performance penalties. For example, by choosing a $view'$ that is maximal with respect to $\prec$, we are unable to use min-conflict or similar value selection heuristics. Similarly, by consistently increasing the length of isgoods, we discourage local-search behaviour and impact performance accordingly.

However, we are not required to select the maximal $view'$ or increase the length of isgoods with *every* iteration of the algorithm. We may instead perform either action after a finite number of iterations, by:

- forcing an increase in $|sent(v)|$ only after a bounded number of iterations of the algorithm;
- permitting heuristic selection of $view'$ until $|sent(v)|$ cannot be increased further;
- permitting a decrease in $|sent(v)|$ when the ordered scope of *view* changes.

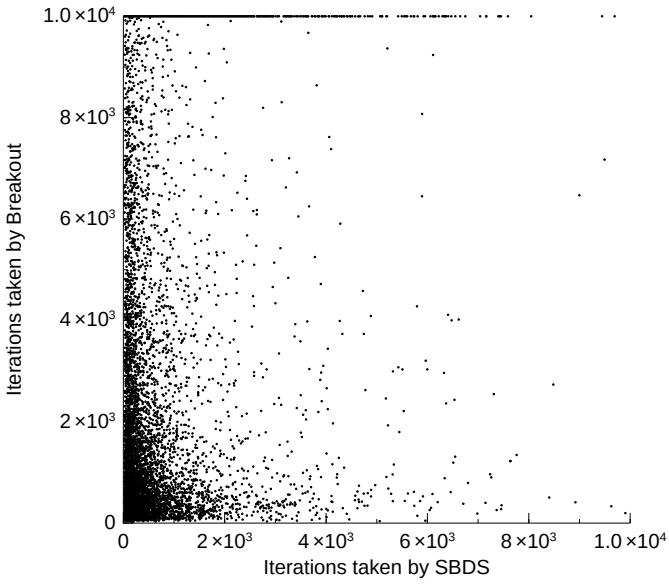


Figure 3: Comparison of SBDS and Distributed Breakout for feasible random problems with 200 variables and 400 constraints, domain size of 5, and constraint tightness of 0.4

These simple techniques significantly improve the performance of SBDS by permitting the use of min-conflict heuristics in value selection. Figures 3 and 4 present comparisons of SBDS and Distributed Breakout. We compare with Distributed Breakout as it is simple to implement, and is often very fast on feasible problem instances. We ran each on approximately 13000 feasible random binary problem instances, with an upper limit of 10000 iterations per instance. We also ran SBDS on approximately 350 infeasible random problem instances generated in the same way.

Figure 3 plots each random problem instance found feasible by SBDS, comparing the number of iterations taken for each algorithm. For the majority of instances where the algorithms completed, both took less than 2000 iterations. However, Distributed Breakout still had a large proportion of instances which took more than 2000 iterations, and a significant proportion were not completed. Particularly important is that there were few problems found 'easy' for Distributed Breakout but hard for SBDS.

Figure 4 graphs the percentage of completed problems for Distributed Breakout and SBDS. As Distributed Breakout is unable to solve infeasible problem instances we have not presented those. As can be seen, of the problem instances found feasible by SBDS, 98% were solved within a 4000-iteration limit. In contrast Distributed Breakout was able to complete only 50% of the same problems within 4000 iterations, and 55% within 10000 iterations. We also present the percentage of problem instances found to be infeasible by SBDS, and the iterations required to find them. While performance is worse on infeasible problems, it is still reasonable.

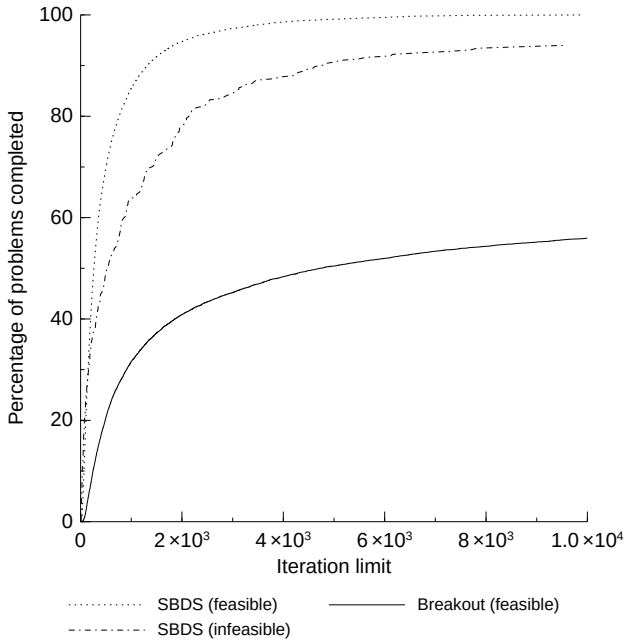


Figure 4: Comparison of SBDS on feasible and infeasible random problems with 200 variables and 400 constraints, domain size of 5, and constraint tightness of 0.4.

6. CONCLUSION

Distributed Constraint Satisfaction Problems provide a natural mechanism for multiagent coordination and agreement. To date, algorithms for Distributed Constraint Satisfaction Problems have tended to mirror existing non-distributed global-search or local-search algorithms. However, there exist natural examples of DisCSPs for which a total variable ordering and/or linking variables is not desired. If we are to solve such DisCSPs we must develop new algorithms designed specifically for distributed environments.

In this paper we have presented one such algorithm. Key to the success of the algorithm is the use of argumentation as a model for agent operation. This technique avoids fixed ranks for agents and the resultant behaviour which is undesirable in natural problems such as meeting scheduling. The placing of a total order over solutions for subsets of variables also provides a novel approach to solving the issue of cyclic behaviour in local search algorithms. This paper also represents a significant simplification, with formal proof results and better performance, of the algorithm presented in [4].

Note that SBDS has been developed with specific goals in mind; it is designed to solve problems for which existing algorithms are unsuitable. It is therefore difficult to compare performance characteristics with other algorithms, in much the same way as comparisons between centralised and distributed algorithms are difficult. The above performance results are promising, but further testing against other distributed search algorithms is required.

7. REFERENCES

[1] C. Bessière, A. Maestre, and P. Meseguer. Distributed dynamic backtracking. In T. Walsh, editor, *CP*, volume 2239 of *Lecture Notes in Computer Science*, page 772. Springer, 2001.

[2] M. L. Ginsberg. Dynamic backtracking. *J. Artif. Intell. Res. (JAIR)*, 1:25–46, 1993.

[3] Y. Hamadi. Interleaved backtracking in distributed constraint networks. *International Journal on Artificial Intelligence Tools*, 11(2):167–188, 2002.

[4] P. Harvey, C. F. Chang, and A. Ghose. Practical application of support-based distributed search. In *ICTAI*. IEEE Computer Society, 2005.

[5] M. Yokoo. Asynchronous weak-commitment search for solving distributed constraint satisfaction problems. In U. Montanari and F. Rossi, editors, *CP*, volume 976 of *Lecture Notes in Computer Science*, pages 88–102. Springer, 1995.

[6] M. Yokoo and K. Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, 2000.

[7] W. Zhang and L. Wittenburg. Distributed breakout revisited. In *AAAI/IAAI*, pages 352–, 2002.

Adaptive Agent Negotiation via Argumentation

Antonis Kakas
Dept. of Computer Science
University of Cyprus
75 Kallipoleos Str.
1678 Nicosia, Cyprus
+35722892706
antonis@cs.ucy.ac.cy

Pavlos Moraitis
Dept. of Mathematics and Computer Science
University René Descartes
45 rue des Saints-Pères
75270 Paris Cedex 6, France
+33144553559
pavlos@math-info.univ-paris5.fr

ABSTRACT

In this paper, we study how argumentation can be used as a basis for negotiation between autonomous agents, where negotiation strategies of the different parties are represented as argumentation theories within their knowledge. We propose an argumentation based negotiation protocol in which offers by the negotiating parties are linked to different arguments that they can build according to their individual negotiation strategy. This protocol is able to take into account the different roles of agents and context of interaction, where the strength of the arguments supporting an offer can depend on these factors. The agents can adapt their negotiation strategies and offers, as their environment changes, in particular during the course of the negotiation as they exchange information. In addition, using abduction alongside with argumentation, agents can find negotiating conditions to support an argument for an offer, thus extending the negotiation object in order to help find an agreement. To illustrate further the advantages of the approach we also study how we can capture and extend negotiation strategies within another negotiation mechanism, in particular the well-known mechanism of bargaining with multiple parties. This exposes further the main advantages of modularity and flexibility of our argumentation based representation of negotiation strategies and the adaptability that this offers in a changing environment of negotiation.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Multi-Agent Systems

General Terms

Theory

Keywords

Negotiation, Argumentation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

1. INTRODUCTION

Negotiation is an important problem in multi-agent systems where autonomous agents need to be able to develop their own high-level strategy of operation. The diversity of strategies that different agents set can raise conflicts between them requiring a negotiation procedure to resolve these. Typically, negotiation aims to modify the local decisions of agents in order to avoid negative interactions and to emphasize the situations where positive or helpful interactions are possible.

The design of computational agents needs automated negotiation for various reasons [18]: 1) Agents interact in an open environment and may pursue different goals; 2) Several new applications such as electronic-business require an operational decision making level; and 3) The development of virtual enterprises requires negotiation and dynamic alliances. In all these cases, automated negotiation should provide a great help to save labor time in the sense that it may replace, at least partially, human negotiators. Automated negotiation has long been studied in the multi-agent systems field and different negotiation mechanisms have been proposed. These are based on game theoretic models (see e.g. [17]), heuristic approaches (see e.g. [6]) and some of them on argumentation (see e.g. [2, 14, 19]).

According to [8, 16] the approaches based on argumentation remove many of the limitations of the other approaches. These limitations stem from the facts that (a) the proposals made by the negotiating agents denote single points in the space of the negotiation agreements, (b) the only feedback that can be made to a proposal is a counter proposal, which itself is another point in the space, or an acceptance or withdrawal and (c) it is hard to change the set of issues under negotiation in the course of a negotiation. The added value of the approaches based on argumentation is that they provide additional information, over and above proposals. The arguments that agents can give to support their offers allow them to explicitly express their needs and rational of proposal. Agents have the possibility to question the arguments used by the other agents and to state their specific preferences for supporting their proposals. This gives the possibility to alter, during the negotiation process, the basic position of a negotiating party and its offers, e.g. by extending the negotiation space, or by offering treats or rewards in order to achieve an agreement.

In this paper, we study negotiation using as a basis an argumentation framework [9, 10] that is able to express in a simple and direct way policies and negotiation strategies that can vary according to the particular circumstances in which these are applied. Based on this we propose a negotiation protocol where the decision making for negotiation is done through argumentative self-deliberation of the agents with their own theory. This is able to take into account the different roles that agents can have and specific contexts that describe particular or exceptional situations that agents could find themselves during the course of negotiation. Roles and context define in a natural way dynamic preferences on the decisions implied by the negotiation strategies of the agents as their environment (that includes the other negotiating agents) changes. In addition, aiming to provide agents with a level of robustness in the face of incomplete information from the negotiation environment, we integrate abduction within the argumentation framework. Agents are then able to deliberate on alternative choices and take decisions which are conditional on assumptions about the environment of the negotiation, e.g. assumptions on the preferences of other agents, the specific conditions in the current negotiation context, etc.

In order to illustrate further how argumentation can be used to enhance negotiation we will also study how we can capture and extend negotiation strategies within the traditional negotiation mechanism of bargaining with multiple parties. We show that their formulation within argumentation provides a representation that is more flexible in accommodating special or new requirements and allows us to enhance these negotiation strategies.

The rest of the paper is organized as follows. Section 2 presents briefly the background argumentation framework that we will use. Section 3 describes a new negotiation protocol based on argumentation. Section 4 develops a buying-selling scenario illustrating the main features of our protocol. It also studies representations of negotiation strategies within the negotiation mechanism of bargaining with multiple parties. Section 5 discusses related work and concludes.

2. BACKGROUND

2.1 Argumentation with Dynamic Preferences

In this section, we present the basic concepts of the argumentation framework on which our study of negotiation model will be based. This framework is presented in detail in [9, 10]. Each party in the negotiation will represent its own negotiation strategy (and background knowledge) within this framework.

Our aim is to use the declarative nature of the framework to allow the representation of flexible policies of negotiation strategies that can be sensitive to different circumstances as the negotiation process evolves. In particular, we will assume that negotiating agents are always associated with an (social) environment of interaction. We call this the *context of interaction* (i.e. market, enterprise, etc.). This determines the relationship between the possible roles the different agents can have within this environment and can influence the negotiation strategies used by the agents. Generally, roles are associated to a *default context* that defines shared social relations of different forms (e.g. authority, friendship, relationship, etc.) and specifies the interaction between different roles. For instance in the army context an officer gives orders that are obeyed by a soldier, or in the market a regular customer has benefits over a normal customer.

Consequently, a default context can be understood via priorities between the different roles. However, a default context determining the basic roles filled by the agents is not the only environment where they could interact and negotiate. For example two friends can also be colleagues, an officer and a soldier can be family friends in civil life. Therefore we consider a second level of context, called *specific context,* which can overturn the pre-imposed, by the default context, ordering between roles and establish a different relation between them. For instance, the authority relationship between an officer and a soldier would change under the specific context of a social meeting or regular and normal customers may have an equivalent consideration during the high season.

The argumentation framework that we will use was originally proposed in [11] and extended further in [9, 10] in order to deal with the dynamic nature of interaction of autonomous agents. It has been also adapted to model communication theories and dialogues (see e.g. [4, 12]). This framework uses a priority relation between arguments in the theory which is not a static relation but context dependent that captures the non-static preferences associated to roles and specific context of interaction. We define an agent's argumentation theory for describing a preference policy, e.g. a negotiation strategy, in an environment with roles and context as follows. The full presentation of the argumentation framework can be found in (e.g. [10]).

Definition 1. An agent's argumentative **policy theory, T**, is a theory $\mathbf{T}= ((\mathcal{T}, \mathcal{T}_\circ), \mathcal{P}_R, \mathcal{P}_C)$ where $\mathcal{T}$ contains the **argument** rules in the form of definite Horn logic rules, $\mathcal{P}_R$ contains **priority** rules which are also definite Horn rules with head h-p(r_1, r_2), s.t. $r_1, r_2 \in \mathcal{T}$ (h-p standing for higher priority) and all rules in $\mathcal{P}_C$ are also priority rules with head h-p(R_1, R_2) s.t. $R_1, R_2 \in \mathcal{P}_R \cup \mathcal{P}_C$. $\mathcal{T}_\circ$ contains auxiliary rules of the agent's background knowledge

We therefore have three levels in an agent's theory. In the first level we have the rules $\mathcal{T}$ that refer directly to the subject domain of the agent. We call these the **Object-level Decision Rules** of the agent. In the other two levels we have rules that relate to the policy under which the agent uses his object-level decision rules according to roles and context. We call the rules in $\mathcal{P}_R$ and $\mathcal{P}_C$, **Role (or Default Context) Priorities and Specific Context Priorities** respectively.

In a more general setting, we can associate the dynamic preferences expressed through the rules of the two later levels, to *normal situations* (related to a default context) and *specific situations* (related to specific contexts).

Definition 2. Let $(\mathcal{T}, \mathcal{P})$ be a theory, $T \subseteq \mathcal{T}$ and $P \subseteq \mathcal{P}$. Then (T, P) is **admissible** iff $(T \cup P)$ is consistent and for any (T', P') if (T', P') attacks (T, P) then (T, P) attacks (T', P').

We remind that the notion of attack between arguments in a theory is based on the possible conflicts between a literal L and its negation and on the priority relation given by *h-p* in the theory.

Given a ground literal L then L is a **credulous** (respectively **skeptical**) **consequence** of the theory iff L holds in a (respectively every) maximal, w.r.t. set inclusion, admissible subset of the theory.

Hence for an object-level argument (from $\mathcal{T}$) to be admissible it needs to take along with its priority arguments (from $\mathcal{P}$ where $\mathcal{P} = \mathcal{P}_R \cup \mathcal{P}_C$) to make itself at least "as strong" as the opposing counter-arguments. This need for priority rules can repeat itself when the initially chosen ones can themselves be attacked by opposing priority rules and again we would need to make now the priority rules themselves at least as strong as their opposing ones.

In several cases the admissibility of an argument depends on whether we have or not some background information about the specific case in which we are reasoning. Sometimes this information maybe just unknown and agents can reason further to find *assumptions* related to the unknown information under which they can build an admissible argument. We can formalize this conditional form of argumentative reasoning by defining the notion of *supporting information* and extending argumentation with *abduction* on this missing information. This, as we will see later, is very useful for negotiation.

Definition 3. Let T= $((\mathcal{T}_\circ, \mathcal{T}), \mathcal{P})$ be a theory, and, $\mathcal{A}$, a distinguished set of predicates in the language of the theory, called **abducible** predicates[1]. Given a goal G, a set S of ground abducible literals consistent with the *non-defeasible* part of the background theory $\mathcal{T}_\circ$, is called a **strong (**respectively **weak) supporting information** for G iff G is a **skeptical** (respectively **credulous**) consequence of $((\mathcal{T}_\circ \cup S, \mathcal{T}), \mathcal{P})$ and S is a minimal (w.r.t. set inclusion) such set that has this property.

An argument can then be generalized as follows.

Definition 4. Let T = $((\mathcal{T}_\circ, \mathcal{T}), \mathcal{P})$ be a theory, and $\mathcal{A}$ its *abducible* predicates. A **supported argument** in T is a tuple (Δ, S), where S is a set of abducible literals consistent with $\mathcal{T}_\circ$ and Δ is a set of argument rules, which is not admissible in $((\mathcal{T}_\circ, \mathcal{T}), \mathcal{P})$, but is admissible in the theory $((\mathcal{T}_\circ \cup S, \mathcal{T}), \mathcal{P})$. We say that S **supports** the argument Δ and that Δ is **conditional** on S.

The supporting information refers to the incomplete and evolving information of the external environment of interaction. Typically, this information pertains to the context of the environment, the roles between agents or any other dynamic aspect of the agent's knowledge.

Given the above framework the **argumentative deliberation** of an agent can be formalized via the following basic reasoning functions that will be useful in our study of negotiation.

Definition 5. Let *Ag* be an agent, T his argumentation theory, G a goal and S a set of abducibles consistent with the background part, $\mathcal{T}_\circ$, of its theory. Then we say that *Ag* **deliberates** on G, under S, to produce S^{ag}, denoted by **deliberate**(*Ag*, G, S; S^{ag}), iff S^{ag} is a strong supporting information for G in the theory T and $s^{ag} = S^{ag} - S$. If S^{ag} is a subset of S then we say that *Ag* accepts G under T and S and is denoted by **accept(*Ag*, G, S)**.

[1] For simplicity, we will assume that abducible predicates are a subset of the background predicates.

3. THE NEGOTIATON PROTOCOL

In this section, we will set up a negotiation protocol for intelligent agents by exploiting the internal argumentative deliberation that each agent can perform. The negotiation object(s), e.g. the price of a product, decided by the involved agents are associated to a class of goals for each agent, e.g. sell at high prices for a seller and buy at low prices for a buyer. During the negotiation the agents accumulate extra information about the external environment via supporting information conditions that each agent generates to support, in its own theory, the argument for its offer (see definition 3). This supporting information is then associated with the offer and in some cases it can be seen as extending the negotiation object. The various items of supporting information are collected together and build gradually during the negotiation allowing thus a form of incremental deliberation of the agents as they acquire more information. We note that in this paper, we are not interested in the communication protocol supporting the negotiation. The main elements of the negotiation protocol we propose are as follows:

1. Each agent insists in making proposals corresponding to its own high-level goal as long as its deliberation within its theory and the accumulated supporting information (agreed by the two agents so far) produces new supporting information (see definitions 3, 4) for this goal. A proposal contains an offer corresponding to the negotiation object (e.g. a specific price), together with supporting information representing conditions under which this offer is made. Different offers will typically have different supporting information, e.g. a goal to buy at a low price may contain several prices as possible offers, each of them being supported by the appropriate information. Thus the negotiation object can be extended and may cover several issues related to the initial issue, e.g. terms and conditions under which an agent could accept a specific price.

2. The first of the two negotiating agents who is unable to produce a new offer with supporting information for his goal, abandons it and searches for supporting information, if any, under which it can accept the (counter) proposal of the other agent. For example, a seller agent unable to find another way to support offers with high prices considers selling at a low price and looks for supporting information under which it may be able to do so.

3. In such a case, the negotiation enters a conciliation phase and if the receiver agent can endorse the proposed supporting information the negotiation ends with agreement on this offer and the supporting information accumulated so far. Otherwise, the sender takes this into account and tries again to find another way to support the goal (offer) of the other agent. If this is not possible, then the negotiation ends in failure.

Besides the argumentative functions *deliberate* and *accept* (see definition 5) we use three other functions of an agent, external to his argumentative reasoning, that are needed in the negotiation process. The function *propose(Offer, e_j, s_i)* is used by a sender agent to determine what information to send to the other agent: *Offer* is an offer drawn from an agent's goal and proposed

as an alternative on which negotiating agents could agree; e_j is the evaluation (see below) by the sender of the supporting information s_j sent to it in the previous step by the other agent; and s_i is the new supporting information produced by the deliberation of the sender for its *Offer*.

The function *evaluate(Ag, s_i)* produces an evaluation e_i, where for each (abducible) literal in the supporting information s_i this is either accepted or rejected (i.e. negated) according to some process of evaluation by a receiving agent *Ag.* This can therefore be considered as a *critique* (see e.g. [16]) of s_i by *Ag*. This function can vary in complexity from a simple check in the agent's database to an argumentative deliberation on s_i and a new subsidiary negotiation process on this.

The function *update(S,e)* updates, through an external mechanism, the store of accumulated supporting information *S* with the new information *e* produced by the agent's evaluation of the supporting information sent by the other agent. Part of the task of this function is also to maintain the consistency of the store *S*. We require that conflicts are resolved in favour of the latest information i.e. in favour of the information in *e*. In this paper we will not be concerned with the details of this mechanism as they are not important for the specification of the protocol. Thus *S* is related to the concept of *commitment store* (see [7, 13]) adopted by the agent literature (see e.g. [1]). A commitment store holds all the statements to which an agent has committed during the negotiation. In our case *S* corresponds to the union of the commitment stores and contains all the additional information produced in the course of the negotiation and accepted by each negotiating agent.

The negotiation protocol we are proposing operates via the alternate application by each one of the two negotiating agents of the algorithm given below. This defines formally the steps of the negotiation process. Note that in the presentation of this algorithm, we assume that it is agent X who is applying it. Agent Y applies exactly the same algorithm, and therefore the reader has only to replace X with Y in order to see how a negotiation process between two agents would evolve (see Section 4 for a buying-selling scenario).

Below we will denote by G^X the argumentation goal corresponding to an offer O^X and by $e_n^{X\to Y}$ the evaluation by agent X of the supporting information sent by agent Y. Also we will denote by $s^{I,J}$ the supporting information generated by agent, I, for the goal of agent, J, (I, J can take the values X or Y).

Each agent (agent X here) can receive three types of proposals. In the first one, denoted by *propose(O^Y, $e^{Y\to X}$, $s_n^{Y,Y}$)*, another agent, Y, sends to X an offer, O^Y, with a new alternative supporting information, $s_n^{Y,Y}$. This proposal also includes, $e^{Y\to X}$ (empty in the initial proposal), which is the evaluation by Y of the supporting information sent to it in the previous step by X for its offer O^X. In the second type of proposal, denoted by *propose(O^Y, $e_m^{Y\to X}$, ⊗)*, the other agent, Y, sends to X (via step 4.4 for agent Y) its answer to an offer made previously by X to consider the offer O^Y of Y, i.e. X has entered the conciliation phase and Y is responding to this. This answer contains the evaluation, $e_n^{Y\to X}$, by Y of the terms (or conditions) proposed by X in order to accept an offer O^Y satisfying the goal of Y. Finally, in the third type of proposal, denoted by *propose(O^X, ⊗, $s_n^{Y,X}$)* another agent, Y, sends to X an answer where it proposes the conditions, $s_n^{Y,X}$, under which it could accept the offer O^X of X, i.e. Y has previously entered the conciliation phase.

Begin
S={}; n=0
1. agent X receives a proposal O from an agent Y

2. **if** O is of the form propose(O^Y, $e_{n-1}^{Y\to X}$, $s_n^{Y,Y}$) **then**
2.1. $e_n^{X\to Y}$←evaluate(X, $s_n^{Y,Y}$);
2.2. S←update(S, $e_{n-1}^{Y\to X} \cup e_n^{X\to Y}$)
2.3. **if** $e_n^{X\to Y} = s_n^{Y,Y}$ and accept(X, G^Y, S) **then** END(agreement, O^Y)
else
2.4. n←n+1; find $s_n^{X,X}$ s.t. deliberate(X, G^X, S; $s_n^{X,X}$) and $s_n^{X,X} \neq s_i^{X,X}$ ∀ i < n
2.5 **if** s_n^X exists **then** propose(O^X, $e_{n-1}^{X\to Y}$, $s_n^{X,X}$) to Y
else (*Enter Conciliation Phase*) m = 0
2.6. S←update(S, $e_n^{Y\to X}$)
2.7. m←m+1; find $s_m^{X,Y}$ s.t. deliberate(X, G^Y, S; $s_m^{X,Y}$) and $s_m^{X,Y} \neq s_j^{X,Y}$ ∀ j < m
2.8. **if** $s_m^{X,Y}$ exists **then** propose(O^Y, ⊗, $s_m^{X,Y}$) to Y
else END(Failure)
2.9. endif
2.10. endif
2.11. endif
2.12. endif

3. **if** O is of the form propose(O^Y, $e_m^{Y\to X}$,⊗) **then**
3.1. goto 2.6; endif

4. **if** O is of the form propose(O^X, ⊗, $s_k^{Y,X}$) **then**
4.1. $e_k^{X\to Y}$←evaluate(X, $s_k^{Y,X}$)
4.2. S←update(S, $e_k^{X\to Y}$)
4.3. **if** $e_k^{X\to Y} = s_k^{Y,X}$ and accept(X,G^X,S) **then** END(agreement, O^X)
4.4. **else** propose(O^X, $e_k^{X\to Y}$, ⊗)
4.5. endif
4.6. endif
end

We can show the following formal properties for the above negotiation protocol.

Proposition 1. Let X, Y be two agents and T_X, T_Y their respective argumentation policy theories which contain a finite number of sentences. Then,
(1) any negotiation process will terminate.
(2) if a negotiation process terminates with agreement on an offer O and S is the final set of supporting information accumulated during the negotiation then the corresponding goal, G(O), is a skeptical conclusion of both $T_X \cup S$ and $T_Y \cup S$.

Proof
(1) Termination follows directly from the fact that, when an agent deliberates again, either on its own goal or the goal of the other agent, the algorithm requires that it produces new supporting information and the fact, that there can only be a finite number of different supporting information generated by the theories of the agents.
(2) There are two cases for the process to terminate in agreement. In the first case (step 2.3) agent X agrees on the offer G^Y of the other agent (agent Y) when accept(X, G^Y, S) holds. By the definition of accept(X, G^Y, S) this means that G^Y is a skeptical conclusion of $T_X \cup S$. This is also a skeptical conclusion of $T_Y \cup S$ since agent Y has produced this offer for its own goal G^Y (at its step 2.4) and its supporting information has been accepted by agent X and updated in S at step 2.2. Similarly, we show that the result holds also in the second case where agreement can be reached, at step 4.3.

4. EXAMPLES OF NEGOTIATON

In this section, we will present two examples of negotiation in order to illustrate how our argumentation-based approach to negotiation works in practice. First we will present a bargaining example of buying-selling transactions that illustrates how the specific negotiation protocol that we have proposed in the previous section applies and how the argumentative deliberation of the agents is performed within this protocol. Then we will investigate how our argumentation framework can be used for representing and extending negotiation strategies within another negotiation mechanism, in particular the well-known mechanism of bargaining with multiple parties.

4.1 A Buying-Selling Scenario

Let us consider a buying-selling scenario between a seller agent, called X, who has the goal, G^X, to sell a product at a high price and a buyer agent, called Y, who has the (opposing) goal, G^Y, to buy this product at a low price. These goals generate corresponding offers and counter-offers containing different values for the negotiation object, namely the price for the product. Therefore agents are trying to find an agreement on a specific price by agreeing either on an offered selling price O^X or on an offered buying price O^Y. We assume that the seller has the following simplified (where only two prices high and low exist) argumentation policy for selling products. We present only a part of this theory[2].

[2] Here and below we will use Logic Programming notation where any term starting with a capital letter represents a variable. Abusing this notation we will denote the constant names of the priority rules in the role part, $\mathcal{P}_R$, of the theory with capital R letters (subscripted) and the priority rules in the specific context part, $\mathcal{P}_C$, of the theory with subscripted capital C letters. All variables are universally quantified over the whole rule in which they appear. Each rule represents the collection of ground rules that we obtain by grounding its variables over the Herbrand universe of the theory.

The goal G^X of the seller is G^X=sell(prd,buyer,high_price). There are two ways or methods to get this "high_price", either through a normal payment method or through payment via installements. The object-level theory $\mathcal{T}^X$ contains the rules:
r_1: sell(Prd, Ag, high_price, method1)← pay_normal(Ag,Prd)
r_2: sell(Prd, Ag, high_price, method2)← pay_install(Ag,Prd)
r_3: sell(Prd, Ag, low_price)← pay_cash(Ag,Prd)
r_4: sell(Prd, Ag, Price)← sell(Prd, Ag, Price, Method)
r_5: ¬sell(Prd, Ag, P_2)← sell(Prd, Ag, P_1), $P_2 \neq P_1$
r_6: ¬sell(Prd, Ag, P, M_2)← sell(Prd, Ag, P, M_1), $M_2 \neq M_1$

Its role and context priority theories, $\mathcal{P}^X_R$ and $\mathcal{P}^X_C$, are given below. They contain the policy of the seller under which it should treat or negotiation with the various types of customers. For example, it should prefer to sell with normal paying conditions (e.g. cash or visa) over payment by installments when the buyer is a normal customer (see R_1). Also that there is always a preference to sell at high price (see R_2, R_3) but for regular customers there are conditions under which the seller would sell at low price (see R_4, R_5). This low price offer to a regular customer applies during a sales season (see C_3, C_4) and not during a high season (see C_1, C_2) where the preference of a high price is stronger.

R_1: h-p(r_1(Prd, A), r_2(Prd, A)) ← normal(A)
R_2: h-p(r_1(Prd, A), r_3(Prd, A))
R_3: h-p(r_2(Prd, A), r_3(Prd, A))
R_4: h-p(r_3(Prd, A), r_1(Prd, A)) ←regular(A), buy_2(A, Prd)
R_5: h-p(r_3(Prd, A),r_1(Prd,A))←regular(A), late_del(A, Prd)

C_1: h-p(R_2(Prd, A), R_4(Prd, A)) ← high_season
C_2: h-p(R_2(Prd, A), R_5(Prd, A)) ← high_season
C_3: h-p(R_4(Prd, A), R_2(Prd, A)) ← sales_season
C_4: h-p(R_5(Prd, A), R_2(Prd, A)) ← sales_season
C_5: h-p(R_4(Prd, A), R_5(Prd, A))

Lets us consider the particular negotiation scenario given below as it evolves through the application of our proposed negotiation protocol and study how the seller uses its own argumentative reasoning, with the above theory, in this scenario. For simplicity of presentation we will show the *update* function simply as union. We would also like to note that in the following example we only present the reasoning made by each agent during the negotiation process. The presentation of the communication protocol (e.g. structure of exchanged locutions, performatives used, etc.) implementing this negotiation dialogue is out of the scope of this paper. This is part of our future work and it will be based on the work we proposed in [4, 12].

Seller X (step 1)
-propose(high_price, {}, $s_0^{X,X}$)
-$s_0^{X,X}$={ pay_normal, normal(Y)})

Buyer Y (step 2)
-$e_0^{Y\to X}$={ pay_normal, ¬normal(Y)}
-S= $e_0^{Y\to X}$
-deliberate(Y, buy(prd, X, low_price),S; $s_1^{Y,Y}$)
-$s_1^{Y,Y}$={expensive(Y, high_price)}
-propose(low_price,$e_1^{X\to Y}$, $s_1^{Y,Y}$)

Seller X (step 3)
-$e_1^{X\to Y}=s_1^{Y,Y}$
-$S\leftarrow S\cup e_1^{X\to Y}$
-Cannot Accept
-deliberate(X, sell(prd, Y, high_price), S; $s_1^{X,X}$)
-$s_1^{X,X}$={pay_install}
-propose(high_price, $e_1^{X\to Y}$, $s_1^{X,X}$)

Buyer Y (step 4)
-$e_2^{Y\to X}=\neg s_1^{X,X}$
-$S\leftarrow S\cup e_2^{Y\to X}$
-deliberate(Y, buy(prd, X, low_price), S; $s_2^{Y,Y}$)
-$s_2^{Y,Y}$={pay_cash}
-propose(low_price, $e_2^{Y\to X}$, $s_2^{Y,Y}$)

Seller X (step 5)
-$e_3^{X\to Y}=s_2^{Y,Y}$
-$S\leftarrow S\cup e_3^{X\to Y}$
-Cannot Accept
-deliberate(X, sell(prd, Y, high_price), S; s)
-fails to find new s.

Seller X (step 6): ***X Enters Conciliation Phase***
-deliberate(X, sell(prd,Y,low_price), S; $s_1^{X,Y}$)
-$s_1^{X,Y}$={regular(Y), buy_2(Y,prd)}
-propose(low_price, ⊗, $s_1^{X,Y}$)

Buyer Y (step 7)
-$e_4^{Y\to X}$={regular(Y), ¬buy_2(Y, prd)}
-propose(low_price, $e_4^{Y\to X}$, ⊗)

Seller X (step 8)
-$S\leftarrow S\cup e_4^{Y\to X}$
-deliberate(X, sell(prd, Y, low_price), S; $s_2^{X,Y}$)
-$s_2^{X,Y}$={later_del(Y,prd)}
-propose(low_price, ⊗, $s_2^{X,Y}$)

Buyer Y (step 9)
-$e_5^{Y\to X}= s_2^{X,Y}$
-$S\leftarrow S\cup e_5^{Y\to X}$
-accept offer; END(agreement, low_price)

At step 1 the seller agent proposes to the buyer agent to buy the product at a high price (which is the preferred price for the seller according to his theory). Step 2 of the example corresponds to the second step of the negotiation algorithm. At this stage the buyer agent cannot accept the offer. He thus deliberates on his own goal and proposes a low price. At step 3 the seller cannot accept the proposal for low price: his theory even under the fact that this is not a normal customer contains admissible arguments for the contrary of this proposal to sell low. He now needs to find a new argument for his goal of selling high. Deliberating on his goal, he finds such an argument, using the object-level rule r_2. This new argument needs the support (or is conditional on) *pay_install(buyer, prd)* and the seller offers this information to the buyer insisting on his offer of a high_price.

At step 5 the seller deliberates again on his own goal (to sell high) but cannot find a new solution anymore. It therefore initiates at step 6 a conciliation phase of the negotiation where he considers the offer made by the buyer, i.e. to sell at *low_price*. He deliberates on this goal and finds (an admissible argument using r_3 and R_4) that it is possible to do so if the customer can pay cash (which is already in the store S), is a *regular* one and he accepts to buy two of the products. In this way the seller agent extends the negotiation object by submitting for negotiation these latter two conditions. Note that for this to skeptically derive the offer to sell low, we need to be at a sales season and that the context C_1 rule must not apply i.e. that we are not in high season, facts that the seller knows in its background theory. The buyer confirms that he is a regular one but refuses to buy two (i.e. it generates a critique to the extended offer). The seller then has another solution to sell low to a regular customer conditional on late delivery, which is accepted by the buyer agent and the process ends in agreement on "low price with a late delivery".

Note that the late delivery condition has been generated dynamically in the particular scenario extending the negotiation object of the price in order to reach agreement. In some other scenario it could happen that this will not enter at all the process.

As we have already remarked, the evaluation function *evaluate(Ag, s_i),* can take, in its more complex case the form of a new subsidiary argumentative deliberation of *Ag* on s_i. This could lead to a nested negotiation process, where the negotiation object is extended further and covers additional issues related to the original (or parent) object. Let us illustrate this subsidiary argumentative deliberation, along with the possible nested negotiation steps, by considering that the buyer agent in the above example has the following argumentation theory for his general policy with regards to methods of payment and how he uses this to evaluate proposals from the seller.

r_1: pay(Prd, Price, install)← reasonable(Prd, Price)
r_2: pay(Prd, Price, cash)← ¬expensive(Prd, Price)
r_3: ¬pay(Prd, P, M_2)←pay(Prd, P, M_1), $M_2 \neq M_1$
R_1:h-p(r_1(Prd, P), r_2(Prd, P))
R_2:h-p(r_2(Prd, P_1), r_1(Prd, P_2)) ←¬cash_flow_problems, $P_1 \leq P_2$
cash_bonus(Prd, Bonus)
C_1:h-p(R_2(Prd,P_1,P_2),R_1(Prd,P_2))←discount(Prd,P_1,D), D>30%

According to this policy buyer prefers to pay by installments rather than cash. When he has no cash flow problems and there is a cash bonus he prefers to pay in cash (provided also that the cash price is not higher than the installments price). In the specific case where the price is at a discount of more than 30% then this preference to pay cash is stronger than the preference to pay by installments.

At the fourth step of the example above the buyer evaluates the proposal of the seller to pay by installments via an argumentative deliberation on *pay(prd, high_price,install).* The negotiation object is extended from <Price> to <Price, MethodOfPayment>. Despite the fact that *high_price* is a reasonable price for the buyer *pay(prd,high_price,install)* it is not a skeptical conclusion of his theory as he has an admissible argument for *pay(prd, low_price, install).* The buyer would then make this counter offer (in a new subsidiary step) to pay by installments but at a low price. If the seller rejects this offer the buyer could then re-evaluate and, assuming that there is a cash

bonus at a discount of over 30% for *low_price*, the above theory of the buyer would then give the skeptical conclusion *pay(prd, low_price, cash)*. Hence the buyer would then offer to pay cash for a low price.

4.2 Bargaining with Multiple Parties

In a bargaining setting, the negotiation process starts with an initial offer formulated by one of the participating parties and then proceeds with an alternate exchange of offers. This process stops either when an agreement is found or one of the involved parties decides to terminate the negotiation with a specific party (or all the parties) without any agreement. Normally, in many realistic scenarios a member participates in several bargaining processes simultaneously, e.g. a buyer contacts several merchants in search for the best terms for buying a given product.

We will study an example type of bargaining considered in [5]. According to this scenario an agent is responsible for negotiating a price with several potential sellers on behalf of a buyer. The offers sent by the sellers are arranged in a queue and then are taken from the queue and evaluated. A buyer agent can accept an offer, propose a counter-offer or terminate the negotiation with the corresponding party. If the offer is accepted, the negotiating agent asks the other party to confirm the agreement, and upon receiving the confirmation, he terminates all the negotiation processes in which he is involved. In the contrary case where a confirmation refusal message arrives or the confirmation message is not received after an agreed delay, an occurrence of the event "no_confirmation" is generated and the agent continues to negotiate with the other remaining parties. In the case where there are no more potential sellers remaining, the overall negotiation process is terminated with a failure.

Below we will show how this evaluation strategy of the buyer agent for each offer it receives can be captured in a natural and direct way within our argumentation framework[3]. We will use the term "offer" to designate the prices proposed by the sellers, and the term "counter-offer" to designate the prices proposed by the buyer. We will also use the following predicates and constants in the representation of the buyer's theory:

- *offer(P,X):* the current offer of the party P is X
- *first_offer(P,X)*: indicates the first offer X of the party P
- *previous_offer(P,X)*: the previous offer of the party P was X
- *counter_offer(P,X)*: X is a counter offer suggested as a reply to P's most recent offer reception
- *previous_counter_offer(P,X)*: X is a counter offer suggested as a reply to the P's second most recent offer reception
- *accept(P, X)*: indicates that the most recent offer X of P can be accepted
- *terminate(P)*: indicates that the negotiation with the party P must be terminated
- *good_price*: indicates a price that can be accepted
- *increment*: indicates an acceptable amount of raising the price
- *acceptable_price*: it is the top price that can be accepted
- *best_offer(X)*: X is the current best offer
- *time_remaining(T)*: indicates the time before the deadline of the bargaining

In the theory that we will present here we will also extend the standard bargaining scenario in order to illustrate further the flexibility of our direct and modular representation of the problem. For example, we can allow for a price greater to "good_price" to be acceptable under some specific circumstances, e.g. when we have only a few remaining sellers or little time remaining before the deadline of the bargaining, provided that the buyer has a big urgency for buying a specific product.

We will represent one by one the policy rules of the evaluation strategy of the buyer directly from their natural specification. Note that "counter_offer", "accept" and "terminate" are all contrary to each other, i.e. each one implies the negation of all others. Similarly, different prices for "accept" and "counter_offer" are contrary to each other.

"If the first offer of P is greater than the "good_price" then ask for the "good_price"."
r_1:counter_offer(P,good_price)←first_offer(P,X), X>good_price.

"If P has not improved his offer terminate the negotiation with him."
r_2: terminate(P) ← offer(P,Y), previous_offer(P,Z), Y≥ Z.

"If P has improved his offer with respect to the previous one, then concede a small amount with respect to the previous counter-offer but do not counter-propose a price below the current best offer."
r_3: counter_offer(P, min(X+increment, W)) ← offer(P,Y), previous_offer(P,Z), Y<Z, previous_counter_offer(P,X), best_offer(W)

"If the offer of P is below the acceptable price and it is the best one that has been received then you can accept it"
r_4: accept(P,Y) ←best_offer(Z), offer(P,Y), Y≤Z, Y≤ acceptable_price

"If somebody offers the "good_price" then this offer can be accepted."
r_5: accept(P,Y) ← offer(P,Y), Y≤good_price

"A good price has preference over any other response and over other prices below the acceptable price"

R_1: h-p(r_5(P,Y), r_2(P))
R_2: h-p(r_5(P,Y), r_3(P,Z))
R_3: h-p(r_5(P,Y), r_4(P,Y1)) ← Y < Y1

"Prefer to accept a price below the acceptable price over terminating the process"
R_4: h-p(r_4(P,Y), r_2(P))

"Prefer to counter offer asking for a better price over accepting a price below the acceptable price"
R_5: h-p(r_3(P,H), r_4(P,Y))

"When we are in a critical situation, i.e. there are only a few sellers left and time is running out, then prefer to accept an offer over making a counter offer. This applies only when there is a big urgency to acquire the product."

[3] Note that we will not be concerned here with the issue of how the buyer selects the next offer to be evaluated. This can be done using another modularly separated preference policy written analogously in the theory of the buyer.

R_6: h-p(r_4(P,Y), r_3(P,H)) ← few_sellers_remaining, time_remaining(T), T≤time_threshold
C_1: h-p(R_5(P,H), R_6(P,Y))
C_2: h-p(R_6(P,Y), R_5(P,H)) ← big_urgency

5. RELATED WORK AND CONCLUSIONS

We have presented an argumentation-based negotiation framework, which like other approaches that use argumentation (see e.g. [8, 16]), removes many of the limitations of the alternative approaches. Our approach confirms the advantages of an argumentation-based approach to negotiation and covers the requirements defined in [8] for an effective argumentation-based negotiation. Comparing with other argumentation-based approaches (see e.g. [2, 14, 19]), the integration of roles (or default context) and contexts (or specific contexts) within the deliberation process of the agents facilitates agents to dynamically adapt their strategies, offers and counter-offers as the environment of negotiation changes. In addition, integrating abduction within the argumentative reasoning of the agents we allow them to extend, during the process of negotiation, the negotiation object with other objects pertaining to conditions under which an agent could accept an offer. In most current approaches (see e.g. [14]), agents generate proposals and counter-proposals until an agreement or a failure is reached, by reasoning only on their own goals or objectives. In our framework we have proposed a negotiation protocol where an agent, when he cannot satisfy his own goal, can consider in a conciliation phase, the other agent's goal and search for conditions under which he could accept it. This can lead again to the extension of the negotiation object helping to reach an agreement. We have shown that our argumentation framework is well suited for modeling this negotiation protocol. However, the negotiation protocol itself, is not crucially dependent on the specific underlying logical framework for argumentation and it could be built similarly on top of other such frameworks (e.g. [3, 15]) provided also that they are extended with abduction so that they can have supported arguments. We also have investigated through the extension of a case study, originally presented in [5], the application of our argumentation framework for representing negotiation strategies within another negotiation mechanism, in particular the well-known mechanism of bargaining with multiple parties. Other negotiation problems, e.g. the resource allocation problem, can also be captured and extended in the same way. This framework and the presented examples have been implemented using the Gorgias [20] system.

Our future work concerns the introduction of multi-criteria techniques from decision theory in the evaluation, by the agents, of the exchanged offers and counter-offers and the development of an associated communication protocol to support the negotiation protocol, based on the work we proposed in [4, 12]. We wish also to investigate how to integrate within the same framework a "personality" component for each one of the negotiating agents, expressed also as an argumentation theory, and study how this can influence the negotiation process between different types (e.g. cooperative or self-interested) of agents.

6. REFERENCES

[1] Amgoud L. and Parsons S. Agent dialogues with conflicting preferences, in *Proc. ATAL'01*, pp. 190-205, 2001.

[2] Amgoud L., Parsons S., and Maudet N. Arguments, dialogue and negotiation, in *Proc. ECAI'00*, pp. 338-342, 2000.

[3] Brewka, G. and Eiter, T. Preferred Answer Sets for Extended Logic Programs, *Artificial Intelligence* 109, 297-356, 1999.

[4] Dimopoulos, Y., Kakas, A., and Moraitis, P. Argumentation Based Modeling of Embedded Agent Dialogues, in *ArgMAS'05*, Utrecht, The Netherlands, 2005.

[5] Dumas M., Governatori G., ter Hofstede, A.H.M., and Oaks, P. "A formal approach to negotiating agents development", in *Electronic Commerce and Application*s, Vol. 1, No 2, Elsevier Science Publications, 2002.

[6] Faratin, P., Sierra, C., and Jennings, N.R. "Using similarity criteria to make issue trade-offs in automated negotiation", in *Artificial Intelligence* 142 (2) 205-237, 2001.

[7] Hamblin, C.L. *Fallacies*, Methuen, London, 1970.

[8] Jennings, N. R., Faratin, P., Lomuscio, A.R., Parsons, S., Sierra C., and Wooldridge, M. Automated Negotiation: Prospects, methods and challenges, in *J. of Group Decision and Negotiation* 10 (2) 199-215, 2001.

[9] Kakas A. and Moraitis, P. Argumentative Agent Deliberation, Roles and Context, in *Electronic Notes in Theoretical Computer Science, (ENTCS)*, 70(5), 2002.

[10] Kakas, A. and Moraitis, P. Argumentation based decision making for autonomous agents", in *Proc. AAMAS'03*, pp. 883-890, 2003.

[11] Kakas, A., Mancarella P., and Dung, P.M. The acceptability semantics for logic programs, in *Proc. ICLP'94*, pp. 504-519, 1994.

[12] Kakas, A., Maudet, N., and Moraitis, P. Modular Representation of Agent Interaction Rules through Argumentation, in *J. of AAMAS*, Springer, vol. 11, no. 2, pp. 189-206, 2005.

[13] MacKenzie, J. Question-begging in non-cumulative systems, in *Journal of Philosophical Logic*, 8:117-133, 1979.

[14] Parsons, S., Sierra, C., and Jennings, N.R. Agents that reason and negotiate by arguing, in *Journal of Logic and Computation*, 8(3), pp. 261-292, 1998.

[15] Prakken, H. and Sartor, G. A dialectical model of assessing conflicting arguments in legal reasoning, in *Journal of Artificial Intelligence and Law*, Vol. 4, pp. 331-368, 1996.

[16] Rahwan, I., Ramchurn, S. D., Jennings, N. R., McBurney, P., Parsons S., and Sonenberg, L. Argumentation-based negotiation, in *Knowledge Engineering Review*, 18 (4) 343-375, 2003.

[17] Rosenschein, J.S. and Zlotkin, G. *Roles of Encounter*, MIT Press, 1994.

[18] Sandholm, T.W. Distributed Rational Decision Making, In *Multiagent Systems: A modern approach to distributed artificial intelligence*, G. Weiss (eds.), pp. 201-258, 1999.

[19] Sycara, K. Argumentation: planning other agents' plans, in *Proc. IJCAI'89*, pp. 517-523, 1989.

[20] Gorgias: Argumentation with Abduction, http://www.cs.ucy.ac.cy/~nkd/gorgias

Monotonic Concession Protocols for Multilateral Negotiation

Ulle Endriss
Institute for Logic, Language and Computation
University of Amsterdam, The Netherlands
ulle@illc.uva.nl

ABSTRACT

The most natural way of thinking about negotiation is probably a situation whereby each of the parties involved initially make a proposal that is particularly beneficial to themselves and then incrementally revise their earlier proposals in order to come to an agreement. This idea has been formalised in the so-called monotonic concession protocol, a set of rules defining the range of acceptable moves during a negotiation process intended to follow this general scheme. In the case of negotiation between just two agents, the monotonic concession protocol has become a textbook example and its formal properties are well-understood. In the case of multilateral negotiation, where more than two agents need to come to an agreement, on the other hand, it is not at all clear how to set up a monotonic concession protocol. As it turns out, the design of such a protocol boils down to the question of what constitutes a multilateral concession. In this paper, we make several proposals as to what might be an appropriate definition and analyse the properties of the proposed concession criteria.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*; J.4 [**Social and Behavioral Sciences**]: Economics; K.4.4 [**Computers and Society**]: Electronic Commerce

General Terms

Economics, Theory

Keywords

Negotiation, Protocols, Game Theory

1. INTRODUCTION

Negotiation between self-interested agents is a blossoming research area at the heart of multiagent systems, incorporating many ideas and models originally developed in different branches of economics (notably game theory), and at the same time benefiting significantly from the adoption of computational methods [6, 7, 10]. Much work on negotiation is either concerned with *bilateral* ("one-to-one") negotiation [10] or with *auctions* [2]. The latter are highly structured forms of negotiation between several agents.[1] Truly *multilateral* negotiation, on the contrary, where groups of more than just two agents can freely come together and agree on a deal between them has received far less attention. This is very much due to the significant technical challenges posed by such rich negotiation models (and certainly not because they would not be considered important).

For instance, in the context of distributed task and resource allocation, it is well-known that protocols only permitting two agents at a time to negotiate agreements are usually not expressive enough to reach optimal allocations [3, 11]. A typical example would be a scenario with three agents and three resources. Think of the agents as being arranged in a circle. Suppose the agents all differ in their assessment of the respective values of the three items and also suppose that, initially, each agent holds their second-most preferred item, their lefthand neighbour holds their favourite item, and their righthand neighbour holds their least-preferred item. Now any potential deal between two agents would either involve one of them losing all resources, or having to exchange their second-best item for the worst one. Neither would be considered beneficial by that agent, *i.e.* a *rational* agent would have to refuse any such deal.[2] Hence, using bilateral deals alone, it is not possible for rational agents to move away from the initial allocation and reach the ideal situation where each agent gets their preferred resource. Such limitations can be overcome by negotiation mechanisms that allow for more than just two agents to make a deal. In our example, for instance, it is clear that there is a multilateral deal between all three agents that would be considered rational by all of them.

Other examples for the need for multilateral negotiation abound (international politics being just one of them). Clearly, there are many situations where the options to agree on deals between more than just two parties at a time can greatly enrich the opportunities available to the negotiators

[1] But note that the *outcome* of an auction is usually also just a collection of separate bilateral deals between the auctioneer and different bidders.

[2] Even with monetary side payments, it is not difficult to set up an example where any bilateral deal would be considered irrational by at least one of the agents concerned [3, 11].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

and can lead to much better negotiation outcomes. On the downside, of course, such multilateral negotiation models are a lot more complex than bilateral negotiation (and typically also more complex than the highly structured negotiation schemes between several agents given by auctions). Acknowledging these difficulties, this paper aims at clarifying some of the more fundamental issues at stake.

Specifically, in this paper we are interested in *monotonic concession protocols* for multilateral negotiation. A negotiation *protocol* is a set of rules specifying the range of "legal" moves available to each agent at any stage of a negotiation process (in contrast to this, an agent's negotiation *strategy* is a function from the negotiation history to a follow-up move respecting the protocol). Monotonic concession protocols, in particular, formalise what might be considered the most natural way of thinking about negotiations: Initially, each of the parties involved make a proposal that is particularly beneficial to themselves, and then they incrementally revise their earlier proposals in order to eventually come to an agreement. In the case of negotiation between two agents, the monotonic concession protocol has become a textbook example for negotiation mechanisms [10, 13] and its formal properties are well-understood. In the multilateral case, as we shall see, there appear to be many different ways of generalising the (single) definition for the two-agent case. Crucially, the design of multilateral monotonic concession protocols boils down to the question of what constitutes a concession to a group of agents, rather than to a single opponent. This paper identifies (at least some of) the available options, and analyses their formal properties. We also include a preliminary discussion of the question of how to design suitable negotiation strategies for these protocols.

Paper Overview

Section 2 reviews the monotonic concession protocol for the *two-agent case* and an associated negotiation strategy, as developed in the work of Zeuthen [15], Harsanyi [4], and Rosenschein and Zlotkin [10]. Following this, Section 3 sets out the general framework of generalising the protocol to the multilateral case. Such a generalisation requires us to decide what constitutes an *agreement* between several agents and, crucially, how to characterise a *concession* to more than one opponent. These concepts are discussed in Section 4. In particular, Section 4 presents seven alternative definitions of the term *multilateral concession*. Each of these gives rise to a different variant of the monotonic concession protocol. The *properties* of these variants of the basic protocol, in particular *termination* and *deadlock-freedom*, are then analysed in Section 5. Some of the issues pertaining to the definition of a suitable *strategy* in a multilateral negotiation context are discussed in Section 6. Section 7 concludes.

2. THE TWO-AGENT CASE

In this section, we are going to describe the monotonic concession protocol for *two* agents, as presented by Rosenschein and Zlotkin [10]. We are also going to discuss a particular negotiation strategy, originally introduced by Zeuthen [15] and shown to be based on sound game-theoretical foundations by Harsanyi [4], that agents may adopt during an interaction regulated by the protocol.

Notation and Assumptions

Let $\mathcal{A} = \{1, 2\}$ be a set of two agents and let $\mathcal{X}$ be a finite set of *potential agreements* (which we are also going to refer to as *deals* or *proposals*). Each of the agents $i \in \mathcal{A}$ is equipped with a *utility function* $u_i : \mathcal{X} \rightarrow \mathbb{R}_0^+$ mapping agreements to non-negative reals. That is, we exclude any potential agreements that would yield negative utility for either one of the agents *a priori*. $\mathcal{X}$ includes a specific agreement, called the *conflict deal*, that yields utility 0 for both agents. This will be the agreement chosen in case negotiation breaks down and is assumed to be the worst possible outcome.

Monotonic Concession Protocol

The monotonic concession protocol proceeds in *rounds*. In each round, both agents make simultaneous *proposals*, *i.e.* they each propose one of the potential agreements from $\mathcal{X}$. In the first round, each agent is free to make any proposal. In any subsequent round, each agent $i \in \mathcal{A}$ has got two options (let $x_i \in \mathcal{X}$ be the most recent proposal made by i):

(1) Make a *concession* and propose a new deal x_i' that is preferable to the other agent j: $u_j(x_i) < u_j(x_i')$.

(2) Refuse to make a concession and stick to proposal x_i.

An *agreement* is found when one agent makes a proposal that its opponent rates at least as high as its own current proposal, *i.e.* when $u_1(x_1) \leq u_1(x_2)$ or $u_2(x_2) \leq u_2(x_1)$ or both. *Conflict*, on the the hand, arises when both agents refuse to make a concession during a given round. The protocol runs until either an agreement is found or conflict occurs. In case of agreement, the proposal satisfying both agents is chosen (in case both proposals are satisfiable, we flip a coin to pick one). In case of conflict, the conflict deal is the outcome of the negotiation.

The monotonic concession protocol is known to always *terminate* [10], *i.e.* there can be no infinite sequence of rounds.

Zeuthen Strategy

What would be a suitable strategy for agents to adopt when following the monotonic concession protocol? Zeuthen [15] proposed that agents should evaluate their respective *willingness to risk conflict*, and that the agent with the lower value for this measure should make the next concession (and both agents should concede in case their risk values are equal). Furthermore, a concession should be *sufficient* to ensure that the other agent will have to concede in the following round.

Zeuthen proposed to calculate an agent's willingness to risk conflict as the quotient of the loss in utility incurred in case the other agent's proposal is being adopted and the loss incurred by causing conflict (both with respect to the utility of the agent's own current proposal). As the utility of the conflict deal is 0, the willingness of agent i to risk conflict (with j being the other agent) is given by the following formula (we assume $u_i(x_i) \neq 0$ for ease of presentation):

$$Z_i \quad = \quad \frac{u_i(x_i) - u_i(x_j)}{u_i(x_i)}$$

Zeuthen's intuitive justification for this strategy has been complemented by Harsanyi's work [4], who showed how the same strategy can be derived from a small number of *fundamental postulates*. The most important of these is the

postulate of *utility maximisation*. Harsanyi argues as follows: Suppose the two agents are playing a single-round game; each agent can either accept or reject the other's proposal. Let p_2 be the probability that agent 2 will reject x_1. Then the expected payoff for agent 1 of rejecting x_2 is $(1 - p_2) \cdot u_1(x_1)$. On the other hand, Harsanyi argues, the *certain* payoff associated with accepting x_2 is $u_1(x_2)$. Hence, agent 1 should accept iff $u_1(x_2) \geq (1 - p_2) \cdot u_1(x_1)$. This is equivalent to $\frac{u_1(x_1) - u_1(x_2)}{u_1(x_1)} \leq p_2$. Together with a *symmetry* and a *monotonicity* postulate, this leads to the Zeuthen criterion: agent 1 should concede whenever $Z_1 \leq Z_2$.[3]

The Zeuthen strategy is *efficient* in the sense of leading to a final agreement maximising the product of utilities [4, 10].[4] This follows from the fact that $Z_1 \leq Z_2$ is equivalent to $u_1(x_1) \cdot u_2(x_1) \leq u_1(x_2) \cdot u_2(x_2)$, *i.e.* in each round the agent whose current proposal yields the lower product of utilities makes a concession.

The strategy is also "almost" *stable* (*i.e.* in Nash equilibrium [9]). The only critical case is when both agents have the same willingness to risk conflict after the penultimate round. In that case, an agent could benefit from deviating and sticking to their latest proposal. This problem can be overcome by extending the Zeuthen strategy with a mixed equilibrium strategy for the two-person game induced by this "last step situation", albeit at the cost of giving up on efficiency [9, 10].

3. NEGOTIATION PROTOCOLS

In this section, we are going to discuss the design of negotiation protocols that can be used by more than just two agents.

Notation and Assumptions

Again, let $\mathcal{X}$ be a finite set of potential agreements and let $\mathcal{A}$ be a finite set of agents. Each agent $i \in \mathcal{A}$ is equipped with a utility function $u_i : \mathcal{X} \to \mathbb{R}_0^+$, *i.e.* we only admit non-negative utilities, and $\mathcal{X}$ includes the so-called conflict deal to which every agent assigns utility 0. In one case we are going to comment on the effects of only allowing strictly positive utilities (for anything but the conflict deal).

A Multilateral One-Step Protocol

We should stress that, in this paper, we are not interested in finding the "best" protocol for multilateral negotiation. Instead, we are only interested in generalising the ideas present in the two-agent monotonic concession protocol. Also in the two-agent case, it is well-known that the monotonic concession protocol is not the "best" protocol. Indeed, there is a simple alternative, termed the *one-step protocol* by Rosenschein and Zlotkin [10], that is much simpler than the monotonic concession protocol and permits a negotiation strategy that is both stable and efficient. Recall that the Zeuthen strategy cannot guarantee both stability *and* efficiency (depending on how the "last step situation" is handled, either stability or efficiency need to be sacrificed). The idea of the one-step protocol is to simply ask both agents for a proposal and to adopt the proposal that yields the higher product of utilities (and to flip a coin in case the products are the same). The best strategy that agents can follow in this protocol is to propose the agreement that is best for themselves amongst those with maximal product of utilities. This is clearly both stable and efficient.

The one-step protocol can easily be extended to the multilateral case: Collect a proposal from each agent and randomly choose an outcome amongst those proposals that maximise the product of utilities. The obvious strategy, which is the same as for the two-agent case, is also both stable and efficient. This simplistic solution not withstanding, we still remain interested in the monotonic concession protocol, because —unlike the one-step protocol— it is a direct formalisation of natural negotiation behaviour.

Multilateral Monotonic Concession Protocols

How can we generalise the monotonic concession protocol to cases where more than two agents need to come to an agreement? The general structure of the negotiation protocol will stay the same:

(1) In the first round, each agent makes an initial proposal.

(2) In each subsequent round, each agent either makes a *concession* or sticks with their current proposal.

(3) The above is being iterated until either a conflict situation arises (no agent makes a concession) or *agreement* is reached.

The two critical terms here, which we have as yet left undefined, are *concession* and *agreement*.

4. AGREEMENT AND CONCESSION

In this section, we are going to discuss the meaning of the terms *agreement* and *concession* in the context of multilateral negotiation.

Multilateral Agreements

In the case of agreement, the generalisation from the two-agent case to the general multilateral case is straightforward:

> *Agreement is reached iff one agent makes a proposal that is at least as good for each other agent as their own current proposal.*

To put it more formally, agreement is reached iff there exists an agent $i \in \mathcal{A}$ such that $u_j(x_i) \geq u_j(x_j)$ for all agents $j \in \mathcal{A}$.

Options for Defining Multilateral Concessions

In the case of concession, however, it is not at all clear what would be the "correct" way of generalising to the multilateral case. Indeed, what does it mean to make a concession to a group of opponents? Do we have to make our new proposal more palatable to everyone else, to just one of them, or do we have to propose something that would somehow increase the utility of the group as a whole (and what would that actually mean)?

[3] What may be considered questionable in Harsanyi's argument is that he implicitly assumes that x_2 will be selected as soon as agent 1 accepts it. This disregards the possibility that both agents accept each others proposal, in which case the choice should be random. However, interestingly, this more complex assumption leads to the same solution. In this case, the expected payoff for accepting x_2 is $p_2 \cdot u_1(x_2) + (1 - p_2) \cdot \frac{1}{2}(u_1(x_1) + u_1(x_2))$. Using this in place of $u_1(x_2)$ eventually leads to the same inequation $Z_1 \leq Z_2$

[4] This also implies that the final agreement will be *Pareto optimal* (no alternative agreement would be better for one agent without decreasing utility for the other [8]).

All of the following interpretations of the term *concession* seem to have some merit:

(1) *Strong concession:* Make a proposal that is strictly better for each of the other agents.

(2) *Weak concession:* Make a proposal that is strictly better for at least one of the other agents.

(3) *Pareto concession:* Make a proposal that is at least as good for all of the other agents and strictly better for at least one of them.

(4) *Utilitarian concession:* Make a proposal such that the sum of utilities of the other agents increases (utilitarian social welfare).

(5) *Egalitarian concession:* Make a proposal such that the minimum utility amongst the other agents increases (egalitarian social welfare).

(6) *Nash concession:* Make a proposal such that the product of utilities of the other agents increases (Nash product).

(7) *Egocentric concession:* Make a proposal that is worse for yourself.[5]

We are first going to clarify the choice of name for some of these concession criteria and then turn to the analysis of their properties.

Relation to Social Welfare Concepts

Recall that, in welfare economics, a *Pareto improvement* is a move to a different agreement that harms nobody but benefits at least one member of society (and a Pareto optimal agreement is one that does not admit any further Pareto improvements). Analogously, a Pareto *concession* by agent i is a concession that results in a Pareto improvement for the group of agents $\mathcal{A} \setminus \{i\}$. The names for criteria (4)–(6) are also inspired by concepts from welfare economics. These are all concepts that are used to assess the well-being of a group (or *society*) of agents in terms of the individual utilities of its members [1, 8].

The *utilitarian social welfare* enjoyed by a group, for instance, is defined as the sum of the utilities of its members. This is a useful metric for assessing the well-being of a group if we adopt a classical utilitarian point of view (note that this is also a metric for *average* utility, *i.e.* an agent making a utilitarian concession would improve the average utility of its opponents). The *egalitarian social welfare*, on the other hand, is defined as the utility of the agent with the currently lowest utility level (*i.e.* in an egalitarian system we would measure the success of society in terms of the well-being of its poorest members). The *Nash product* is an interesting compromise; it measures social welfare in terms of the product of individual utilities. This means that both overall increases in utility and inequality-reducing utility transfers would be considered beneficial to society. Other notions of social welfare studied in the literature would give rise to further concession criteria.

[5]Here *egocentric* is not to be misread as *selfish*; it simply means that this notion of concession is based on the agent's own valuations rather than that of other agents. This makes sense in domains where a reduction in utility for one agent can generally be assumed to result in an increase in utility for (some of) the other agents.

Relation to the Two-Agent Case

If there are just two agents, then making a concession of either one of the first six types simply amounts to making a new proposal that is strictly better for your (single) opponent. Hence, criteria (1)–(6) are all faithful extensions of the two-agent case.

In fact, criterion (7) is also a faithful extension of the two-agent case, at least if we assume that an agent would never make a proposal that is dominated by another potential proposal. This is so, because moving to a new proposal that is worse for yourself but not better for your opponent than your current proposal amounts to moving to a proposal that is dominated by your current proposal.

5. PROTOCOL PROPERTIES

In this section, we are going to analyse various interesting properties for the different monotonic concession protocols entailed by our seven multilateral concession criteria.

Termination

An important property of a negotiation protocol is whether it can guarantee that any negotiation process following the protocol will eventually terminate. Each of the multilateral concession criteria we have proposed gives rise to a different negotiation protocol. The following proposition summarises some simple termination results for these protocols:

PROPOSITION 1. *Except for weak concessions, each of the seven proposed concession criteria guarantees termination of the negotiation protocol.*

PROOF. Each of the concession criteria, except that of weak concession, is defined in terms of a metric that induces a strict partial (in some cases even total) order $\prec_i$ over the space of proposals from the viewpoint of each individual agent i. For instance, in the case of a utilitarian concession by agent i, the sum of utilities for the remaining agents has to increase, *i.e.* you may define $x \prec_i y$ iff $\sum_{k\in\mathcal{A}\setminus\{i\}} u_k(x) < \sum_{k\in\mathcal{A}\setminus\{i\}} u_k(y)$. Recall that the overall number of possible proposals is assumed to be finite. Hence, each agent i can only make a finite number of concessions, as each concession will take its own current proposal higher up in the $\prec_i$-order. Hence, as the overall number of agents is assumed to be finite as well, there can only be a finite number of concessions overall, *i.e.* negotiation is bound to terminate and either conflict or an agreement will be reached eventually. □

Note that any such termination result can only ever apply to the *protocol* as such, not to negotiation processes in general. If one of the agents does not make a move during one of the rounds (because, for instance, its reasoning process for deciding the next move does not terminate), then the negotiation gets stuck. In practice, this problem can be tackled by introducing deadlines ("if you do not make a move within 1 minute, then this counts as a refusal to concede").

Clearly, in the case of weak concessions, we cannot guarantee termination (unless there are only two agents). Indeed, an agent could alternate between two proposals indefinitely and make a weak concession each time. This is possible when there are two candidate proposals, such that one of them is strictly preferred by the agent's first opponent and the other is strictly preferred by its second opponent.

Compositionality

Closely related to the question of termination is what one might want to term *compositionality*. This property holds iff the composition of two consecutive concessions each meeting a given concession criterion will always meet that same criterion as well. That is, whenever the move from x to x' is considered a valid concession, and the move from x' to x'' is as well, then the direct move from x to x'' will constitute a valid concession as well.

Clearly, all of our proposed concession criteria satisfy the compositionality property, with the sole exception of weak concessions. In the case of a Nash concession, for instance, if two consecutive concessions each increase the product of utilities for the other agents, then the product of utilities will have increased as well with respect to the entire negotiation process. For weak concessions, on the other hand, the same counterexample given for termination applies: the second concession may take us back to the initial proposal.

Deadlock-freedom

Another important property of negotiation protocols to consider is deadlock-freedom. A *deadlock* is a situation where no agent can make any further move that would be compliant with the negotiation protocol, but none of the terminal states (either agreement of conflict) of the protocol has been reached either. Of course, in the context of a monotonic concession protocol, each agent always has the move of not conceding available to them. That is, for any of our concession criteria, it will always be possible for agents to continue negotiation in a fashion deemed legal by the protocol. However, there may be situations where an agent would not be able to make a concession, even if they would be prepared to give up their current position. Naturally, we would prefer to avoid such situations and use a concession criterion satisfying the following property of deadlock-freedom:

> *We call a concession criterion deadlock-free iff it guarantees that at least one agent can make a concession satisfying the criterion at any stage during negotiation, until an agreement has been reached.*

It is not difficult to see that any of our criteria must be deadlock-free in case there are only two agents in the system:

PROPOSITION 2. *In the two-agent case, each of our seven concession criteria is deadlock-free.*

PROOF. Let x_1 and x_2 be the current proposals of agents 1 and 2, respectively. If no agreement has been reached yet, then we must have both $u_1(x_1) > u_1(x_2)$ and $u_2(x_2) > u_2(x_1)$. As argued earlier, in the two-agent case criteria (1)–(6) coincide; they are fulfilled if one of the agents makes a new proposal that is better for its opponent than its current offer. Hence, x_2 would be a legal concession for agent 1 for any of the criteria (1)–(6). Furthermore, it would also be a legal concession according to criterion (7). Hence, there can never be a deadlock in the two-agent case. □

In the general case, with potentially more than just two agents, however, the situation is different. Now just some of our criteria are deadlock-free, while others are not.

Furthermore, for one of them (the Nash concession), deadlock-freedom depends on a seemingly minor detail of the negotiation scenario. Recall that we require all utility functions to be non-negative. We say that utilities are *required to be positive* in case we do require all utilities, except for the conflict deal, to be greater than 0. We are now able to state our results regarding deadlock-freedom:

PROPOSITION 3. *The weak, the utilitarian, and the egocentric concession criteria are all deadlock-free. The Pareto, the strong, and the egalitarian concession criteria are not deadlock-free. The Nash concession criterion is deadlock-free iff utilities are required to be positive.*

PROOF. We begin with the negative results, more specifically with the case of *Pareto concessions*. Suppose there are three agents with the following current proposals (which are also the only possible outcomes) and associated utilities:

	x_1	x_2	x_3
u_1	4	2	2
u_2	3	2	2
u_3	1	2	2

It is easy to see that agreement has not been reached and that no agent is in a position to make a Pareto concession. Hence, this example constitutes a deadlock. Every *strong concession* is also a Pareto concession, *i.e.* strong concessions cannot be deadlock-free either.

To construct an example that proves that *egalitarian concessions* are not deadlock-free, let ϵ be the minimal utility assigned to any outcome by any agent. Then in any game where for each possible outcome x there are at least two agents assigning utility ϵ to x, no agent can ever make an egalitarian concession (because the minimum utility of the other agents is the same for all possible proposals).

Now let us move on to the positive results. The *weak concession* criterion is clearly deadlock-free. In fact, at any stage *each* agent i could make a weak concession (until an agreement is reached). If no agreement has been reached, then there must be at least one other agent j such that $u_j(x_j) > u_j(x_i)$. But then proposing x_j would constitute a weak concession for i. An analogous argument can be used for the case of *egocentric concessions*.

Next consider the case of *utilitarian concessions*. Let i be (one of) the agent(s) making a proposal x_i yielding a maximal sum of utilities (amongst the current proposals). If there is no agreement yet, then there must be another agent j who likes x_i less than their own proposal: $u_j(x_i) < u_j(x_j)$. But we also have $\sum_{k\in\mathcal{A}} u_k(x_i) \geq \sum_{k\in\mathcal{A}} u_k(x_j)$. Hence, $\sum_{k\in\mathcal{A}\setminus\{j\}} u_k(x_i) > \sum_{k\in\mathcal{A}\setminus\{j\}} u_k(x_j)$. That is, x_i constitutes a legal utilitarian concession for agent j and there is no deadlock.

An analogous argument can be used in the case of *Nash concessions*. However, in this case the validity of the last step in the argument relies on $u_j(x_i)$ being greater than 0. That is, this proof will only go through in case utilities can be assumed to be positive. If zero utilities are admissible, then we can use a similar argument as the one used in the case of egalitarian concessions to show that Nash concessions are *not* deadlock-free. Take a game where for each possible outcome x there are at least two agents assigning utility 0 to x. Then the product of utilities of the "other" agents will always be 0, *i.e.* no agent can ever make a Nash concession. □

Lack of deadlock-freedom is a serious deficiency of a negotiation protocol. Arguably, deadlock-freedom is even more im-

portant than termination, because a lack of termination can be addressed by refining the concession criterion in question, while deadlock-freedom could only be recovered by loosening the definition of a criterion.

Wooldridge *et al.* [14], for instance, who propose a multilateral negotiation mechanism that draws on ideas from the monotonic concession protocol for the two-agent case, require concessions to satisfy both the utilitarian and the Pareto concession criteria. They recognise that their approach is not deadlock-free (without using that expression), and suggest to overcome this problem by introducing the option of *backtracking*, *i.e.* of proposing a deal that is neither a concession nor the same deal proposed in the previous round. Arguably, such an extension of the protocol takes us outside of the realm of monotonic concession protocols.

Verifiability and Privacy

Next we turn our attention to two inter-related issues: *verifiability* and *privacy*. A negotiation protocol should be verifiable in the sense that it should be possible to check whether everybody is really following the rules laid down by the protocol. But who should check?

The first type of verifiability is verifiability by an *outside observer*. Clearly, an outside observer could verify some of the more operational issues, e.g. whether agents really make simultaneous proposals. However, an outside observer usually cannot check whether every agent who is changing their proposal is really making a true concession. This would require an accurate knowledge of the utility functions of the agents concerned and, for reasons of privacy, these agents may not be prepared to publish their utility functions (or if they do, they may not always be entirely truthful about them). Of course, if we assume the presence of an *omniscient observer*, then full verifiability is clearly given for any of the variants of the protocol considered here, but it is also not very interesting.

The issue of privacy does not just affect verifiability, but is a somewhat critical issue about the whole setup of the monotonic concession protocol. Nevertheless, the two-agent monotonic concession protocol has the very desirable property that each individual agent can verify whether their opponent is conforming to the rules of the protocol. They only need to know their *own* utility function. Whether or not they report their utility function truthfully is up to them, but they can always check whether a new proposal constitutes a concession with respect to their true utility function (and, of course, also with respect to their reported utility function). In the case of more than two agents, this is not the case anymore, simply because concession criteria are defined in terms of more than one utility function (except for the egocentric criterion).

In the multilateral case, it seems interesting to introduce a concept of *distributed verifiability*. Is it possible that all the other agents together verify whether or not the latest move by a particular agent constitutes a true concession? Of course, the other agents may also be lying to each other, but we can at least check whether a concession criterion admits distributed verifiability in the following sense:

> *A concession criterion admits distributed verification iff for any move that is not a true concession, at least one of the other agents can detect that violation.*

It turns out that only the strong concession criterion is verifiable in a distributed manner, although the weak and the Pareto concession criteria also admit a slightly weaker form of distributed verification:

(1) The *egocentric* concession criterion is not verifiable, because only the proposing agent could check.

(2) The *utilitarian*, *egalitarian*, and *Nash* concession criteria are not verifiable in the distributed sense, because no individual agent can check whether the sum/minimum/product has increased.

(3) The *strong* concession criterion is verifiable in a distributed manner: for any violation, at least one of the agents will be worse off.

(4) The *weak* and the *Pareto* concession criteria are verifiable in a distributed manner if agents can communicate (if an agent has received a strict improvement they should announce that fact). If no agent reports a strict improvement, then a violation has occurred (in addition, for a Pareto concession, each agent would have to check that they did not suffer a loss in utility).

This concludes our discussion of the properties of the seven variants of the monotonic concession protocol.

6. NEGOTIATION STRATEGIES

Our main interest in this paper has been concerned with monotonic concession *protocols* for multilateral negotiation (and thereby also the nature of multilateral concessions), but this investigation also clearly raises the question what *strategy* an individual agent participating in a negotiation regulated by such a protocol should follow. A proper game-theoretical analysis of this question would be of considerable interest, but also technically demanding. Such an analysis lies outside the scope of this paper.

Still, in this section, we want to briefly review a number of seemingly obvious choices.

Choicepoints

It seems reasonable to assume that agents always start with a proposal that is optimal for themselves. Of course, from a "psychological" point of view it may be considered advantageous to start with a more realistic proposal straight away, but we are not going to be concerned with such questions here. We are also going to ignore the question as to *which* of the optimal proposals to select in case there are several.

In every subsequent round, each agent has to answer two questions:

(1) Should I make a concession?

(2) If yes, how much should I concede?

Of course, just as it is unclear how to characterise the concept of a concession in a multilateral negotiation context, it may also seem unclear how to quantify the extent of such a concession. Nevertheless, there is in fact a rather simple answer to our second question: A concession should always be *minimal* with respect to the utility loss incurred by the agent making the concession. That is, amongst the pool of potential proposals that would satisfy the concession criterion enforced by the protocol, the agent should choose the

proposal that gives the highest utility to themselves (again, in this tentative discussion, we are going to ignore the question of what to do in case there are several potential concessions yielding maximal utility).

The concept of minimal concessions is useful when discussing these questions at an abstract level. In practice, it can and should be further refined. For the Zeuthen strategy as formulated by Rosenschein and Zlotkin [10], for instance, agents are supposed to make *sufficient* rather than *minimal* concessions (sufficient in the sense of forcing the opponent to concede in the next round). Of course, for most concession criteria, a sufficient concession can always be decomposed into a series of minimal concessions; so there is no important qualitative difference at the abstract level.[6]

To summarise, the question of strategy reduces to the question of *when* to make a concession. In the two-agent case, the Zeuthen strategy settles this question by letting the agents compute the *willingness to risk conflict* for both agents and asking the agent for whom this value is lower to make the next concession (with suitable modifications in case the two values are equal). In the sequel, we are going to consider three possible generalisations of the Zeuthen strategy to the multilateral case.

Willingness to Risk Conflict

In the two-agent case, the Zeuthen strategy is built around the concept of an agent's willingness to risk conflict. This value is computed as the quotient of the utility loss incurred by accepting the opponent's proposal and the utility loss incurred by settling with the conflict deal (both with respect to the utility of the agent's own current proposal). So for agent 1, we get the following value: $\frac{u_1(x_1)-u_1(x_2)}{u_1(x_1)}$.

A natural generalisation of this criterion to the multilateral case would be to evaluate the loss in utility in case of concession assuming the worst possible outcome for the agent. That is, for agent i, we obtain the following formula for its willingness to risk conflict:

$$Z_i \quad = \quad \begin{cases} 1 & \text{if } u_i(x_i) = 0 \\ \dfrac{u_i(x_i) - \min\{u_i(x_k) \mid k \in \mathcal{A}\}}{u_i(x_i)} & \text{otherwise} \end{cases}$$

Then the agent with the lowest willingness to risk conflict should make the next concession. This criterion has also been proposed by Wooldridge *et al.* [14].

While this strategy seems to be a natural generalisation of the two-agent Zeuthen strategy, it can unfortunately lead to a deadlock. Consider the following example. There are three agents and three possible outcomes. Suppose agent 1 is currently proposing x_1, agent 2 is proposing x_2, and agent 3 is proposing x_3:

	x_1	x_2	x_3
u_1	10	8	8
u_2	9	10	5
u_3	9	5	10

$\rightsquigarrow$ $Z_1 = 0.2$, $Z_2 = 0.5$, $Z_3 = 0.5$

Note that agreement has not yet been reached. According to our strategy, agent 1 should make a concession. However, agent 1 is not *able* to make a concession that would either affect the Z-value of any of its opponents or increase its own.

[6] An exception to this rule are the weak concessions, which —as we have seen earlier— are not decomposable. That is, here choosing a certain minimal concession may block the way to a suitable sufficient concession later on.

This observation suggests that simply checking for which agent the Z-value is minimal is not a sufficiently sophisticated criterion for deciding who should make the next concession. A potential way around this problem may be to select the agent with the minimal Z-value amongst those that are actually able to make an *effective* concession.

A Product-increasing Strategy

As discussed in Section 2, Harsanyi [4] showed that the Zeuthen strategy corresponds to a product-increasing strategy. That is, the agent whose proposal yields the lower product of utilities should make the next concession. This suggests another obvious generalisation of the strategy, namely that also in the multilateral case any agent whose proposal yields a minimal product of utilities should make a concession and propose a new deal with a higher product.

$$Z_i' \quad = \quad \prod_{k \in \mathcal{A}} u_k(x_i)$$

One may be inclined to believe that this strategy is bound to result in an agreement with maximal product of utilities, *i.e.* that the product-increasing strategy should be efficient. However, it is possible to construct scenarios where this is not so. The problem is that also a proposal with non-maximal product may be found to be acceptable to everyone. Take the following example (suppose agent 1 proposes x_1, agent 2 proposes x_2, and agent 3 proposes x_3):

	x_1	x_2	x_3
u_1	5	4	4
u_2	3	2	10
u_3	3	10	2

$\rightsquigarrow$ $u_1(x_1) \cdot u_2(x_1) \cdot u_3(x_1) = 45$, $u_1(x_2) \cdot u_2(x_2) \cdot u_3(x_2) = 80$, $u_1(x_3) \cdot u_2(x_3) \cdot u_3(x_3) = 80$

Here x_1 is the proposal with the lowest product of utilities, but it is also the only proposal that would be acceptable to all agents (as it gives them at least as much utility as their own current proposals). Whether or not such a situation could really occur in an actual negotiation remains to be investigated (note that the example is somewhat unrealistic, because agents 2 and 3 did not propose their favourite agreements first).

Sum of Products of Pairs

Yet another potential generalisation of the Zeuthen strategy, also building on its interpretation as a product-increasing strategy, would be to compute the sum of products of pairs of utilities for every proposal and to ask the agent for whose proposal that value is minimal to make a concession.

$$Z_i'' \quad = \quad \sum_{j \neq k \in \mathcal{A}} u_j(x_i) \cdot u_k(x_i)$$

This appears to be another promising approach, although further investigation is required before any kind of conclusive judgement would be possible.

Stability

We conclude our preliminary discussion of issues surrounding the definition of suitable strategies for multilateral negotiation with a general observation regarding the stability of such mechanisms. Recall that a set of strategies is said to be stable (or in Nash equilibrium) iff no individual agent would have an incentive to deviate unilaterally from their assigned strategy [9].

Any pure strategy where conflict occurs whenever a single agent deviates is stable. This follows from the fact that conflict is assumed to be the worst possible outcome for any of the participating agents. Hence, as in the two-agent case, the only critical situation is when several agents should concede in the same round (and in fact this only makes a difference when this situation occurs immediately before an agreement is found). Whatever the exact criterion may be, to guarantee stability it would have to be enhanced with a mixed equilibrium strategy for the n-person game of the final round.

7. CONCLUSION

As we have argued in the introduction, *multilateral negotiation* is often a necessity, *i.e.* it may not be possible to decompose a multilateral deal into a sequence of bilateral deals that would still be acceptable to all parties. In this paper, we have discussed the generalisation of the well-known *monotonic concession protocol* for two negotiating agents to a multilateral scenario where any number of agents should be able to forge agreements. As we have seen, the central question in this context concerns the appropriate definition of a multilateral concession. We have proposed seven alternative definitions, and also argued that there may well be further candidate definitions of interest. Further social welfare orderings, such as the *leximin* ordering for example [8], could provide useful hints for additional definitions.

We have then analysed some of the properties of the negotiation protocols induced by these different definitions: termination, compositionality, deadlock-freedom, and verifiability. All but one of the proposed concession criteria, namely weak concessions, guarantee termination; and weak concessions are also the only ones that are not composable. Our results on deadlock-freedom show that some of the proposed criteria (weak, utilitarian, and egocentric) are deadlock-free, while others are not (Pareto, strong, and egalitarian). Whether or not the Nash concession criterion is deadlock-free depends on the restrictions we put on utility functions, namely whether or not they are required to assign strictly positive utilities to all outcomes but the conflict deal. Our discussion of verifiability issues shows that only the strong concession criterion is verifiable by the participating agents in a distributed manner. The weak and the Pareto criterion can also be taken to be verifiable under certain assumptions on the ability and willingness of the agents to communicate with each other on this matter.

Finally, we have also discussed some of the issues at stake when trying to define a suitable negotiation strategy for agents following a multilateral monotonic concession protocol. In particular, we have suggested three possible generalisations of the well-known Zeuthen strategy, but also pointed out some of the problems inherent in these somewhat simplistic generalisations. A serious game-theoretical analysis of the problem promises to provide a number of new and very interesting insights.

Monotonic concession protocols are but one way of coming to an agreement within a group of agents. As discussed already at the beginning of Section 3, a simple one-step protocol that directly selects a Nash bargaining solution, *i.e.* an agreement maximising the product of individual agent utilities [9, 10], would be another option. In negotiation domains with transferable utilities (*i.e.* if agents can reimburse each other through monetary side payments, and there is global agreement on the marginal utilities associated with such payments), the well-known Shapley value [8, 9] provides yet another solution. The work of Hart and Mas-Colell [5] offers a unified view on this type of approach to multilateral negotiation.

We should stress again that we do not claim that monotonic concession protocols are the "best" way of regulating negotiation, or even that they would be appropriate in all cases. Winoto *et al.* [12], for example, argue against monotonic concessions in domains where utilities can change over time. For instance, a seller's valuation of a perishable good may be assumed to decrease over time, *i.e.* a potential buyer may be well-advised to lower (rather than improve) their offers in case negotiation continues over a significant amount of time. Our argument in favour of monotonic concession protocols is simply that they are such an immediate formalisation of our everyday-understanding of negotiation.

8. REFERENCES

[1] K. J. Arrow, A. K. Sen, and K. Suzumura, editors. *Handbook of Social Choice and Welfare*. North-Holland, 2002.

[2] P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006.

[3] U. Endriss, N. Maudet, F. Sadri, and F. Toni. On optimal outcomes of negotiations over resources. In *Proc. 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2003)*. ACM Press, 2003.

[4] J. C. Harsanyi. Approaches to the bargaining problem before and after the theory of games: A critical discussion of Zeuthen's, Hick's, and Nash's theories. *Econometrica*, 24(2):144–157, 1956.

[5] S. Hart and A. Mas-Colell. Bargaining and value. *Econometrica*, 64(2):357–380, 1996.

[6] N. R. Jennings, P. Faratin, A. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: Prospects, methods and challenges. *Group Decision and Negotiation*, 10(2):199–215, 2001.

[7] S. Kraus. *Strategic Negotiation in Multiagent Environments*. MIT Press, 2001.

[8] H. Moulin. *Axioms of Cooperative Decision Making*. Cambridge University Press, 1988.

[9] R. B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, 1991.

[10] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter*. MIT Press, 1994.

[11] T. W. Sandholm. Contract types for satisficing task allocation: I Theoretical results. In *Proc. AAAI Spring Symposium: Satisficing Models*, 1998.

[12] P. Winoto, G. I. McCalla, and J. Vassileva. Non-monotonic-offers bargaining protocol. *Journal of Autonomous Agents and Multiagent Systems*, 11(1):45–67, 2005.

[13] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley and Sons, 2002.

[14] M. Wooldridge, S. Bussmann, and M. Klosterberg. Production sequencing as negotiation. In *Proc. 1st International Conference on Practical Application of Agents and Multi-Agent Systems (PAAM-1996)*, 1996.

[15] F. Zeuthen. *Problems of Monopoly and Economic Warfare*. Routledge, 1930.

Negotiating using Rewards

Sarvapali D. Ramchurn[1], Carles Sierra[2], Lluis Godo[2], and Nicholas R. Jennings[1]

[1]School of Electronics and Computer Science,
University of Southampton, Southampton,
SO17 1BJ, UK.
{sdr,nrj}@ecs.soton.ac.uk

[2]Institute of Artificial Intelligence, CSIC,
08193 Bellaterra,
Spain.
{sierra,godo}@iiia.csic.es

ABSTRACT

In situations where self-interested agents interact repeatedly, it is important that they are endowed with negotiation techniques that enable them to reach agreements that are profitable in the long run. To this end, we devise a novel negotiation algorithm that generates promises of rewards in future interactions, as a means of permitting agents to reach better agreements, in a shorter time, in the present encounter. Moreover, we thus develop a specific negotiation tactic based on this reward generation algorithm and show that it can achieve significantly bettter outcomes than existing benchmark tactics that do not use such inducements. Specifically, we show, via empirical evaluation, that our tactic can lead to a 26% improvement in the utility of deals that are made and that 21 times fewer messages need to be exchanged in order to achieve this under concrete settings.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multi-Agent Systems

General Terms

Algorithms, Experimentation

Keywords

Persuasive Negotiation, Bargaining, Argumentation

1. INTRODUCTION

Negotiation is a fundamental concept in multi-agent systems (MAS) because it enables (self-interested) agents to find agreements and partition resources efficiently and effectively. Recently, a growing body of work has advocated the use of arguments as a means of finding good agreements [9]. Specifically, it is hypothesised that negotiation using persuasive arguments (such as threats, promises of future rewards, and appeals) allows agents to influence each others' preferences to reach better deals either individually or as a group. Most approaches to *persuasive negotiation* (PN), however, either focus mainly on the protocol [6, 8] used to argue and do not give any insight into the negotiation strategies to be used or fail to give clear semantics for the arguments that are exchanged in terms of their relationship with the negotiated issues [11, 5]. Moreover, most PN reasoning mechanisms adopt a defeasible logic approach [1, 9], rather than the utilitarian approach that we use here. The downside of this logic focus is that it cannot cope as well with the many forms of uncertainty that inevitably arise in such encounters and that it can hardly be benchmarked against standard negotiation algorithms [2, 3].

Against this background, in this work we present a novel reasoning mechanism and protocol for agents to engage in persuasive negotiation in the context of repeated games. We choose repeated games because it is a type of encounter where we believe that persuasive techniques are likely to be most effective (as arguments can be constructed to directly impact future encounters). Now, such encounters have been extensively analysed in game theory [7], but are seldom considered by agent-based negotiation mechanisms. This is a serious shortcoming because in many applications agents need to interact more than once. Specifically, our mechanism constructs possible rewards[1] in terms of constraints on issues to be negotiated in future encounters (hence their semantics are directly connected to the negotiated issues) and our protocol is an extension of Rubinstein's alternating offers protocol [12] that allows agents to negotiate by exchanging arguments (in the form of promises of future rewards or requests for such promises in future encounters).

In more detail, our mechanism gives agents a means of influencing current and future negotiations through promises of rewards, rather than just exchanging offers and counter offers that only impact on the outcome of the present encounter [5, 11]. Thus, we make the rewards endogenous to the negotiation process by assimilating a promise of a reward to promised constraints on resources that need to be negotiated in future. In so doing, we directly connect the value of the argument to the value of the negotiated issues and this allows us to evaluate arguments and offers on the same scale. For example, a car seller may reward a buyer (or the buyer might ask for the reward) who prefers red cars with a promise of a discount of at least 10% (i.e. a constraint on the price the seller can propose next time) on the price of her yearly car servicing if she agrees to buy a blue one instead at the demanded price (as the buyer's asking price for the red car is too low for the seller). Now, if the buyer accepts, it is a better outcome for both parties (the buyer benefits because she is able to make savings in the future that match her preference for the red car and the seller benefits in that he reduces his stock and obtains immediate profit).

Such promises are important in repeated interactions for a number of reasons. First, agents may be able to reach an agreement faster in the *present* game by providing some guarantees over the outcome of subsequent games. Thus, agents may find the current offer and the reward worth more than counter-offering (which only delays the agreement and future games). Second, by involving issues from future nego-

[1]We focus on rewards because of their clear impact on agreements in the context we consider and because we expect threats and appeals to follow similar principles to those we elucidate here.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

tiations in the *present* game (as in the cost of servicing in the example above), we effectively expand the negotiation space considered in the present game and, therefore, provide more possibilities for finding (better) agreements [4]. For example, agents that value future outcomes more than their opponent (because of their lower discount factors) are able to obtain a higher utility in future games, while the opponent who values immediate rewards can take them more quickly. Thirdly, if guarantees are given on the *next game*, the corresponding negotiation space is constrained by the reward, which should reduce the number of offers exchanged to search the space and hence the time elapsed before an agreement is reached. Continuing the above example, the buyer starts off with an advantage next time she wants to negotiate the price to service her car and she may then not need to negotiate for long to get a reasonable agreement.

Given this, this work advances the state of the art in the following ways. First, we develop a Reward Generation Algorithm (RGA) that calculates constraints (which act as rewards) on resources that are to be negotiated in future games. The RGA thus provides the first heuristics to compute and select rewards to be given and asked for in our new extension of Rubinstein's protocol. Second, we develop a specific Reward Based Tactic (RBT) for persuasive negotiation that uses the RGA to generate combinations of offers and rewards. In so doing, we provide the first persuasive negotiation tactic that considers the current negotiation game as well as future ones, to generate offers and arguments and thus reach better agreements faster than standard tactics in the long run.

The rest of the paper is structured as follows. Section 2 provides the basic definitions of the negotiation games we consider, while section 3 describes how persuasive negotiation can be used in such games. Section 4 details RGA and section 5 shows how offers and promises are evaluated. Section 6 describes RBT and section 7 evaluates its effectiveness. Finally, section 8 concludes.

2. REPEATED NEGOTIATION GAMES

Let Ag be the set of agents and X be the set of negotiable issues. Agents negotiate about issues $x_1, \cdots, x_n \in X$ where each one has a value in its domain $D_1, \cdots, D_n$. Then, a contract $O \in \mathcal{O}$ is a set of issue-value pairs, noted as $O = \{(x_1 = v_1), \cdots, (x_m = v_m)\}$.[2] We will also note the set of issues involved in a contract O as $X(O) \subseteq X$. Agents can limit the range of values they can accept for each issue, termed its negotiation range and noted as $[v^{min}, v^{max}]$. Each agent has a (privately known) utility function over each issue $U_x : D_x \rightarrow [0, 1]$ and the utility over a contract $U : \mathcal{O} \rightarrow [0, 1]$ is defined as $U(O) = \sum_{(x_i=v_i)\in O} w_i \cdot U_x(v_i)$, where w_i is the weight given to issue x_i and $\sum w_i = 1$. We consider two agents $\alpha, \beta \in Ag$ having utility functions designed as per the Multi-Move Prisoners' Dilemma (MMPD) (this game is chosen because of its canonical and ubiquitous nature) [13]. According to this game, α's marginal utility δU is higher than β's for some issues, which we note as O^α, and less for others, noted as O^β, where $O^\alpha \cup O^\beta = O$.

While it is possible to apply rewards to infinitely or finitely repeated games, we focus on the base case of one repetition in this work because it is simpler to analyse and we aim to understand at a foundational level the impact that such promises may have on such encounters. These games are played in sequence and there may be a delay θ between the end of the first game and the beginning of the second one. In a game, one agent (α or β) starts by making an offer $O \in \mathcal{O}$ and the opponent may then counter-offer or accept. The agents may then go on counter-offering until an agreement is reached or one of the agents' deadlines (t^α_{dead} or t^β_{dead}) is reached. If no agrement is reached before the deadline, the agents obtain zero utility (in either the first or second game). We also constrain the games, and further differentiate them from the case where agents play one game each time independently of the previous one, by allowing the second game to happen *if and only if* the first game has a successful outcome (i.e. an agreement is reached within the agents' deadlines and the contract is executed). In so doing, there is no possibility for agents to settle both outcomes in one negotiation round. The agents may also come to an agreement in the first game but fail to reach one in the second one, in which case the agents only obtain utility from the outcome of the first game.

If an agreement is reached, the agents are committed to enacting the deal settled on. This deal comes from the set of possible contracts which, in the first game, is captured by $\mathcal{O}_1$ and, in the second one, by $\mathcal{O}_2$. During these games, as time passes, the value of the outcome decreases for each agent according to their discount factor (noted as ϵ_α for agent α). This factor denotes how much the resources being negotiated decrease in usefulness over time.'The time between each illocution trasmitted is noted as τ. Then, the discount due to time is calculated as $e^{-\epsilon(\theta+t)}$ between the two games and $e^{-\epsilon(\tau+t)}$ between offers [7] where t is the time since the negotiation started (note that we expect $\theta >> \tau$ generally). The value of ϵ scales the impact of these delays, where a higher value means a more significant discounting of an offer and a lower value means a lower discounting effect. Each agent is also assumed to have a *target utility* to achieve over the two games (noted as $L \in [0, 2]$). This target can be regarded as the agent's aspiration level for the combined outcomes of the two games [3]. This target must, therefore, be less than or equal to the sum of the maximum achievable utility over the two games (2 in the case an agent has a $\epsilon = 0$ and exploits both games completely); that is $L \leq 1 + e^{-\epsilon(\theta+t)}$, where 1 is the maximum achievable utility in an undiscounted game.

Agents use the illocutions $propose(\alpha, \beta, O)$ and $accept(\alpha, \beta, O)$ to make and accept offers respectively. Additionally, they may use persuasive illocutions such as $reward(\alpha, \beta, O_1, O_2)$ and $askreward(\alpha, \beta, O_1, O_2)$. The former means α makes an offer O_1 and promises to give reward O_2. The latter that α asks β for a promise to give reward O_2 (we detail the contents of O_2 in the next section). Hence, while the promise of a reward aims to entice an opponent to accept a low utility contract in the current encounter, asking for a reward allows an agent to claim more in future negotiations (in return for concessions in the current one).

3. APPLYING PERSUASIVE NEGOTIATION

In persuasive negotiation, agents either promise to give rewards to get their opponent to accept a particular offer or ask for such promises in order to accept an offer. In our case, rewards are specified in the second game in terms of a range of values for each issue. Thus, giving a reward equates to specifying a range such as $v_x > k$ (where $k \in D_x$ is a constant) for issue x in $O_2 \in \mathcal{O}_2$ to an agent whose utility increases for increasing values of x. Conversely, asking for a reward means specifying $v_x < k$ in O_2 for the asking agent (whose utility increases for decreasing values of x). Now,

[2] Other operators $\geq, \leq$ could also be used.

agents may find it advantageous to accept such rewards if it costs them more to counter-offer (due to their discount factor) or if they risk passing their deadline (or their opponent's). Here, we do not deal with the issues related to whether the agents keep to their promises or how to tackle the uncertainty underlying this (we simply assume they do), but rather we focus on the reasoning mechanism that the agents require in order to negotiate using rewards.

Specifically, we propose that agents use the level to which they concede in the first game in order to decide on what to offer or ask for as a reward in the second one. This is graphically illustrated in figure 1 where $O_1 \in \mathcal{O}_1$ and $O_2 \in \mathcal{O}_2$ are the proposed offer and reward respectively.

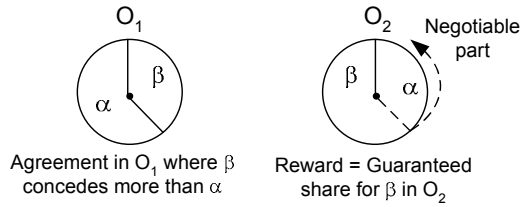


Figure 1: Representation of an offer O_1 made in the first game and a reward O_2 for β in the second game.

As can be seen, α exploits β (by taking a larger share of the pie) through the offer O_1 or alternatively β concedes more than α (depending on who makes the offer). The promised reward offered or asked for in the second game then tries to compensate for the exploitation/concession applied to the first game. Now, one strategy that produces this behaviour is the following: the higher the concession, the higher will be the reward demanded, while the lower the concession, the higher will be the reward given.[3] This strategy can be seen as a type of trade-off mechanism whereby agents take gains in the present (or the future) in return for losses in the future (or in the present) [10].

4. REWARD GENERATION

Building on the reasoning mechanism presented in section 3, we now develop our reward generation algorithm (RGA) that determines the level of concession made in the first game and hence determines the value of the corresponding reward, and finally decides whether to send it or not. First, we assume that an agent has some means of generating offers $\mathcal{O}_1$. In line with most work on negotiation in the presence of deadlines, we assume the agent's negotiation tactic concedes to some extent until an agreement is reached or the deadline is passed [2]. Then, at each step of the negotiation, based on the concessions made in an offer $O_1 \in \mathcal{O}_1$, RGA computes the reward $O_2 \in \mathcal{O}_2$ and decides if it is to be asked for or given. In more detail, algorithm 1 outlines the main steps of RGA which are then detailed in the following subsections.

4.1 Step 1: Compute Concession Degrees

In this context, the degree to which an agent concedes in any game is equivalent to the value it loses on some issues to its opponent relative to what the opponent loses to it on other issues. Assuming $(x = v_x) \in O$ is the value of an issue x, and $[v_x^{max}, v_x^{min}]$ is its negotiation range, then we define $\mu_x = U_x(v_x)$, $\overline{U}_x = \max\{U_x(v_x^{min}), U_x(v_x^{max})\}$, and $\underline{U}_x = \min\{U_x(v_x^{max}), U_x(v_x^{min})\}$. From these, we can compute the maximum an agent could get as $\overline{U} = \sum_{x \in X(O)} w_x \overline{U}_x$, the minimum as $\underline{U} = \sum_{x \in X(O)} w_x \underline{U}_x$ and the actual utility as $\Upsilon = \sum_{x \in X(O)} w_x \mu_x$ where w_x is α's relative weight of issue x and $\sum w_x = 1$. These weights can be ascribed the same values given to the weight the issue has in the utility function and can be normalised for the number of issues considered here. Then, an agent α computes the concession degree on the offer O as:

$$con^{\alpha}(O) = \frac{\overline{U} - \Upsilon}{\overline{U} - \underline{U}} \quad (1)$$

It is then possible to calculate concessions on issues with higher and lower δU for α using $con^{\alpha}(O^{\alpha})$ and $con^{\alpha}(O^{\beta})$ respectively. Then, the complement of these functions (i.e. $1 - con^{\alpha}(O^{\alpha})$ and $1 - con^{\alpha}(O^{\beta})$) represents how much β concedes to α from α's perspective (or how much α exploits β).

Algorithm 1 Main steps of the RGA

Require: $O_1 \in \mathcal{O}_1, L$

1: Compute concessions in O_1^{α} and O_1^{β}. {Here the agent determines how much both agents concede on the issues for which they have a higher and lower δU than their opponent.}
2: Select $O_2 \in \mathcal{O}_2$ that matches the level of concession in O_1
3: Check whether the combination of O and O_2 satisfies L, adjust $[v^{min}, v^{max}]$ for second game according to values in O_2 and send offer and reward.

4.2 Step 2: Determine Rewards

To determine which agent concedes more in the game (given that they play a MMPD), α needs to compare its degree of concession on the issues with higher δU than β (i.e. O_1^{α}) and those with lower δU than β (i.e. O_1^{β}) (in a zero sum game this is calculated for all issues). To this end, we define three conditions which refer to the case where α concedes as much as β ($COOP$), concedes more to β ($CONC$), and concedes less than β ($EXPL$) respectively as follows:

- $COOP = true$ when $con^{\alpha}(O_1^{\alpha}) + con^{\alpha}(O_1^{\beta}) = 1$ (i.e. α has no grounds to give or ask for a reward).
- $CONC = true$ when $con^{\alpha}(O_1^{\alpha}) + con^{\alpha}(O_1^{\beta}) > 1$ (i.e. α can ask for a reward).
- $EXPL = true$ when $con^{\alpha}(O_1^{\alpha}) + con^{\alpha}(O_1^{\beta}) < 1$ (i.e. α should give a reward).

The above conditions capture the fact that an agent can only ask for a reward if it is conceding in the first game and can only give one if it is exploiting in the first game. It is possible to envisage variations on the above rules as agents may not always want to give a reward to their opponent if they are exploiting in the first game or they may want to ask for one even if they are not conceding. However, these behaviours could be modelled in more complex strategies (which we will consider in future work). But, in so doing, an agent may also risk a failed negotiation. Here, therefore, we focus on the basic rules that ensure agents try to maximise their chances of reaching a profitable outcome.

Now, having determined whether an argument is to be sent or not and whether a reward is to be asked for or given, we can determine the value of the reward. Given that an agent aims to achieve its target L, the value chosen for a reward will depend on L and on $(con^{\alpha}(O_1^{\alpha}), con^{\alpha}(O_1^{\beta}))$ (i.e. the

[3]It should be noted that while the figure pictures a zero-sum game, the applicability of rewards is not limited to such situations. Instead, they can be applied to more complex games such as the MMPD, for which we detail the procedure in the next section (and which we use in our experiments in section 7).

degrees of concession of the agent). We will consider each of these points in turn.

Given O_1, the first game standing offer, the minimum utility α needs to get in the second game is $l_2 = L - U(O_1)$. We then need to consider the following two cases (remember $e^{-\epsilon(\theta+t)}$ is the maximum that can be obtained in the second game with discounts). Firstly, if $l_2 \leq e^{-\epsilon(\theta+\tau+t)}$ it is still possible for α to reach its target in the second game (provided the agents reach an agreement in the first one) and, therefore, give (or ask for) rewards as well. The larger l_2 is, the less likely that rewards will be given (since less can be conceded in the second game and still achieve L). Secondly, if $l_2 > e^{-\epsilon(\theta+\tau+t)}$, it is not possible to give a reward, but an agent may well ask for one in an attempt to achieve a value as close as possible to l_2.

For now, assuming we know $l_2 \leq e^{-\epsilon(\theta+\tau+t)}$, it is possible to determine how much it is necessary to adjust the negotiation ranges for all or some issues in O_2 in order to achieve l_2. Specifically, the agent calculates the undiscounted minimum utility $\frac{l_2}{e^{-\epsilon(\theta+\tau+t)}}$ it needs to get in the second game. Then, it needs to decide how it is going to adjust the utility it needs on each issue, hence the equivalent bound ν_{out} for each issue, in order to achieve *at least* $\frac{l_2}{e^{-\epsilon(\theta+\tau+t)}}$. Here, we choose to distribute the utility to be obtained evenly on all issues.[4] Thus, the required outcome ν_{out} of an issue in the second game can be computed as $\nu_{out} = U_x^{-1}\left(\frac{l_2}{e^{-\epsilon(\theta+t)}}\right)$.

Having computed the constraint ν_{out}, the agent also needs to determine how much it should reward or ask for. To this end, the agent computes the contract $\bar{O}$ which satisfies the following properties:

$$con^{\alpha}(\bar{O}_2^{\alpha}) = 1 - con^{\alpha}(O_1^{\beta}) \text{ and } con^{\alpha}(\bar{O}_2^{\beta}) = 1 - con^{\alpha}(O_1^{\alpha})$$

This is equivalent to our heuristic described in section 3 where the level of concession or exploitation in the offer in the first game (i.e. here $O_1 = O_1^{\alpha} \cup O_1^{\beta}$) is mapped to the reward asked for or given in the second one (i.e. here $\bar{O}_2 = \bar{O}_2^{\alpha} \cup \bar{O}_2^{\beta}$). Here also we adopt the same approach as for ν_{out} and distribute the concessions evenly on all issues. Then, assuming linear utility functions and finite domains of values for the issues, the above procedure is equivalent to reflecting the level of concession on issues with higher δU by α onto those with higher δU for β. This is the same as inverting equation 1 given a known $\overline{U}$ and $\underline{U}$ (as defined in step 1), and finding v_x by assigning $\mu_x = \Upsilon$ and inverting μ_x for each issue (a procedure linear in time with respect to the number of issues considered). Let us assume that for an issue x this results in a **bound** ν_r (a maximum or minimum according to the type of argument to be sent). Thus, from $\bar{O}_2$, α obtains bounds for all issues in the rewards it can ask from or give to β. Given this, we will now consider whether to send a reward based on how ν_r and ν_{out} compare for an issue x.

4.3 Step 3: Sending Offers and Rewards

Assume that α prefers high values for x and β prefers low ones and that it has been determined that a reward should be offered (the procedure for asking for the reward is broadly similar and we will highlight differences where necessary).

[4] Other approaches may involve assigning a higher ν_{out} (hence a higher utility) on some issues which have a higher weight in the utility function. In so doing, ν_{out} may constrain the agent's negotiation ranges so much for such issues that the two agents' ranges may not overlap and hence result in no agreement may be possible. Our approach tries to reduce this risk.

Now, α can determine whether a reward will actually be given and what its value should be according to the following constraints:

1. $\nu_r \geq \nu_{out}$: α can promise a reward implying an *upper bound* ν_r on the second game implying that α will not ask for more than ν_r. This is because the target ν_{out} is less than ν_r and α can, therefore, negotiate with a revised upper bound of $v^{max'} = \nu_r$ and a lower bound of $v^{min'} = \nu_{out}$. When asking for a reward, α will ask for a *lower bound* ν_r (i.e. $v^{min'} = \nu_r$) and negotiate with the same upper bound $v^{max'}$ in order to achieve a utility that is well above its target.
2. $\nu_{out} > \nu_r$: α cannot achieve its target if it offers a reward commensurate with the amount it asks β to concede in the first game. In this case, α revises its negotiation ranges to $v^{min'} = \nu_{out}$ (with v^{max} remaining the same). In this case, the agent does not send a reward but simply *modifies its negotiation ranges*. Now, if it were supposed to ask for a reward, α cannot achieve its target with the deserved reward. However, it can still ask β for the reward ν_r (as a lower bound) *and* privately bound its future negotiation to $v^{min'} = \nu_{out}$ while keeping its upper bound at v^{max}. In so doing, it tries to gain as much utility as possible.

Now, coming back to the case where $l_2 > e^{-\epsilon(\theta+\tau+t)}$ (implying $\nu_{out} > \nu_r$ as well), the agent that intends to ask for a reward will not be able to constrain its negotiation range to achieve its target (as in point 2 above). In such cases, the negotiation range is not modified and the reward may still be asked for (if $CONC = true$).

Given the above final conditions, we can summarise the rules that dictate when particular illocutions are used and negotiation ranges adjusted, assuming an offer O_1 has been calculated and O_2 represents the associated reward as shown below:

Algorithm 2 Step 3 of RGA.

if $COOP$ or ($EXPL$ and $\nu_{out} > \nu_r$) for at least one $x \in X(O_2)$ **then**
  $propose(\alpha, \beta, O_1)$.
end if
if $CONC$ and $l_2 \leq e^{-\epsilon(\theta+\tau+t)}$ **then**
  $askreward(\alpha, \beta, O_1, O_2)$ and modify $[v^{min}, v^{max}]$ for second game.
end if
if $CONC$ and $l_2 > e^{-\epsilon(\theta+\tau+t)}$ **then**
  $askreward(\alpha, \beta, O_1, O_2)$.
end if
if $EXPL$ and $\nu_{out} \leq \nu_r, \forall x \in X(O_2)$ **then**
  $reward(\alpha, \beta, O_1, O_2)$ and modify $[v^{min}, v^{max}]$ for second game.
end if

With all this in place, the next section describes how the recipient of the above illocutions reasons about their contents.

5. EVALUATING OFFERS AND REWARDS

We now describe how an agent evaluates the offers and rewards it receives. Generally, when agents negotiate through Rubinstein's protocol, they accept an offer only when the next offer O_{next} they intend to put forward has a lower (additionally discounted due to time) utility than the offer O_{given} presented to them by their opponent. However, agents using persuasive negotiation also have to evaluate the incoming offer together with the reward they are being asked for or

are being promised. To address this, we follow a similar line of reasoning as above and evaluate a received offer and reward against the offer and reward the agent would have sent in the next negotiation step. From the previous section, we can generally infer that a reward will imply a value ν_r for a given issue which defines either a lower or an upper bound for that issue in the next negotiation game. Therefore, given this bound, the agent may infer that the outcome ev of any given issue will lie in $[v^{min'}, v^{max'}]$ which might be equivalent to or different from the agent's normal negotiation ranges $[v^{min'}, v^{max}]$ and may take into account the agent's target ν_{out} (given its target l_2) or the value ν_r itself (as discussed in the previous section).

Specifically, assume β is the agent that is the recipient of a reward (given or asked for) and that β prefers small values for the issue x being considered. Then, let β's negotiable range be $[v^{min}, v^{max}]$ for the issue x and β's target be l_2^{β} in the second game (which implies that it needs at least ν_{out} for the issue in the second game). Now, if β receives $reward(\alpha, \beta, O, O_a)$ (or $askreward(\alpha, \beta, O, O'_a)$) for the second game, O_a implies that ν_r^{α} is the upper bound proposed for each issue x in O_a (ν_r^{α} would be a lower bound in O'_a). In the meantime, β has calculated another offer O_{new} with a reward O_b in which a bound ν_r^{β} is to be given to each issue x in O_b. Then, for each issue x, β calculates the negotiable ranges given ν_r^{α} as $[v^{min}, \min\{\nu_r^{\alpha}, \nu_{out}\}]$ (or $[\nu_r^{\alpha}, \min\{\nu_{out}, v^{max}\}]$ if O'_a is asked for[5]) while it calculates $[\nu_r^{\beta}, \min\{\nu_{out}, v^{max}\}]$ given ν_r^{β}. We assume β can then calculate (e.g. by picking a value over a normal distribution defined in the negotiation range[6]) the expected outcome of each range as ev_x^{α} for $[v^{min}, \nu_r^{\alpha}]$ (or $[\nu_r^{\alpha}, \min\{\nu_{out}, v^{max}\}]$ in the case of O'_a) and ev_x^{β} for $[\nu_r^{\beta}, \min\{\nu_{out}^{\beta}, v^{max}\}]$ in the case of O_b. Given each of these expected outcomes for each issue, the overall expected outcomes, $EO_a \in \mathcal{O}_2$ and $EO_b \in \mathcal{O}_2$, of the second game can be calculated given a reward. Thus, EO_a is the expected outcome of the reward given by α and EO_b is that for the promise of β. Given that these outcomes have been calculated, the agent then decides to accept or counter offer using the rule below. This evaluates the offer generated against the offer received to decide whether to accept the offer and promise received or send a *reward* illocution (note the addition of discount factors to reflect the time till the next game and between illocutions, that is, sending the counter offer, receiving an accept, and sending the first offer in the second game):

---

if $U(O_{new}) \cdot e^{-\epsilon_\beta(\tau+t)} + (U(EO_b) \cdot e^{-\epsilon_\beta(\theta+\tau+t)} \leq U(O) \cdot e^{-\epsilon_\beta(2\tau+t)}) + (U(EO_a) \cdot e^{-\epsilon_\beta(\theta+3\tau+t)}$ **then**
  $accept(\beta, \alpha, O)$
else
  $reward(\beta, \alpha, O_{new}, O_b)$
end if

As can be seen above, if the sum of the utility of the offer and the expected utility of the promise is higher than the offer and reward to be proposed by β (discounted over time), α's proposition is accepted. Otherwise, β counteroffers with its promise. If instead, a reward O'_b were to be asked for by β along with an offer O_{new}, then β will apply a similar decision rule as above (where EO'_b is the expected outcome β calculates for the reward it asks from α) in which we simply replace EO_b with EO'_b. Finally, in the case where β has received a persuasive offer and can only reply with another offer without any argument, β calculates the expected outcome of the second game using only its altered negotiation range $[v^{min}, \min\{\nu_{out}, v^{max}\}]$ to elicit EO''_b (which we use to replace EO_b with in the rule above). Note that the second game is left more uncertain in the latter case since the negotiation range has not been tightened by any reward and so the agents may take more time to reach an agreement in the second game (as per section 1).

Having described our mechanism for sending and evaluating rewards and offers, we will now propose a novel tactic that uses it to perform persuasive negotiation.

6. THE REWARD BASED TACTIC

As described in section 4, RGA requires an offer to be generated by some negotiation tactic in order to generate the accompanying reward. In this vein, the most common such tactics can be classified as: (i) behaviour-based (BB) – using some form of tit-for tat or (ii) time-based – using Boulware (BW) (concedes little in the beginning before conceding significantly towards the deadline) or Conceder (CO) (starts by a high concession and then concedes little towards the deadline) [2].[7] Now, many of these tactics start from a high utility offer for the proponent (here α) and gradually concede to lower utility ones. In our model, this procedure automatically causes RGA to start by promising rewards and then gradually move towards asking for rewards.

To ground our work, we present a novel reward-based tactic (RBT) (based on Faratin's trade-off tactic [3]) that either asks for or gives a reward at any point in the negotiation in order to reach an agreement. To do so, however, the agent needs to know how to evaluate incoming offers and rewards and generate counter-offers accordingly. Given this, we will consider the three main cases in calculating the response to having received an offer and a proposed reward (see algorithm 3).

Case 1: *An offer and a reward have been received and it is possible to counter offer with a reward.*
In this case, α needs to calculate combinations of rewards and offers and choose the combination that it deems most appropriate to send to β. To calculate these combinations, α first needs to determine the overall utility each combination should have. To achieve this, we use a hill climbing method similar to Faratin et al.'s tactic. In this method, the agent tries to find an offer that it believes is most favourable to its opponent, while not necessarily conceding too much. In our case (particular for the MMPD), this procedure equates to the agent trying to gain more utility on the issues on which it has a higher δU and less on those for which it has a lower δU than β.[8] In so doing, the strategy can maximise joint gains in the repeated negotiation encounter.

Thus, the utility to be conceded in the next offer (or utility step), Su, is calculated according to the difference that exists between the agent's previous offer and the last one sent by

[5]This range assumes $\nu_{out} \geq \nu_r^{\alpha}$, but in cases where this is not true the reward proposed by α is automatically rejected.

[6]This is the technique we adopt here. However, other techniques such as fuzzy reasoning or learning mechanisms could also be used to get this value.

[7]Other negotiation tactics might also be resource-based or dependent on other factors. The tactics we select here have been chosen because they have been demonstrated to be relatively successful and are among the most common ones studied in the literature [10, 2].

[8]Note this is different from the point discussed in footnote 4 since here we do not constrain the negotiation ranges, but rather search for offers that may be profitable to both parties.

its opponent.

$$Su(O_1, O_2, O'_1, O'_2) = \begin{array}{l} U(O_1)e^{-t} + U(EO_2)e^{-(\theta+t)} \\ -U(O'_1)e^{-(\tau+t)} + U(EO'_2)e^{-(\theta+2\tau+t)} \end{array}$$

where O_1 and EO_2 are the previous offer and expected outcome in the second game from α's reward O_2 respectively and O'_1 and EO'_2 are the current offer and the expected outcome of β's argument O'_2, respectively. If α does not specify a reward O_2, EO_2 is calculated as per section 5 given the normal negotiation ranges. Similarly, EO'_2 is also calculated in the same way if β does not specify a reward with the previous offer.

Given the utility step Su, it is then possible to calculate the utility Nu of the combination of the next offer and reward using the following equation:

$$\begin{array}{rl} Nu = & U(O_1)e^{-(2\tau+t)} + U(EO_2)e^{-(\theta+3\tau+t)} \\ & -Su(O_1, O_2, O'_1, O'_2, f) \end{array} \quad (2)$$

The next step involves generating combinations of offers and rewards whose combined utility is as close as possible to Nu. To this end, we use an optimisation function $OptComb$: $[0,2] \times \mathcal{O}_1 \times \mathcal{O}_2 \times \mathcal{O}_1 \times \mathcal{O}_2 \rightarrow \mathcal{O}_1 \times \mathcal{O}_2$, based on linear programming, that calculates the reward and offer whose values are most favourable to β (but still profitable for α). $OptComb$ therefore runs through RGA to find the best possible rewards and the associated offers whose combined utility is less than or equal to Nu and that concede more on issues for which β has a higher marginal utility. RGA also informs RBT whether the reward is to be asked for or given and whether negotiation ranges need to be modified (as described in section 4). However, $OptComb$ can also *fail* to find an optimal output (as a result of the constraints being too strong (e.g. the target L being too high) or the optimizer not being able to find the solution in the specified number of steps) and in these cases, we resort to another procedure described next (i.e. Cases 2 and 3).

CASE 2: *OptComb fails and the last offers made involved rewards.*
The agent cannot find a combination of a proposal and a reward whose utility matches Nu. Therefore, it calculates an offer using the time-based heuristics presented earlier.[9]

CASE 3: *OptComb fails and the last offers made did not involve rewards.*
It is possible to continue the same step-wise search for an agreement as in case 1. Here, our tactic calculates the offer whose utility is as close as possible to Nu (without $U(EO'_2)$ or $U(EO_2)$). Moreover, the offer calculated is such that it is the one that is most similar to the offer by β. This is achieved by running an optimization function $OptProp : [0,2] \times \mathcal{O}_1 \times \mathcal{O}_1 \rightarrow \mathcal{O}_1$ that calculates an offer O_1 such that O_1 maximises the level of concession on issues with higher marginal utility for the opponent (as in case 1) while still achieving Nu. In case the issues being negotiated are qualitative in nature, the similarity based algorithm by [3] may be used.

[9]In this case, BB tactics would not be appropriate to generate an offer given previous offers by the opponent. This is because some offers have been proposed in combination with a reward such that the concessions in the offers may not be monotonic (an asked for reward may compensate for a concession in the offer or a concession in the given reward may be compensated for by the higher utility of the offer). The latter property is a requirement for BB (or even all hill-climbing tactics [3]) to work . Therefore, either BW or CO is used to generate the offer since these are independent of the previous offers made by the opponent.

Algorithm 3 The RBT algorithm.

Require: O_1, O_2, O'_1, O'_2
1: Use a mechanism to calculate EO_2, EO'_2 {α calculates the expected outcomes of the arguments as discussed in section 5.}
2: step = $Su(O_1, O_2, O'_1, O'_2)$ {calculate the utility concession.}
3: nu = $U(O_1)e^{-(2\tau+t)} + U(EO_2)e^{-(\theta+3\tau+t)}$ − step {calculate the utility of the combination of offer and reward to be generated.}
4: $(O''_1, O''_2) = OptComb(\text{nu}, O_1, O_2, O'_1, O'_2)$ s. t. $U(O''_1)e^{-(2\tau+t)} + U(EO''_2)e^{-(\theta+3\tau+t)} \leq$ nu {here the values in the combination are optimised to be more favourable to β and as close as possible to nu.}
5: **if** $OptComb$ succeeds **then** {**Case 1**}
6: send O''_1 and O''_2 {RGA decides whether the reward is asked from or given to β.}
7: **else if** $OptComb$ fails & (both or one of O_2 or O'_2 is not null) **then** {**Case 2**}
8: use BW or CO to generate O''_1
9: send offer O''_1 and modify $[v^{min}, v^{max}]$ to achieve L as in RGA.
10: **else if** $OptComb$ fails & (both O_2 and O'_2 are null) **then** {**Case 3**}
11: step' = $Su(O_1, null, O'_1, null)$ {calculate the step in utility.}
12: nu' =$U(O_1)e^{-(2\tau+t)}$ − step' {calculate the utility of the offer to be generated.}
13: O''_1 = $OptProp(\text{nu}', O_1, O'_1)$ s.t. $U(O''_1) \leq$ nu' {find the offer that is most favourable to β but as close as possible to nu'.}
14: send offer O''_1 and modify $[v^{min}, v^{max}]$ to achieve L as in RGA.
15: **end if**

We capture all the above three cases in algorithm 3. As can be seen, RBT only generates offers and rewards in the first game. In the second one, we use a time-based or behaviour-based heuristic to calculate offers. While it is certainly possible to generate offers using the optimisation function of RBT in the second game, we do not do so in order to focus our analysis on the effect the bounds imposed by rewards have on the outcome of the second game when agents use basic tactics.

7. EXPERIMENTAL EVALUATION

In this section, we describe a series of experiments that aim at evaluating the effectiveness and efficiency of our PN model in repeated interactions. To this end, we evaluate it against basic tactics using standard benchmark metrics. In the following sections, we first detail the experimental settings and then provide the results of these experiments.

7.1 Experimental Settings

Agents α and β negotiate over 4 issues $x_1, ..., x_4$ and their preferences are as per a MMPD. Thus, $\delta U^{\alpha}_x > \delta U^{\beta}_x$, where $x \in x_1, x_2$ such that x_1 and x_2 are more valued by α than β, while x_3 and x_4, are more valued by β than α (i.e. $\delta U^{\alpha}_y < \delta U^{\beta}_y$, where $y \in x_3, x_4$). t_{max} is set to 2 seconds which is equivalent to around 300 illocutions being exchanged between the two agents (in one game).[10] The agents' deadlines, t^{α}_{dead} and t^{β}_{dead}, are defined according to a uniform distribution between 0 and 2 seconds. The discount factors, ϵ_{α} and ϵ_{β}, are set to a value between 0 and 1 and are drawn from a uniform distribution. The targets of the agents L^{α} and L^{β} are drawn from a uniform distribution between 0 and 2. We set $\theta = 0.5$ and $\tau = 0.0001$ to simulate instantaneous replies and set the degree of intersection of the negotiation ranges

[10]Preliminary experiments with the negotiation tactics suggest that if the agents do not come to an agreement within this time period, they never achieve any agreement (even if the maximum negotiation time is extended).

to 0.8 (which means that $[v_\alpha^{min}, v_\beta^{max}]$ overlap $[v_\alpha^{min}, v_\alpha^{max}]$ and $[v_\beta^{min}, v_\beta^{max}]$ by 80%).

	Utility function and weight of each issue			
	U_{x_1}, w_{x_1}	U_{x_2}, w_{x_2}	U_{x_3}, w_{x_3}	U_{x_4}, w_{x_4}
α	$0.4x_1$, 0.5	$0.9x_2$, 0.2	$1 - 0.2x_1$, 0.2	$1 - 0.6x_2$, 0.1
β	$1 - 0.2x_1$, 0.4	$1 - 0.6x_2$, 0.1	$0.9x_2$, 0.3	$0.4x_1$, 0.2

Table 1: Utility functions and weights of issues for each agent.

We will further assume that the first offer an agent makes in any negotiation is selected at random from those that have the highest utility. Also, the agent that starts the negotiation is chosen at random. This reduces any possible first-mover advantage that one strategy may have over another (i.e. which loses less utility due to discount factors). Moreover, in order to calculate the expected outcome of the second game (as discussed in section 5), agents draw the outcome for each issue from a normal distribution with its mean centred in the middle of the agent's negotiation range for the second game with a variance equal to 0.5. Finally, in all our experiments we use ANOVA (ANalysis Of VAriance) to test for the statistical significance of the results obtained.

Given these game settings, we define the populations of negotiating agents in terms of the tactics they use. As discussed in section 6, a number of tactics are available in the literature for experimentation and we will use BB tactics, as well as BW and CO, to generate offers for the RGA algorithm. Moreover, we will compare the performance of these with RBT. The settings of the strategies (i.e. the combination of tactics for the two games) played by the agents is given in table 2. Here, the populations of standard non-persuasive agents (i.e. disconnected from RGA) using only BB, BW, or CO in both games are noted as NT (negotiation tactics), while those that are connected to RGA are noted as PNT (persuasive negotiation tactics). The population of agents using RBT is noted as RBT.

Game	Strategies		
	Non-Persuasive	Persuasive	
Type	NT	PNT	RBT
1	BB, BW, CO	PBB, PBW, PCO	RBT
2	BB, BW, CO	BB, BW, CO	ANY

Table 2: Settings for agents' tactics.

As can be seen from the above table, agents can use rewards in the first game and revert to standard tactics for the second one. For example, a PNT agent, using BW with RGA in the first game, uses BW in the second game. For RBT agents, we randomly select among the three standard tactics.

Given that persuasive strategies like PNT and RBT can constrain their rewards and negotiation ranges according to their target L (as shown in section 4.2), we also need to allow other non-persuasive tactics to constrain their ranges accordingly to ensure a fair comparison. Thus, we allow all tactics to constrain the ranges of the issues in the second game according to their target whenever they reach agreements without the use of any arguments. The procedure to do so is similar to that described in section 4.2.[11] In the following experiments, we use homogeneous populations of 80 agents for each of NT, PNT, and RBT and also create a population of equal numbers of RBT and PNT agents (40 each) which we refer to as PNT&RBT to study how RBT and PNT agents perform against each other.

[11]The difference between the constraint applied by the reward and by the target is that the former applies the constraint to both agents, while the latter only applies separately to each agent according to their individual targets.

Given the populations of agents described above we next define the means used to decide whether PN indeed achieves *better* agreements *faster* than standard negotiation mechanisms. We therefore apply the following metrics:

1. Average number of offers — the average number of offers that agents need to exchange before coming to an agreement. The smaller this number the less time the agents take to reach an agreement.
2. Success rate — the ratio of agreements to the number of times agents meet to negotiate.
3. Average utility per agreement — the sum of utility of both negotiating agents over all agreements divided by the number of agreements reached.
4. Expected utility — the average utility weighted by the probability that an agreement is reached.

Given these requirements, in the following subsection we detail experiments with populations defined above and evaluate their performances.

7.2 Empirical Results

In this section, we postulate a number of hypotheses regarding the performance of RGA and RBT and describe the results which validate them.

H1 *Negotiation tactics that use RGA are more time efficient than those that do not.*

This hypothesis follows from the fact that we expect arguments to help agents find agreements faster. Here we record the average number of offers (the lower this number the more time efficient the agents are) an agent makes in order to reach an agreement. For all populations of tactics, each agent meets another agent 50 times and this is repeated 15 times and the results averaged. Thus it was found that NT takes an average of 547 offers to reach an agreement, while PNT strategies take 58 and PNT&RBT takes 56.5 offers per agreement (nearly 10 times less than NT). Thus, the performance of RBT is significantly better than the other populations since it reaches agreements within only 26 offers (which is less than NT by a factor of 21). Now, the reason for the superior performance of persuasive tactics in general is that the rewards make offers more attractive and, as we expected, the shrinkage of negotiation ranges in the second game (following from the application of the rewards) further reduces the negotiation space to be searched for an agreement. The additional improvement by RBT can be attributed to the fact that both negotiating agents calculate rewards and offers (through the hill-climbing algorithm) that give more utility to their opponent on issues for which they have a higher marginal utility (as explained in section 6). Hence, this is faster than for PNT&RBT in which only one party (the RBT) performs the hill-climbing.

These results suggest the outcomes of RBT and PNT populations should be less discounted and should also reach more agreements (since they take less time to reach an agreement and hence do not go over the agents' deadlines). However, it is not clear whether the utility of the agreements reached will be significantly higher than for NT agents.

H2 *Negotiation tactics that use the RGA achieve a higher success rate, expected utility, and average utility than those that do not.*

To test this hypothesis, we run the same experiments as above and record the average utility per agreement and the number of agreements reached. Thus, it is possible to calculate the expected utility, average utility per encounter, and the success rate per game as explained earlier.

It was found that the success rate of persuasive strategies is generally much higher than NT (0.87/encounter for NT, 0.99/encounter for PNT only, 1.0/encounter for PNT&RBT, and 1.0/encounter for RBT). This result[12] clearly shows that the use of RGA increases the probability of reaching an agreement. The similar performance of RBT and PNT&RBT and the difference between PNT&RBT and PNT shows that RBT agents, as well as being able to find agreements readily with their similar counterparts, are also able to persuade PNT agents with more attractive offers. This is confirmed by the fact that the average utility of persuasive strategies is generally higher[13] (i.e. 1.9/encounter for PNT, 1.95/encounter for PNT&RBT, and 2.03/encounter for RBT) than NT (i.e. 1.84/encounter). Note that the difference in utility between NT and other tactics would be much greater if discount factors ϵ_α and ϵ_β were bigger (given the high average number of offers NT uses (i.e. 547)).

Given the trends in success rate and average utility, the expected utility followed a similar trend with NT agents obtaining 1.6/encounter, PNT 1.88/encounter (i.e. a 17.5% improvement over NT), PNT&RBT 1.95/encounter, and 2.03/encounter for RBT agents only (representing a 26% better performance than NT). Generally speaking, from the above results, we can infer that RGA, used together with basic tactics, allows agents to reach better agreements much faster and more often.

These results also suggest that PNT agents reach broadly similar agreements (in terms of their utility) to NT agents (if we discount the fact that rewards significantly reduce the time to reach agreements and increase the probability of reaching an agreement). Now, as discussed in section 6, PNT agents usually generate offers first (starting from high utility ones as for the NT agents) and then calculate the rewards accordingly. Given this, the agents tend to start by giving rewards and end up asking for rewards. As the negotiation proceeds (if the offers are not accepted), the offers generally converge to a point where agents concede nearly equally on all issues (irrespective of the marginal utilities of the agents) and the rewards converge to a similar point. This, in turn, results in a lower overall utility over the two games than if each agent exploits the other one in each game in turn. Now, if rewards are selected in a more intelligent fashion, as in RBT, the agents reach much higher overall utility in general. This is because agents exploit each other more on the issues for which they have a higher marginal utility than their opponent. This is further demonstrated by the results of the RBT agents which suggest they reach agreements that have high utility for both participating agents. However, it is not apparent whether RBT agents are able to avoid being exploited by their PNT counterparts in such agreements which RBT tries to make more favourable to PNT agents (as described in section 6).

H3 *Agents using RBT are able to avoid exploitation by standard tactics connected to RGA (i.e. PNT).*

In order to determine which tactic is exploited, we recorded PNT's and RBT's average utility separately. Thus, it was found that on average, both RBT and PNT agents obtained about the same average utility per agreement (i.e. 0.96/agreement). This result[14] validates **H3** and suggests that the hill-climbing mechanism of RBT agents calculates offers that can convince the opponent without reducing the utility of both RBT and PNT agents significantly (i.e. in small steps) and also that it maximises joint gains through *OptComb*.

8. CONCLUSIONS

In this paper we introduced a novel persuasive negotiation protocol that allows agents in the present encounter to give and ask for rewards in future encounters. To complement this protocol, we also developed a reasoning mechanism that consists of a reward generation algorithm (RGA) and a reward based tactic (RBT). We then showed that RGA can improve the utility gain of standard negotiation tactics by up to 17%, and that RBT provides an additional utility gain of 26% while using 21 times fewer messages to reach a deal in our context.

Future work will look at extending RGA and RBT to more than two games and exploring other strategies to generate rewards, as well as other types of arguments such as threats and appeals. Furthermore, using trust models, we will develop techniques to deal with agents that may not fulfill their promises.

9. REFERENCES

[1] L. Amgoud and H. Prade. Formal handling of threats and rewards in a negotiation dialogue. In *Proceedings of the Fourth International Conference on Autonomous Agents and Multi-Agent Systems, Utrecht*, pages 529–536, 2005.

[2] P. Faratin, C. Sierra, and N. R. Jennings. Negotiation decision functions for autonomous agents. *International Journal of Robotics and Autonomous Systems*, 24(3-4):159–182, 1998.

[3] P. Faratin, C. Sierra, and N. R. Jennings. Using similarity criteria to make trade-offs in automated negotiations. *Artificial Intelligence*, 142(2):205–237, 2002.

[4] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: prospects, methods and challenges. *International Journal of Group Decision and Negotiation*, 10(2):199–215, 2001.

[5] S. Kraus, K. Sycara, and A. Evenchik. Reaching agreements through argumentation: A logical model and implementation. *Artificial Intelligence*, 104(1–2):1–69, 1998.

[6] P. McBurney, R. M. van Eijk, S. Parsons, and L. Amgoud. A dialogue-game protocol for agent purchase negotiations. *Journal of Autonomous Agents and Multi-Agent Systems*, 7(3):235–273, 2003.

[7] A Muthoo. *Bargaining Theory with Applications*. Cambridge University Press, 1999.

[8] S. Parsons, C. Sierra, and N. R. Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1998.

[9] I Rahwan, S D Ramchurn, N R Jennings, P McBurney, S D Parsons, and L Sonenberg. Argumentation-based negotiation. *The Knowledge Engineering Review*, 18(4):343–375, 2003.

[10] H. Raiffa. *The Art and Science of Negotiation*. Belknapp, 1982.

[11] S. D. Ramchurn, N. R. Jennings, and C. Sierra. Persuasive negotiation for autonomous agents: A rhetorical approach. In C. Reed, editor, *Workshop on the Computational Models of Natural Argument, IJCAI*, pages 9–18, 2003.

[12] A. Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50:97–109, 1982.

[13] G. Tsebelis. Are sanctions effective? A game theoretic analysis. *Journal of Conflict Resolution*, 34:3–28, 1990.

[12] Using ANOVA, it was found that for a sample size of 15 for each population of PNT, PNT and RBT, and PNT only, with $\alpha = 0.05$, $F = 8.8 > F_{crit} = 3.15$ and $p = 4.41 \times 10^{-4}$. These results prove that there is a significant difference between the means of PNT and the other strategies. The success rate of NT agents was found to be always lower than the other populations.

[13] These results were validated statistically using ANOVA, where it was found that $F = 3971 > F_{crit} = 2.73$, and $p = 7.36 \times 10^{-80}$, for a sample size of 15 per population and $\alpha = 0.05$. These results imply that there is a significant difference between the means of the populations.

[14] We validated this result using ANOVA with a sample of size 15 per strategy and $\alpha = 0.05$. Thus it was found that the null hypothesis (i.e. equal means for the two samples) was validated with $F0.13 < F_{crit} = 4.10$ and $p = 0.71 > 0.05$.

Negotiation Coalitions in Group-Choice Multi-Agent Systems

Tom Wanyama,
University of Calgary
Department of Electrical and Computer Engineering
2500 University Drive, N.W.Calgary, AB. CA.T2N 1N4.
Telephone: 1-403-247 0516
twanyama@ucalgary.ca

Behrouz H. Far
University of Calgary
Department of Electrical and Computer Engineering
2500 University Drive, N.W.Calgary, AB. CA. T2N 1N4.
Telephone: 1-403-210-5411
far@ucalgary.ca

ABSTRACT
In Group-Choice Decision Making (GCDM) where a number of stakeholders are involved in choosing a single solution from a set of available solution options, it is common for the stakeholders to form coalitions during negotiations in order to increase their individual welfare. It is also common to use Multi-Agent Systems (MAS) to automate GCDM processes. In such MAS, agents have to form coalitions like their human counterparts. Within each coalition, the individual agents behave according to the strategies of their clients. In this paper we present a coalition formation model which can employ multiple coalition formation algorithms. The model is motivated by a real-world group-choice multi-agent system that we developed; in the system, agents can form coalitions dynamically based on the similarity between their preferences, and based on the ownership criteria. Finally, this paper presents simulation results that illustrate the operational effectiveness of our coalition formation model.

Keywords
Coalition-formation, Negotiation, Group-Choice, Game-Theory, Qualitative-Reasoning.

1. INTRODUCTION
Coalition formation has the potential of increasing the welfare of negotiating entities. Therefore, the notion of forming coalitions is normally applied in various domains, including commerce, engineering and social planning. Furthermore, coalition formation in MAS has attracted a lot of attention because agents can easily be used to represent the negotiating entities, and they are capable of finding more profitable coalitions than humans in complex environments.

Although a variety of automatic coalition formation models have been proposed in literature, they all fall under the following two distinctive categories:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

1. Utility based models: These models (e.g. Game Theoretic Models) normally aim at identifying stable coalitions of negotiating agents, as well as determining appropriate models for sharing coalition benefits [1, 2, 3 and 8]. Most of these algorithms are naturally centralized; this is against the principle of decentralization which is fundamental to the concept of MAS. The computational requirements of utility based coalition formation algorithms usually make them impossible to implement [4]. Moreover, the algorithms normally assume that the agents surrender all their autonomy to their coalitions.
2. Knowledge based models: Knowledge based models for coalition formation [5, 6 and 7], depend on the skills of the individual agents to establish the most beneficial coalition structures. These models are normally computationally feasible, but they invariably have the following shortfalls:
   - They are usually feasible for small MAS.
   - They implicitly assume that there is no coalition structure that is more beneficial to all agents than the grand (coalition of all negotiating agents) coalition [4].
   - Like the utility based coalition formation algorithms, knowledge based algorithms normally assume total loyalty of agents to their coalitions.

This paper presents a coalition formation model that can employ multiple coalition formation algorithms. Besides increasing the utility of the negotiating agents and controlling the dynamics of the negotiation process, our coalition formation model has the following advantages:

- Our coalition formation model allows each agent to have different commitment levels for different coalitions to which it belongs.
- The model is not limited to any particular type of coalition structures.
- The model does not add any extra communication requirements to the MAS.

- The model is effective for both small and large scale MAS.

The rest of this paper is arranged as follows, In Section 2, we present our coalition formation model. Section 3 deals with simulation experiments that illustrate the capabilities of the coalition formation model. Finally, conclusions are given in Section 4.

2. COALITION FORMATION MODEL

Our coalition formation model works in the context of Group-Choice Multi-Criteria Decision Making (GC-MCDM). Each agent in MAS evaluates the available solution option individually based on the preference model of its client, using a Multi-Criteria Decision Making (MCDM) model. Thereafter, the agents select the solutions with the highest score as the offers to their negotiation opponents. At the end of every negotiation round, each agent adjusts its preference value function in a way so to increase the utility associated with the solution that the agent regards to be the "best-fit" for its coalition. Figure 1 shows that the main components of our coalition formation model are the following: the Algorithm Component, the Social Component, and the Commitment Component.

The Algorithm Component of our coalition formation model can have multiple coalition formation algorithms. In view of that, the GC-DSS we developed has two coalition formation algorithms. The first algorithm is based on agent ownership. This algorithm is very simple; all it requires is for the agents to have the ability to identify the owners of their negotiation opponents. The second algorithm is based on the similarity of preferences, which is determined using the *Qualitative Reasoning* model presented in Wanyama and Far [11].For the ownership algorithm, the input is the identification of the negotiation opponent and the offer that the opponent has made. On the other hand, the input for the similarity algorithm is the offer made by the negotiation opponent. The output of both algorithms is the preference model of the negotiation opponent (proposer).

The preference model of the proposer is sent to the Social Component, which uses a MCDM model to determine the estimated scores of all the solution options for the proposer. The estimated scores of the solution options for all coalition members which have already made their offers, together with the scores of the solution options for the concerned agents are fed into a *Game Theory* model to generate the *Social Fitness Factors* (SF_f) of the Solutions [9, 10]. These factors represent the tradeoffs associated with the solution options in respect of the coalition.

Besides generating the estimated preference value functions of the offer proposers, the Algorithm Component of our GC-DSS determines the *Tradeoff Factors* (T_f) of the solution options. These factors represent the amount of tradeoff associated with the solution options with respect to the concerned agent. The *Tradeoff Factors* and the *Social Fitness Factors* of each of the solution options are combined in the Commitment Component to generate the *Acceptance factors* (A_f) of the solution options [10]. The *Acceptance Factor* of a solution option is a measure of the ability of the solution to satisfy the combined preferences of the coalition members, from the perspective of a given coalition member. The generated Acceptance factors are used to update the offer acceptance/rejection criteria in the decision making component of the agent before responding to the current offer. Furthermore, the *Acceptance factors* associated with the last offer in each negotiation round are used in the modification of the *preference value function* of the concerned agent [11].

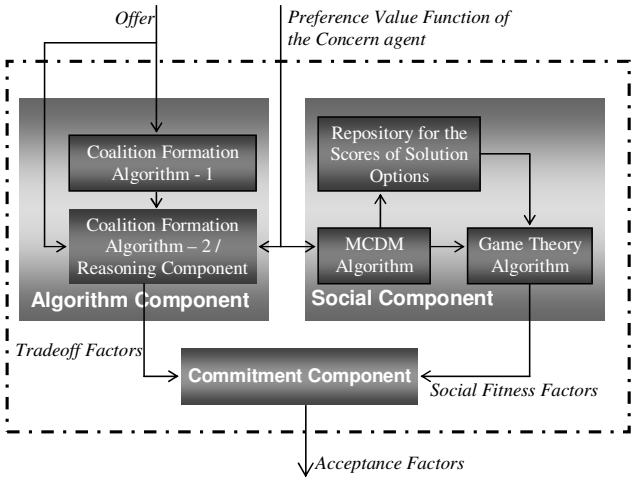


Figure 3. Coalition formation model

The proposed coalition formation model enables each agent to select individually its allies or coalition. Therefore, the model supports the formation of any coalition structure, and it does not require any extra communication or central coordination entity or agent. However, the model results in increased processing for each of the negotiating agents. The focus of this paper is the modeling of the coalition formation process, as well as the modeling of the commitment level of the agents to their coalitions. Therefore, in the following subsections, we briefly describe each of the three components on our coalition formation model.

3. EXAMPLE

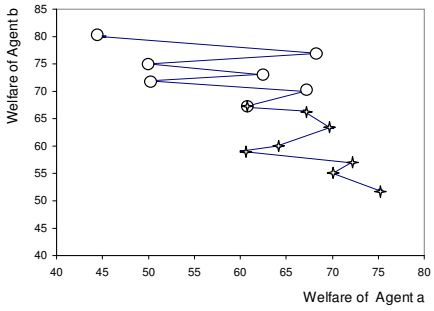
We adopted a group-choice decision making problem from Wanyama and Far [10]. In the problem, three agents were required to evaluate four COTS products based on the preference models of their individual clients, and then

negotiate to select the best-fit product. The following simulation experiments were carried out:

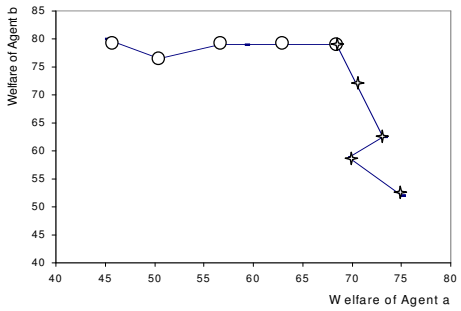
- Negotiation without coalition formation.
- Negotiation with a grand coalition by assuming that all the agents have the same owner.
- Negotiation with each of the three agents being allowed to form an alliance with any other agent based on the similarity of their preferences.

3.1 Results

The results of the first and last simulation experiments are presented in Figures 2(a) and 2(b).



5(a) No coalition formation



5(f) All the three agents are free to form alliances based on the similarity of their preferences

3.2 Discussion of Results

Generally, coalition formation creases the utility (payoff) of the coalition members. That is, the utility of the agreement offer for no coalition formation negotiation (Figure 2(a)) is smaller then the utility of agreement offers for any of the negotiations with coalition formation (Figure 2(b)). Furthermore, Figure 2 illustrates that coalition formation controls the dynamics of the negotiation process, and that coalition formation expedites the negotiation process (forming coalitions resulted in few negotiation rounds).

4. CONCLUSIONS AND FUTURE WORK

This paper presents a coalition formation model that is capable of employing multiple coalition formation algorithms. Currently, the model has two coalition formation algorithms, one based of the ownership of the agents, and the other based on the similarities between the preferences of the agents. Finally, the paper presents an example that illustrates the capabilities of our coalition formation model.

For the future, we would like to model coalition membership such that agents that belong to multiple coalitions can exhibit different levels of commitment to the different coalitions.

5. ACKNOWLEDGMENTS

The authors would like to thank the Canadian Department of Foreign Affairs and the International Institute for Agent-Based Software Systems Research (IIASR) for their financial support to this research

6. REFERENCES

[1] G. Vauvert and A. El Fallah-Segrouhni,, Coalition Formation among Strong Autonomous and Weak Rational Agents, *Proc. 10th European Workshop Modelling Autonomous Agents in a Multi-agent World* (MAAMAW), Springer-Verlag, Heidelberg, 2001.

[2] J.P. Kahan and A. Rapoport, *Theories of Coalition Formation*, Lawrence Erlbaum Associates, London, 1984.

[3] M. Klusch and A. Gerber, Dynamic Coalition Formation Among Rational Agents, *IEEE Intelligent Systems*, May/June 2002, pp. 42-47.

[4] N. Jennings and V. D. Dang, Coalition Formation Algorithms for Virtual Organisations, Intellingence, Agents, Multimedia Group, Electronics and Computer Sceince – University of Southampton, 2004, Available online from: http://www.iam.ecs.soton.ac.uk/projects

[5] R. T. Maheswaran and T. Basar, Coalition Formation in Proportionally fair Division Auctions, *The Autonomous Agents and Multi-Agent Systems Conference*, Melbourne Australia, July 2003, AAMAS 2003.

[6] S. Abdallah and V. Lesser, Organization-Based Coalition Formation, *The Autonomous Agents and Multi-Agent Systems Conference*, New York USA, July 2004. AAMAS 2004.

[7] S. Breban and J. Vassileva, A Coalition Formation Mechanism Based on Inter-Agent Trust Relationships, *The Autonomous Agents and Multi-Agent Systems Conference*, Bologna Italy, July 2002, AAMAS 2002.

[8] S. Kraus, "Negotiation and Cooperation in Multi-agent Environments, *J. Artificial Intelligence*, vol. 94, nos. 1–2, 1997, pp. 79–98.

[9] T. Wanyama and B. H. Far, A Multi-Agent Framework for Conflict Analysis and Negotiation: Case of COTS selection, *Transactions of the Institute of Electronics, Information and Communication Engineers: Special Issue on Software Agent and its Applications* – Vol. E88-D, No.9, September, 2005, pp. 2047-2058.

[10] T. Wanyama and B. H. Far, Multi-Agent System for Group-Choice Negotiation and Decision Support. In *the proceedings of the 3rd Workshop on Agent Oriented Information Systems,* New York USA, July 2004. AAMAS 2004, pp. 194-208.

[11] T. Wanyama and B. H. Far, Qualitative Reasoning Model for Tradeoff Analysis, *The Modeling Decisions for artificial Intelligence Conference*, Tsukuba Japan, July 2005, LNAI 3558, pp. 99-109.

Negotiation-based coalitions in the physical world

Majid Ali Khan and Ladislau Bölöni
School of Electrical Engineering and Computer Science
University of Central Florida
khan@bond.cs.ucf.edu, lboloni@cs.ucf.edu

ABSTRACT

We present a negotiation based coalition formation scheme for autonomous agents residing in the physical world. While in the virtual world we can abstract the negotiation into a calculation of rewards and ignore the details of the physical substrate of the negotiation, in the physical world we need to concern ourselves with variables such as the geographical location of the agent, range of the communication devices, path planning, obstacles and hazards, the necessity of co-location and the temporal constraints assocatiated with it. We concentrate on agents with coordinated location and speed (convoys) which can improve their performance by being able to traverse previously inaccessible areas. We assume a model where agents can dynamically join and leave the coalition.

1. INTRODUCTION

Coalition formation between autonomous agents is a relatively well researched topic. We find however, that the techniques developed for purely software agents acting in the virtual world are not immediately applicable to embodied agents acting in the physical world (or a simulation of it).

In this paper we present a series of algorithms for coalition formation of agents acting in the physical world. We will concentrate on a particular type of coalition we will call a *convoy*[1]: a set of agents with the same location and movement characteristics. The benefits of participating in a convoy is that convoys can traverse geographic areas inaccessible to individual agents. Agents have individual goals which are tied to geographic locations. Agents are selfish, the decision on joining or leaving a convoy is based exclusively of how the action furthers the goals of the agent. The agents and convoys are using negotiation mechanisms to control the joining and leaving of agents from the convoy, as well as the shared goals of the convoy (as reflected in its path and destination).

One example of such a scenario is the search and rescue missions in disaster zones (such as New Orleans after Hurricane Katrina, or the tsunami-affected Asian coastline). A variety of agents are performing a set of operations at distinct geographical locations. The communication services are distrupted, and there are a variety of dangerous areas which can be traversed only by special "convoys" - such as by boat, under the protection of armed guards, with special escorts or in the presence of translators.

[1]Note that the definition of a convoy does not necessarily imply a convoy of vehicles; it can be agents aboard a vehicle such as a boat, or multiple agents physically joined together.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. CONVOY FORMATION MECHANISM

The environment considered in this paper assumes a 2-dimensional geographic area, where we identify: *safe areas* which are traversable by any vehicle and convoy, *danger areas* which are traversable only by convoys and *inaccessible areas*.

In this environment, we consider the actions of a set of *embodied agents*, which have well-defined physical location and movement capabilities. In practice, these agents can be "RAP" (Robots, Agents and/or Persons). The goal of every agent is to execute a certain task at a destination location. For the purpose of this paper, we assume that these tasks are independent, and the collaboration between agents concerns only the movement towards the destination. We are currently working on negotiation models for the cases when the agents can collaborate in the execution of the tasks as well.

The agents try to reach their destination as soon as possible. The time to reach the destination can be improved by the formation of convoys. In certain cases, the agent can not reach the destination except through joining convoys. We assume the agents *self-interested* but *honest*; the agents keep their negotiated commitments. The agents can communicate wirelessly through message based communication, which can be either point to point or broadcasted to all other agents in the transmission range.

2.1 Convoy models

We define a *convoy* as a coalition of embodied agents which agreed on a common path. We allow a hierarchical structure of convoys to allow operations such as a convoy joining another convoy. For the sake of uniformity, we will consider that individual vehicles are being part of single-vehicle convoys.

A convoy C is described by a set of convoys $S = \{C_1, \ldots C_n\}$, a leader agent $A_L \in S$ and a set of commit-

ments $G = \{g_1, \ldots g_n\}$. The of commitments are usually expressed as constraints on the path of the convoy. The role of the leader is to negotiate on behalf of the convoy and to determine its path, taking into account the agreements G. The commitments of the convoys are related to passing through locations and can be classified as before (B) and after (A) commitments. A "before commitment" $B(L,t)$ commits the convoy to pass through location L *not later than* time t. An "after commitment" $A(L,t)$ commits the convoy to leave location L *not sooner than* time t (if the convoy reaches that location sooner, it can, of course wait in the location). We will call a commitment C_1 *stronger than* a commitment C_2 and denote it $C_2 \subset C_1$ if every set of actions which satisfies C_1 also satisfies C_2.

THEOREM 1.
$$\forall L, t_1 < t_2 \Rightarrow B(L,t_2) \subset B(L,t_1)$$
$$\forall L, t_1 < t_2 \Rightarrow A(L,t_1) \subset A(L,t_2)$$

We leave the proof of this theorem as an exercise to the reader.

2.2 Negotiation for an agent joining a convoy

Let us now consider the lifecycle of the embodied agents, and the objectives of the negotiation. Whenever an agent detects the presence of a convoy in its vicinity, it starts a negotiation process. If the negotiation is successful, the agent moves to join the convoy at a rendez-vous point. From then on, the agent moves with the convoy, until the pre-agreed leave point is reached. At the leave point the agent leaves the convoy, and moves independently to the destination. Even while the agent is in the convoy, it might start negotiations with other convoys, or simply consider to leave the convoy on its own. If the agent wants to leave the convoy before the leave point, the agent needs to negotiate for the leaving of the convoy with the convoy leader. If the negotiation is successful, the agent leaves the convoy, and it is free to follow its separate path to the destination, or join a different convoy.

We will denote with $\tau_C(L_1, L_2)$ the time it takes for convoy C to move from location L_1 to location L_2.

Let us consider the objective of the negotiation. In the simplest case, at time t an agent A with the destination D_A and corrent location L_A considers joining a convoy C, which has a current set of commitments G. The agent has its current expected arrival time $t_A = t + \tau_A(L_A, D_A)$. In the first approximation, the agent would join the convoy if it can add to its list of commitments an agreement $B(D_A, t'_A)$ with $t'_A < t_A$, that is, it can reach its final destination faster. However, even if this agreement is not feasible, it might be worth for the agent to join the convoy up to an intermediate location P, called the *leave point*. One sufficient condition for the agent to be worth joining the convoy until leave point L_{leave} is to have a commitment $B(L_{leave}, t_{leave})$ such that $t_{leave} + \tau_A(L_{leave}, L_A) < t_A$. This is however not a necessary condition; the agent might plan ahead for the joining a different convoy after leaving the current convoy at P.

A successful negotiation for an agent joining a convoy will add two commitments to the convoys' set of commitments: a commitment $A(L_{join}, t_{join})$ for the join location of the agent, and a commitment $B(L_{leave}, t_{leave})$ for the leave location of the agent.

Thus, the negotiation between the convoy and the agent is a multi-objective negotiation, with the $< L_{join}, L_{leave}, t_{join}, t_{leave} >$ quadruplet being the minimal negotiation set. In addition to these four objects, the negotiation might involve cost, penalties for contract breaking or other temporal constraints. In the following, we will briefly discuss the four negotiation objectives and their interrelationships.

L_{join} **the location to join the perspective convoy.** The interest of the agent is to negotiate a join location which is as close to its current location as possible, or to be in the general direction of the destination. The preferences of the convoy are (in order of preference) (a) to negotiate a join location L for which he already has an $A(L,t)$ commitment, (b) a location for which it has a $B(L,t)$ commitment, (c) a location which is on the current projected path of the convoy and (d) a location which is close to the current projected path of the convoy. Intuitively, (a) does not involve any new commitment for the convoy (if it manages to negotiate a join time earlier or the same as than the current commitment), (b) requires only a temporal commitment, without new restrictions on the path of the convoy. A location of type (c) restricts the ability of the convoy to change its path (although its current path remains ok), while a point of type (d) requires the convoy to change its path. These preferences will be inevitably reflected in the negotiation strategy of the convoy. One additional complexity is the number of negotiation choices. While the points of type (a) and (b) are coming from a limited set of discrete choices, the points (c) are coming from a one-dimensional while points (d) from a two dimensional continuum, limited only by the resolution of the raster maps on which the systems operate. This leads to an unrealistically large negotiation space. To reduce the negotiation space to a more realistic size, we chose to identify *segment locations* on the convoy path. These locations have special properties, such as they are situated on the intersection between safe and danger zones, or on the intersection between the convoys path and the agents path. In the first pass, we can eliminate the points which are not feasible because of one of the following reasons:

(i) The convoy has already passed that segment location. (ii) The vehicle can not reach the segment location before the convoy passes it. (iii) The segment location is unreachable by the vehicle (e.g. if it lies within a safe region surrounded by unreachable and danger zones).

t_{join} **- the joining time.** Once the agent and the convoy had identified its join location they need to negotiate the join time. In broad lines, the agent negotiates for the latest possible join time (to increase its safety margin in getting there), while the convoy for the earliest time (because that minimizes its commitment in waiting for the agent). The join time has to be at least the minimum time needed by the agent to reach the join point (the convoys minimal arrival time is not strictly relevant, as its commitment is to be arrive after the negotiated time). This is a simple linear negotiation, which (for all other negotiation objectives fixed) can be resolved with a monotonic concession protocol with Zeuthen strategy. However, we need to observe that once the hard requirement of $\tau(L_{current}, L_{join}) < t_{join} - t_{current}$ is met, the rest of the negotiation is only about safety margins. Thus an agent is more likely to concede in this parameter, which does not affect its predicted performance.

L_{leave} **- the leave location.** For this location, the interest of the agent is to negotiate a location as close as possible to its final destination (except the case when it is planning

to join another convoy on the leave location). The choices of the convoy are, in the order of preference (a) a location for which an existing B commitment exists, (b) a location for which an existing A commitment exists, (c) a point on the current planned path and (d) a point close to the current planned path. Note that the order of preferences for types (a) and (b) is reversed for this point compared to the join point. Similarly to the join location, for types (c) and (d) we consider segmentation approaches. Thus, the interest of the convoy might be to negotiate for leave points as far down as possible on its projected path. The ideal organization is a single leave point where all the participant agents depart for their individual destinations.

t_{leave} **- the leave time**. This parameter represents the guaranteed arrival time at the leave location. This parameter has a lower bound, limited by the physical time a convoy needs to reach the location on the optimal path, while still meeting its other commitments. The upper bound of this parameter is given by the limit at which it is not worth anymore for an agent to join the convoy $t_{leave}^{upper} = t_{current} + \tau_{agent}(L_{current}, L_{destination}) - \tau_{agent}(L_{leave}, L_{destination})$. Evidently, the interest of the agent is an earliest possible time - preferably the lower bound. The interest of the convoy is to minimize its commitment, by committing to as late time as possible. By accepting the lower bound, the convoy is essentially committing that it will not change its current path. This limits its ability to accommodate agents joining in the future.

2.3 Related Work

Coalition formation between objects residing in the physical world has been addressed by several researchers. Recently, [2] presents a coalition formation scheme in a Robocup team. [1] presents a scheme of operating large number of mobile robots using plan merging paradigm. Their scheme is based on local knowledge and incremental planning in a distributed manner. They attempt to resolve spatial movement conflicts between mobile robots using this approach. Although we have a similar problem domain, our effort differs in that 1) we use negotiations for coordinating the movement 2) our general goal has been to make mobile agents to agree on a meeting and leaving location rather than resolving the spatial movement conflicts.

Several research efforts like [9, 4] have used negotiation for forming teams of agents. Most of this work concerned agents residing in the virtual world. From the efforts which concern the physical world, we mention the DARPA Autonomous Negotiating Teams (ANTS) program. [8], used negotiation for resource allocation in multi sensor target tracking problem. A negotiation protocol for resource allocation problem in a multi-sensor based target tracking application is presented in [6]. [7] uses an auction based approach for task allocation for multiple robots exploring a map.

The negotiation process in our domain has to be time constrained. [5] is one of the classical papers on this topic. In our negotiation model, both parties lose if an agreement can not be reached in given time. So both parties are willing to accept any offer (even non-optimal) that can satisfy the join and leave constraints.

Part of our problem domain also resembles with multi agent meeting scheduling problem. [3] provides a good introduction to existing work in this domain. The focus in this domain has been mainly to make multiple agents agree on a given time slot for meeting, under static or dynamic user preferences. Our work differs in that our mobile agents schedule for spatial locations under dynamic temporal constraints.

3. CONCLUSIONS

In this paper we presented an approach for negotiation based coalition formation in the physical world. We find that many of the formal negotiation models are applicable, but the constraints of the physical world, such as temporal and spatial distances, communication constraints, as well as real-time operation requirements add new requirements. The model presented was simulated for convoy formation on real world maps. Future work include extending the coalition model to task oriented domains, development of better algorithms for real-time operation as well as implementation on physical embodied agents.

4. REFERENCES

[1] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and S. Qutub. Operating a large fleet of mobile robots using the plan-merging paradigm. In *IEEE International Conference on Robotics and Automation (ICRA'97)*, pages 2312–2317, 1997.

[2] J. Anderson, B. Tanner, and J. Baltes. Dynamic coalition formation in robotic soccer. In *Proceedings of the AAAI-04 Workshop on Forming and Maintaining Coalitions and Teams in Adaptive Multiagent Systems*, San Jose, CA, July 2004.

[3] E. Crawford and M. Veloso. Opportunities for learning in multi-agent meeting scheduling. In *In the Proceedings of the Fall AAAI Symposium: Artificial Multi-Agent Learning, 2004*, 2004.

[4] F. Dignum, B. Dunin-Keplicz, and R. Verbrugge. Agent theory for team formation by dialogue. In *ATAL '00: Proceedings of the 7th International Workshop on Intelligent Agents VII. Agent Theories Architectures and Languages*, pages 150–166, London, UK, 2001. Springer-Verlag.

[5] S. Kraus, J. Wilkenfeld, and G. Zlotkin. Multiagent negotiation under time constraints. *Artif. Intell.*, 75(2):297–345, 1995.

[6] R. Mailler, V. Lesser, and B. Horling. Cooperative negotiation for soft real-time distributed resource allocation. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 576–583, New York, NY, USA, 2003. ACM Press.

[7] S. Sariel and T. Balch. Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments. In *The Twentieth National Conference on Artificial Intelligence (AAAI), Integrating Planning into Scheduling Workshop, 2005*, 2005.

[8] L.-K. Soh and C. Tsatsoulis. Reflective negotiating agents for real-time multisensor target tracking. In *IJCAI*, pages 1121–1127, 2001.

[9] L.-K. Soh and C. Tsatsoulis. Allocation algorithms in dynamic negotiation-based coalition formation. In *IJCAAMAS Workshop on Teamwork and Coalition Formation 02,*, 2002.

A formal model for designing dialogue strategies

Leila Amgoud
IRIT - CNRS
118, route de Narbonne
31062 Toulouse Cedex 9, France
amgoud@irit.fr

Nabil Hameurlain
Université de Pau
Avenue de l'université
BP 1155 64012 Pau Cedex, France
nabil.hameurlain@univ-pau.fr

ABSTRACT

Modeling different types of dialogue between autonomous agents is becoming an important research issue. Several proposals exist with a clear definition of the dialogue protocol, which is the set of rules governing the high level behaviour of the dialogue. However, things seem different with the notion of strategy. There is no consensus on the definition of a strategy and on the parameters necessary for its definition. Consequently, there are no methodology and no formal models for strategies.

This paper argues that a strategy is a two steps *decision* process: i) to select the type of act to utter at a given step of a dialogue, and ii) to select the content which will accompany the act. The first step consists of selecting among all the acts allowed by the protocol, the best option which according to some *strategic beliefs* of the agent will at least satisfy the most important *strategic goals* of the agent. The second step consists of selecting among different alternatives (eg. different offers), the best one which, according to some *basic beliefs* of the agent, will satisfy the *functional goals* of the agent. The paper proposes then a formal model based on argumentation for computing on the basis of the above kinds of mental states, the best move (act + content) to play at a given step of the dialogue.

Categories and Subject Descriptors

I.2.3 [**Deduction and Theorem Proving**]: Nonmonotonic reasoning and belief revision
; I.2.11 [**Distributed Artificial Intelligence**]: Intelligent agents

General Terms

Human Factors, Theory

Keywords

Dialogue, Strategies

1. INTRODUCTION

In a multi-agent system, agents need the ability to engage in *dialogues* such as the ones identified in [4], namely persuasion and negotiation. Work in the literature has focused on defining formal models for dialogues. Generally, a dialogue system contains the following three components: the agents involved in the dialogue (i.e their *mental states* and reasoning capabilities), a *protocol*, and a set of *strategies*. The dialogue protocol is the set of *rules of encounter* governing the high-level behavior of interacting agents. A protocol defines: i) the set of permissible acts (eg. asking questions, making offers, etc.); ii) the legal replies for each act. Indeed, a protocol identifies the different possible replies after a given act. However, the exact act as well as a possible content to utter at a given step of the dialogue is a *strategy* matter. While the protocol is a public notion, strategy is crucially an individualistic matter.

In most works on modeling dialogues, the definition of a protocol poses no problems. However, the situation is different for dialogue strategies. There are very few works on strategies [3, 2]. There is no methodology and no formal models for defining them. There is even no consensus on the different ingredients involved when defining a strategy.

This paper argues that the strategy is a decision problem in which an agent tries to choose among different *alternatives* the best option, which according to its beliefs, will satisfy at least its most important goals. Two kinds of goals (resp. of beliefs) are distinguished: the *strategic* and the *functional* goals (resp. the *strategic* and *basic* beliefs). The strategic goals are the meta-level goals of the agent. Such goals will help an agent, on the basis of the strategic beliefs, to select the type of act to utter. Regarding functional goals, they will help an agent to select, on the basis of the basic beliefs to select the content of a move. We propose a formal model for defining strategies. The model takes as input two sets of goals: the strategic and the functional goals together with the strategic and basic beliefs and returns among the possible replies allowed by the protocol after a given act, the next move (act + its content) to play. The model is an extension of the argument-based decision framework proposed in [1]. The basic idea behind this model is to construct for each alternative the different arguments (reasons) supporting it, then to compare pairs of alternatives on the basis of the quality of their supporting arguments.

2. AGENT'S MENTAL STATES

Two kinds of goals are involved when selecting the next act to play and its content:

Strategic goals: For choosing the type of act to utter, an agent refers to what we call *strategic goals*. By strategic goals we mean the meta-level goals of the agent such as "minimizing the dialogue time", "selling at the end of the dialogue", etc. Suppose that at a given step of a negotiation dialogue, an agent has to choose between making an offer and asking a question. If the agent wants to minimize the dialogue time then it would choose to make an offer

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

instead of spending more time in questions. However, if the agent wants to get a maximum of information about the wishes of the other agent, then the agent would decide to ask a question. These goals are generally independent of the *subject* of the dialogue.

Functional goals: The goals of the agent which are directly related to the subject of the dialogue are called *functional goals*. They represent what an agent wants to achieve or to get regarding the subject of the dialogue. Let us take the example of an agent negotiation the place of a meeting. The agent may prefer a place which is not warm and not expensive. These functional goals are involved when selecting the content of a move. In a negotiation, an agent proposes offers that satisfy its goals.

As for goals, the beliefs involved in the two decision problems (selecting an act and a content) are also of different nature:

Strategic beliefs: which are the meta-level beliefs of the agent. They may represent the beliefs of the agent about the dialogue, about the other agents involved in the dialogue, and about ways for reaching the strategic goals. In negotiation dialogues where agents are trying to find a common agreement, agents may intend to simulate the reasoning of the other agents. Thus it is important for each agent to consider the beliefs that it has on the other agents'goals and beliefs.

Basic beliefs: represent the beliefs of the agent about the environment, the subject of the dialogue, and about ways for reaching the functional goals. Let us consider again the example of the agent negotiating the place of a meeting. Basic beliefs of the agent may include for instance the fact that "London is not warm", "Tunisia is hot", "London is very expensive", etc. This base may also contain some integrity constraints related to the dialogue subject such as the meeting cannot be at the same time in London and in Tunisia.

3. THE LOGICAL LANGUAGE

Let $\mathcal{L}$ be a propositional language and $Wff(\mathcal{L})$ the set of well-formed formulas built from $\mathcal{L}$. Each agent has the following bases:

$\mathcal{B}_b$ = $\{(k_p, \rho_p), p = 1, s_k\}$ where $k_p \in Wff(\mathcal{L})$, is the basic beliefs. The beliefs can be less or more certain. They are associated with certainty levels ρ_p. A pair (k_p, ρ_p) means that k_p is at least certain at a degree ρ_p.

$\mathcal{B}_s$ = $\{(l_j, \delta_j), j = 1, m\}$ where $l_j \in Wff(\mathcal{L})$ is the strategic beliefs base. Each of these beliefs has a certainty level δ_j.

$\mathcal{G}_s$ = $\{(g_q, \lambda_q), q = 1, t\}$ where $g_q \in Wff(\mathcal{L})$, is a base of strategic goals. The strategic goals can have different priority degrees, represented by λ_q. A pair (g_q, λ_q) means that that the goal g_q is important for the agent at least to a degree λ_q.

$\mathcal{G}_f$ = $\{(go_r, \gamma_r), r = 1, v\}$, where $go_r \in Wff(\mathcal{L})$, is the base of the functional goals of the agent. Each functional goal has a degree of importance denoted by γ_r.

The different certainty levels and priority degrees are assumed to belong to a unique linearly ordered scale T with maximal element denoted by 1 (corresponding to total certainty and full priority) and a minimal element denoted by 0 corresponding to the complete absence of certainty or priority.
We shall denote by $\mathcal{B}_b^*$, $\mathcal{B}_s^*$, $\mathcal{G}_s^*$ and $\mathcal{G}_f^*$ the corresponding sets of classical propositions when weights are ignored.

Let $\mathcal{S}$ be the set of speech acts allowed by the protocol. $\mathcal{S}$ may contain acts such as "'Offer' for making offers in negotiation dialogues, "Question" for asking questions, "Assert" for asserting information such as "the weather is beautiful", "Argue" for presenting arguments in persuasion dialogues, etc. The protocol precises for each act the possible replies to it. Let us suppose that the function `Replies` returns for each act, all the legal replies to it.

$$\texttt{Replies}: \mathcal{S} \longmapsto 2^{\mathcal{S}}$$

During a dialoge, agents exchange *moves* which are pairs: a *speech act* and its *content*. Formally:

DEFINITION 1 (MOVES). *A* move M *is a pair* (a, x), *where* $a \in \mathcal{S}$ *and* $x \in Wff(\mathcal{L})$.
$\texttt{Act}(M) = a$ *is a function which returns the speech act* a, *and* $\texttt{Content}(M) = x$ *is a function which returns the content* x.

The strategy problem is formalized as follows:

DEFINITION 2 (THE STRATEGY PROBLEM). *Let* (a, x) *be the current move in a dialogue. What is the next move* (a', x') *to utter such that* $a' \in \texttt{Replies}(a)$ *and* $x' \in Wff(\mathcal{L})$ *?*

To answer this question, one should find both a' and x'. Indeed, a' is the "best" element in $\texttt{Replies}(a)$ that satisfies $\mathcal{G}_s^*$ according to $\mathcal{B}_s^*$. This will be denoted by: $\mathcal{B}_s^*, a' \rightarrow \mathcal{G}_s^*$. Here by "best" we mean the act that satisfies as much important goals as possible.

Concerning x', this is also the "best" element among $X \subseteq Wff(\mathcal{L})$ that satisfies $\mathcal{G}_f^*$ according to $\mathcal{B}_b^*$. This will be denoted by: $\mathcal{B}_b^*, x' \rightarrow \mathcal{G}_f^*$. Here the set X is exactly the set of different alternatives concerning the content of a move. This set may contain different offers (eg. different town) if we have to choose the content of the act "offer", it may contain a set of formulas if we have to choose the content of the act "Assert", it may also contain a set of arguments if one has to select the content of the move "Argue", etc.

The solution to the strategy problem is the pair (a', x') such that $(\mathcal{B}_s^*, a' \rightarrow \mathcal{G}_s^*) \wedge (\mathcal{B}_b^*, x' \rightarrow \mathcal{G}_f^*)$.

4. THE ARGUMENTATION-BASED DECISION MODEL

Solving a decision problem amounts to defining a pre-ordering, usually a complete one, on a set $\mathcal{X}$ of possible choices (or alternatives), on the basis of the different consequences of each choice. In our case, the set $\mathcal{X}$ may be either the set $\texttt{Replies}(a)$ of the possible replies to a move, or the set $\texttt{Content}(a')$ with a' is the chosen act. Argumentation can provide such a pre-oredering. Recently, Amgoud [1] has proposed a formal framework for making decisions under uncertainty on the basis of arguments that can be built in favor of and against a possible choice. Such an approach has two obvious merits. First, decisions can be more easily explained. Moreover, argumentation-based decision is maybe closer to the way humans make decisions than approaches requiring explicit utility functions and uncertainty distributions. The basic idea behind an argumentation-based approach is to construct *arguments* in favor of each alternative from the different bases, to evaluate such arguments, and finally to apply some criterion or principle for comparing the alternatives on the basis of the arguments and their quality or strengths.
Let $\mathcal{B} = \{(b_i, \alpha_i), i = 1, \ldots, n\}$ be any *consistent* and *weighted* beliefs base, and $\mathcal{G} = \{(g_j, \beta_j), j = 1, \ldots, m\}$ be a *consistent* prioritized base of goals. $\mathcal{B}^*$ and $\mathcal{G}^*$ will denote the corresponding sets when weights are ignored. From these bases, different arguments in favor of elements of $\mathcal{X}$ can be built. The idea is that a choice is justified and supported if it leads to the satisfaction of at least one goal of the agent, taking into account its beliefs. Formally:

DEFINITION 3 (ARGUMENT). *An* argument in favor of *a choice* x *is a triple* $A = <S, g, x>$ *such that:*

- $S \subseteq \mathcal{B}^*$, $g \in \mathcal{G}^*$, *and* $x \in \mathcal{X}$
- $S \cup \{x\}$ *is consistent*
- $S \cup \{x\} \vdash g$
- S *is minimal (for set inclusion) among the sets satisfying the above conditions.*

S *is the* support *of the argument,* g *is the* goal *which is reached by the decision* x *and* x *is the* conclusion *of the argument. The set* $\mathcal{A}(\mathcal{B}, \mathcal{G}, \mathcal{X})$ *gathers all the arguments which can be constructed from* $\mathcal{B}$, $\mathcal{G}$ *and* $\mathcal{X}$. $\texttt{Arg}(x)$ *is the set of arguments whose conclusion is* x.

Since the bases $\mathcal{B}$ and $\mathcal{G}$ are weighted, arguments in favor of an alternative are more or less strong. Since an argument involves two kinds of information (beliefs and goals), its strength depends on the quality of both information. Thus, to each argument we associate a certainty level representing the degree of the less entrenched belief used in it, and a weight representing the importance of the goal reached by the conclusion of that argument.

DEFINITION 4 (STRENGTH OF AN ARGUMENT). *Let* $A = <S, g, x>$ *be an argument in* $\mathcal{A}(\mathcal{B}, \mathcal{G})$. *The* strength *of* A *is a pair* $<Level(A), Weight(A)>$ *such that:*

- $Level(A)$ = *min*$\{\rho_i \mid k_i \in S$ *and* $(k_i, \rho_i) \in \mathcal{B}\}$. *If* $S = \emptyset$ *then* $Level(A) = 1$. *It represents the* certainty level *of the argument.*
- $Weight(A) = \lambda$ *with* $(\lambda, g) \in \mathcal{G}$. *It represents the* degree of satisfaction *of the argument.*

The strengths of arguments make it possible to compare pairs of arguments in several ways, depending on whether the agent gives the same importance to the certainty level and the weight or not. The aim of this paper is not to give an exhaustive list of comparison criteria, but to show what are the main components that define a dialogue strategy. For the purpose of illustration, let us consider the following particular criterion in which goals take precedence over beliefs:

DEFINITION 5. *Let* A *and* B *be two arguments in* $\mathcal{A}(\mathcal{B}, \mathcal{G})$. A *is* preferred *to* B, *denoted* $A \succeq B$, *iff:*

- $Weight(A) > Weight(B)$, *or*
- $Weight(A) = Weight(B)$ *and* $Level(A) \geq Level(B)$

PROPERTY 1. *The relation* $\succeq$ *is a total preorder, i.e.* $\succeq$ *is reflexive and transitive.*

At the core of our framework is the use of a *principle* that allows for an argument-based comparison of alternatives. Each combination of a principle with a criterion for comparing arguments captures a particular agent *profile*, thus a particular strategy for selecting moves in a dialogue.

Below we present an example of a principle that takes as input the supporting arguments and prefers an alternative which has at least one supporting argument which is preferred to any supporting argument of the other alternative. Formally:

DEFINITION 6. *Let* $x, x' \in \mathcal{X}$. x *is preferred to* x', *denoted* $x \triangleright x'$, *iff* $\exists\ P \in \texttt{Arg}(x)$ *s.t* $\forall\ P' \in \texttt{Arg}(x')$, $P \succeq P'$.

PROPERTY 2. *The relation* $\triangleright$ *is a total preorder.*

DEFINITION 7 (BEST DECISIONS). *Let* $\mathcal{X}$ *be a set of possible decisions. The set of best decisions is* $\texttt{Best}(\mathcal{X}) = \{x \in X, s.t. \forall\ x' \in \mathcal{X}, x \triangleright x'\}$.

DEFINITION 8 (ARGUMENTATION-BASED DECISION FRAMEWORK). *An* argumentation-based decision framework *is a tuple* $<\mathcal{X}, \mathcal{A}(\mathcal{B}, \mathcal{G}, \mathcal{X}), \succeq, \triangleright\!\!>$ *where:*

- $\mathcal{X}$ *is a set of all possible decisions.*
- $\mathcal{A}(\mathcal{B}, \mathcal{G}, \mathcal{X})$ *is a set of arguments built from* $\mathcal{B}$, $\mathcal{G}$ *and* $\mathcal{X}$.
- $\succeq$ *is a (partial or complete) pre-ordering on* $\mathcal{A}(\mathcal{B}, \mathcal{G}, \mathcal{X})$.
- $\triangleright$ *defines a (partial or complete) pre-order on* $\mathcal{X}$.

5. COMPUTING THE NEXT MOVE IN A DIALOGUE

In the previous section, we have presented a formal framework for explaining, ordering and making decisions. In what follows, we will show how that framework can be used for move selection. Let (a, x) be the current move of the dialogue, and an agent has to choose the next one, say (a', x').
The act a' is returned as a best option by the framework $<\texttt{Replies}(a)$, $\mathcal{A}(\mathcal{B}_s, \mathcal{G}_s, \texttt{Replies}(a))$, $\succeq$, $\triangleright\!\!>$ (i.e $a' \in \texttt{Best}(\texttt{Replies}(a))$), whereas a content x' is among the best options returned by the framework $<\texttt{Content}(a')$, $\mathcal{A}(\mathcal{B}_f, \mathcal{G}_f, \texttt{Content}(a'))$, $\succeq$, $\triangleright\!\!>$, i.e. $x' \in \texttt{Best}(\texttt{Content}(a'))$.

6. CONCLUSION

This paper has proposed a formal framework for defining strategies. This framework can be regarded as two separate systems: one of them take as input the possible replies allowed by a protocol, a set of strategic beliefs and a set of strategic goals and returns the best speech act, and the second system takes as input a set of alternatives, a set of basic beliefs and a set of functional goals and returns the best content of a speech act. The two systems are grounded on argumentation theory. The basic idea behind each system is to construct the arguments in favour of and against each choice, to compute the strength of each argument and finally to compare pairs of choices on the basis of the quality of their supporting arguments. We have shown also the agents profiles play a key role in defining principles for comparing decisions. In this paper we have presented two examples: pessimistic agents which represent very cautious agents and optimistic agents which are adventurous ones.

7. REFERENCES

[1] L. Amgoud. A general argumentation framework for inference and decision making. In *Proceedings of the 21th Conference on Uncertainty in Artificial Intelligence*, pages 26–33, 2005.

[2] L. Amgoud and N. Maudet. Strategical considerations for argumentative agents. In *Proc. of the* 10th *International Workshop on Non-Monotonic Reasoning, session "Argument, Dialogue, Decision", NMR'2002*, 2002.

[3] L. Amgoud and K. Souhila. On the study of negotiation strategies. In *In AAMAS 2005 Workshop on Agent Communication (AC05),*, pages 3–16, 2005.

[4] D. N. Walton and E. C. W. Krabbe. *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. State University of New York Press, Albany, NY, 1995.

A Scalable Method for Online Learning of Non-linear Preferences Based on Anonymous Negotiation Data

D.J.A. Somefun
CWI, Center for Mathematics and Computer Science
P.O. Box 94079, 1090 GB Amsterdam,The Netherlands
koye@cwi.nl

J.A. La Poutré
CWI and Technical Univ. Eindhoven
School of Techn. Management
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
hlp@cwi.nl

ABSTRACT

We consider the problem of a shop agent negotiating *bilaterally* with *many* customers about a bundle of goods or services together with a price. To facilitate the shop agent's search for mutually beneficial alternative bundles, we develop a method for online learning customers' preferences, while respecting their privacy. By introducing extra parameters, we represent customers' highly nonlinear preferences as a linear model. We develop a method for learning the underlying stochastic process of these parameters online.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Economics, Algorithms

Keywords

Negotiation, Market-Based Methods, Online Learning

1. INTRODUCTION

We consider the problem of a shop agent negotiating *bilaterally* with *many* customers about a bundle (configuration)—an array of bits, representing the presence or absence of each of the shop's goods in the bundle—together with a price for the bundle. Like the work of [3, 2, 4, 1], the techniques developed in this paper try to benefit from the so-called win-win opportunities that arise with multi-issue negotiations.

The methods proposed in Faratin et al. [2], and Coehoorn and Jennings [1] are geared towards finding win-win opportunities through modeling the preferences of the negotiation partner, for issues with independent valuations. This paper follows a similar approach; like Klein et al. [3] and our earlier work [4], we consider *interdependencies* between issues, however, which make the problem considerably harder. The novelty of our current approach lies in the introduction of a new scalable and very fast method for *online* learning these complex customer preferences, while respecting their privacy. To respect customers' privacy, the method as developed only uses *anonymous* data on past negotiations: i.e., the shop cannot determine if he has dealt with a customer in the past. As the conducted computer experiments show, the necessary aggregate information, to significantly facilitate the search for mutually beneficial alternative bundles in future bilateral negotiations, is learned quickly for relatively large problems.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. NEGOTIATION PROCESS

The shop sells a total of n goods, each of which may be either absent or present in a bundle, so that there are $2^n - 1$ distinct bundle-configurations containing at least 1 good, with $n = 20$. A negotiation concerns a bundle, together with a price for that bundle, and it is conducted in an alternating exchange of offers and counter offers. An example of such a practice may involve the sales of bundles of news items in categories like politics, finance, economy, sports, arts, etc.

During the negotiation, the shop will suggest a *limited number* of alternative bundles. Hopefully, these suggestions will lead to the bargainers finding mutually beneficial alternative bundles. Such bundles represents *Pareto improvements* because switching makes at least one bargainer better off without making the other worse off. Ideally, the shop will eventually find a *Pareto efficient* bundle: a bundle for which no Pareto improvements exits.

Typically, a customer initiates the negotiation process by submitting an initial bundle and opening price. The shop agent constructs an ordered set from all the bundles that lie in the initial bundle's neighborhood (i.e., all bundles which in binary representation have a hamming distance of 1 to the initial bundle); it orders the bundles based on the likelihood that they will result in Pareto improvements, starting with the most likely bundle. By advising the first bundle in this set first, the second bundle second, and so on, the shop agent searches (locally) for Pareto improvements (cf. [4] for further details). To assess the likelihood of a Pareto improvement requires statistical knowledge of customers' preferences. The focus of this paper lies on how we can obtain this knowledge based on anonymous negotiation data.

3. PROBLEM STATEMENT

First we show how we can rewrite customers' highly non-linear preferences as a linear model. In the absence of strategic behavior and stochasticity in customers' preferences, this will give use a direct solution to the learning problem. In the next section, we use the obtained insights to develop a fast method for online learning more about customer preferences given strategic behavior and stochasticity.

There are n individual goods from which customers can compile the bundle they wish to purchase. The column vector $\mathbf{b} = [b(1), \ldots, b(n)]^T$ denotes the binary representation of a bundle (i.e., $b(i) = 1$ if and only if good i is part of the bundle). Let a_i denote a customer's valuation for obtaining good i individually and $a_{i,j}$, $a_{i,j,l}$, etc. denote the synergy effects exclusively caused by obtaining a subset of goods $\{i,j\}$, $\{i,j,l\}$, etc.: e.g., a_{ij} is equal to the customer's valuation for bundle $\{i,j\}$ minus the valuation for obtaining the goods individually. There are $\binom{n}{1}$ parameters of the type a_i, $\binom{n}{2}$ parameters of the type a_{ij}, and so on; thus there are at most $k = \sum_{i=1}^{n} \binom{n}{i} = 2^n - 1$ parameters. In the experiments, we consider problems where $n = 20$. To make the number of parameters manageable we assume that beyond subsets of size 3 the synergy effect are always zero: i.e., there are only $k = \binom{n}{1} + \binom{n}{2} + \binom{n}{3}$ number of parameters.

Given some (arbitrary but) fixed encoding, denoted by $<\cdot>$, we can map all k parameters into the (k-dimensional) vector $\mathbf{a}$. We can now define a customer's monetary valuation for bundle $\mathbf{b}$, $v^c_{\mathbf{a}}(\mathbf{b})$, as follows:

$$v^c_{\mathbf{a}}(\mathbf{b}) = \sum_{i=1}^{n} \mathbf{a}_{<i>} b(i) + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbf{a}_{<i,j>} b(i) b(j) + \ldots + \mathbf{a}_{<1,\ldots,n>} b(1) b(2) \ldots b(n). \quad (1)$$

From Eq. (1) (and encoding $<\cdot>$), we can conclude that the seller can construct a mapping $\mathbf{f} : \{0,1\}^n \mapsto \{0,1\}^k$. For all customers, their monetary valuation for any bundle $\mathbf{b}$ can be expressed as a linear function of their k-parameters and $\mathbf{f}(\mathbf{b})$. *For example*, if there are only 3 goods—good 1,2, and 3—then $\mathbf{b} = (1,1,0)$ could (depending on the encoding) give $\mathbf{f}(\mathbf{b}) = (1,1,0,1,0,0,0)$. Irrespective of the encoding, $\mathbf{f}(1,1,0)$ should contain three 1's and four 0's: i.e., the parameters $a_{<1>}$, $a_{<2>}$, and $a_{<1,2>}$ contribute to the valuation of bundle $(1,1,0)$.

In the absence of strategic behavior and stochasticity, $\mathbf{a}$ is the same for all customers. The seller can then just obtain $\mathbf{a}$ by solving a system of linear equations. For example, suppose there are t observations and for each $i = 1, \ldots, t$ the seller has the observations v_i and $\mathbf{b}_i$, where v_i denotes customers' valuations for bundle $\mathbf{b}_i$. Then

$$\mathbf{v} = \mathbf{B}\mathbf{a} \quad (2)$$

with $\mathbf{v} = [v_1, \ldots, v_t]^T$ and $\mathbf{B} = [\mathbf{f}(\mathbf{b}_1), \ldots, \mathbf{f}(\mathbf{b}_t)]^T$ (thus $\mathbf{B}$ is a $t \times k$ matrix). Since the seller also knows $\mathbf{f}$, he knows both $\mathbf{v}$ and $\mathbf{B}$. Now if the rank of matrix $\mathbf{B}$ is equal to k, then the (Moore-Penrose) generalized matrix inverse of $\mathbf{B}$, denoted by $\mathbf{B}^+$, can be used to compute $\mathbf{a}$: i.e., $\mathbf{a} = \mathbf{B}^+\mathbf{v}$.

Typically, customer preferences differ from one customer to the next: i.e., $\mathbf{a}$, which completely describes customers' preferences, is a random vector. We assume $\mathbf{a}$ is normally distributed with a vector of means $\mu_{\mathbf{a}}$ and covariance matrix $\mathbf{\Sigma}_{\mathbf{a}}$. Moreover, since customers will behave strategically the seller will not directly observe their valuations.

4. LEARNING METHOD

The method we develop for estimating the vector of means $\mu_{\mathbf{a}}$ and covariance matrix $\mathbf{\Sigma}_{\mathbf{a}}$, contains (a) a sub method for learning $\mu_{\mathbf{a}}$ and $\mathbf{\Sigma}_{\mathbf{a}}$ in the absence of strategic behavior and (b) a sub method for preprocessing the negotiation data. First method (b) is applied to the data such that a sufficient amount of the noise caused by strategic behavior is removed.

(*Sub Method A*) As was discussed in Section 3, per customer we have an instance of the random vector $\mathbf{a}$, which completely specifies her preferences. For a customer c, let $\mathbf{a}_c$ denote such an instance. Suppose the seller would negotiate for a very long time with this customer c, while constantly suggesting different bundles. In the absence of strategic behavior, eventually the seller would have collected the negotiation data $\mathbf{v}_c = [v_{c1}, \ldots, v_{ct}]^T$, and $\mathbf{B}_c = [\mathbf{f}(\mathbf{b}_{c1}), \ldots, \mathbf{f}(\mathbf{b}_{ct})]^T$ such that $\mathbf{B}_c$ has rank k. Analogous to Eq. (2), the seller can then just obtain $\mathbf{a}_c$ by solving $\mathbf{v}_c = \mathbf{B}_c\mathbf{a}_c$: i.e., $\mathbf{a_c} = \mathbf{B}_c^+\mathbf{v}_c$, where $\mathbf{B}_c^+$ is again the generalized matrix inverse of $\mathbf{B}_c$.

For large k, bargainers will break off the negotiations or reach a deal, before the matrix $\mathbf{B}_c$ grows into a rank k matrix. Therefore, the seller suggests instead a sequence of bundles such that the union of the goods contained in these bundles has strictly less goods than is possible. (In the experiments, 4, 5, or at most 6 individual goods are being considered.) In that case, relative to k the matrix $\mathbf{B}_c$ contains only a few nonzero columns. Whenever k', the rank of the matrix $\mathbf{B}_c$, equals the number of nonzero columns, we can obtain $\mathbf{a}'_c$—the k'-dimensional vector containing all the identifiable parameters of $\mathbf{a}_c$—by solving $\mathbf{v}_c = \mathbf{B}'_c\mathbf{a}'_c$ (i.e., $\mathbf{a}'_c = \mathbf{B}'^{+}_c\mathbf{v}_c$); we get the matrix $\mathbf{B}'_c$ by removing all zero columns from $\mathbf{B}_c$. Based on these newly computed parameter values, it is now relatively straightforward to obtain estimations of the mean, and covariance matrix.

(*Sub Method b*) With strategic behavior the seller does not observe $\mathbf{v}_c = [v_{c1}, \ldots, v_{ct}]^T$. Instead he observes $\mathbf{p}_c = [p_{c1}, \ldots, p_{ct}]$, the sequence of bids submitted by the customers, where there is generally a difference between p_{ci} and v_{ci}. To deal with strategic behavior requires some implicit or explicit model of customers' bidding strategies. For example, in [4, 1] they learn more about customer preferences based on the difference between two consecutive bids. This implies some implicit model of a customer's bidding behavior, however. We want to make this model explicit so that we can check whether the model is consistent with a customer's bidding behavior.

Strategic behavior of customers can roughly be split into a fixed and variable component. The fixed component specifies how much a customer c is at most willing to bid for a bundle $\mathbf{b}$ worth $\mathbf{f}(\mathbf{b})^T\mathbf{a}_c$, irrespective of the current state of the negotiation; the variable component specifies how a customer concedes through time.

Filtering out the fixed component of a customer's bargaining strategy is impossible. Especially in the case of bundling, the seller can eradicate most of the second type of noise, however. The idea is that at the end of a negotiation process the seller will use some of the gathered negotiation data to model how the opponent concedes trough time; the rest of the data can then be used to check whether the developed model correctly predicts how a customer concedes through time. It is very unlikely that such a test is passed incorrectly (error of the first kind) because both the correctness of the estimated values for the identifiable parameters ($\mathbf{a}'_c$) and the concession model are tested jointly. The other pos-

sibility, of not being able to model a negotiation process is even less problematic; the seller can just simple ignore this negotiation. In real world applications the seller could have a collection of typical models and check which of the models fits the data of a particular negotiation best. Consequently, enough negotiation data can be preprocessed and fed to sub method (a) to ensure good estimations of $\mu_{\mathbf{a}}$ and $\mathbf{\Sigma_a}$.

5. NUMERICAL EXPERIMENTS

Customers are at most willing to pay x% of their true valuation; for each customer x is randomly drawn between 85 and 95. Moreover, customers and shop use time dependent bidding strategies: the offer submitted by a customer (shop) is monotonically increasing (decreasing) in both the number of bidding rounds and the valuation.

We compare our new approach of having *no a priori information* and learning customer preferences online (as discussed in Section 4), to the one where—for example because of expert knowledge—the shop already knows the underlying joint probability distribution of all bundle valuations. Besides comparing our new online procedure (referred to as ML) with the previous method (called S), we also perform the same series of experiments with a benchmark procedure (called B), which simply recommends a random bundle from the current bundle's neighborhood. That is, the benchmark does not base the order in which it recommends the next bundle on the estimated likelihood of Pareto improvements, as the ML and S-systems do.

Customers initiate the negotiation by submitting an offer containing an initial bundle b_{init}. To give the shop some room for improvement, we initialize the customer's initial bundle by randomly selecting a bundle b which, in binary representation, has a Hamming distance of 5 to the bundle b^* that is associated with the highest gains from trade.

Both ML and S-system greatly outperform the B-system. To illustrate how well the ML-system performs compared to the S-system, figure 1 shows the relative performance and the number of rounds within which a deal is reached (if a deal is reached in the first place) per customer. Relative performance specifies the percentage of the maximal obtainable Pareto improvements (given the starting point) that is actually realized at the end of a negotiation. To simplify the comparison we hard wired the ML-system to stop exploring after 500 customers. (Remember, learning is completely online, however.) In the exploitation phase of the ML-system, the two systems generate virtually the same performance. Figure 1 illustrates the great advantage of using negotiation data. In the exploration phase 500 customers suffices for the ML-system, because per customer the system obtains roughly 55 high quality data points.

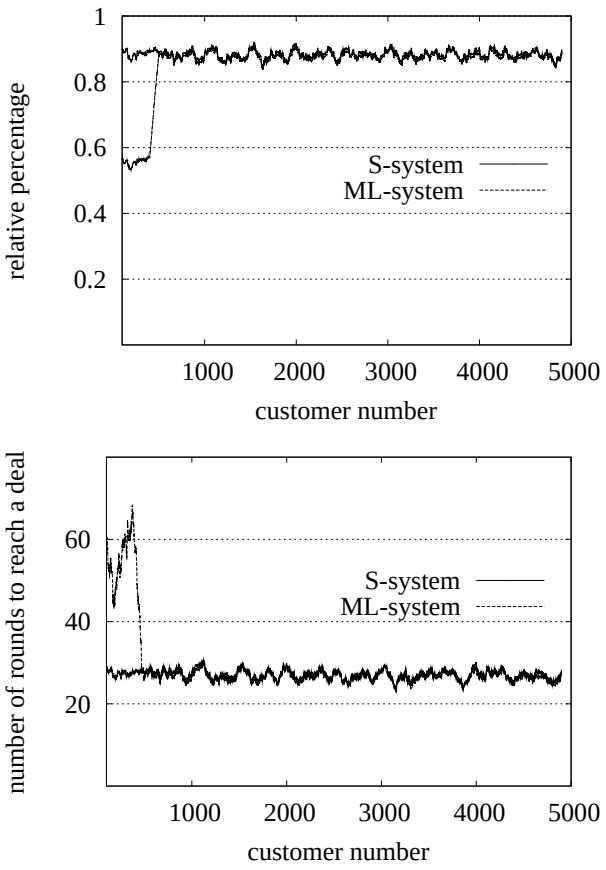


Figure 1: 100-step moving average of the relative % and the number of rounds to reach a deal.

6. CONCLUSIONS

We conduct computer experiments with simulated customers that have highly *nonlinear* preferences and are interested in bundles containing up to 20 individual goods. We compare our new method of having *no a priori information* and learning about customer preferences online, to the one where—for example because of expert knowledge—the shop already knows the underlying joint probability distribution of all bundle valuations. Our experiments show how, after 500 customers, the two approaches have virtually the same performance. Both methods significantly increase the speed with which deals are reached, as well as the number and the Pareto efficiency of the deals reached, compared to a benchmark. Moreover, the experiments show that the new procedure is able to learn the necessary information online, even if due to strategic behavior customers' best offers are significantly less than their reservation values.

7. ACKNOWLEDGMENTS

This work builds on earlier joined work with Tomas Klos.

8. REFERENCES

[1] R. M. Coehoorn and N. R. Jennings. Learning an opponent's preferences to make effective multi-issue negotiation tradeoffs. In *Proc. 6th Int Conf. on E-Commerce*, pages 59–68. ACM press, 2004.

[2] P. Faratin, C. Sierra, and N. R. Jennings. Using similarity criteria to make issue trade-offs. *Journal of Artificial Intelligence*, 142(2):205–237, 2003.

[3] M. Klein, P. Faratin, H. Sayama, and Y. Bar-Yam. Negotiating complex contracts. *Group Decision and Negotiation*, 12:111–125, 2003.

[4] D. J. A. Somefun, T. B. Klos, and J. A. La Poutré. Online learning of aggregate knowledge about nonlinear preferences applied to negotiating prices and bundles. In *Proc. 6th Int Conf on E-Commerce*, pages 361–370. ACM press, 2004.

Towards Formal Model of Adversarial Action in Multi-Agent Systems*

Michal Pěchouček[1], Jan Tožička[1] and Martin Rehák[2]
Gerstner Laboratory[1] and Center for Applied Cybernetics[2]
Department of Cybernetics, Czech Technical University
Technická 2, Prague, 166 27, Czech Republic
pechouc@labe.felk.cvut.cz

ABSTRACT

Detecting and preventing the adversarial action of an agent with respect to the community of agents can be a serious issue in the design of open multi-agent systems. This task is severely domain dependent and it is hard to find a general solution. This contribution presents a utility based model of adversarial action and analyses few properties of adversarial behavior in multi-agent systems. Potential use and important drawbacks of this model are discussed in the paper.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Theory

Keywords

Adversarial action, Cooperation, Competitiveness, Formal model

1. INTRODUCTION

In this contribution, we use the conflict theory [5] and some fundamental principles from the economy and law [2] to consistently define the adversarial behavior in the multi-agent system. The agents in such system can have their own, sometimes completely antagonistic goals and they often compete for the shared resources or opportunities. Tackling a similar problem in conflict theory, James Fearon [5] analyzes the war between perfectly rational states, where the most important property of the war is the war's **ex-post inefficiency** – he argues that the states can reach the same result by negotiation, eliminating the cost of the adversarial actions. In [9], the aggression is defined as an action that is *socially undesirable and imposing net social cost*, consistently with [2], where the criminal activity is defined by imposing that *the gains of criminal are smaller than the social cost of act*.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. ADVERSARIAL ACTION DEFINITION

Consistently with previous work [3], we will define a simple abstract game model featuring agent set $Ag = \{A, B, C, ...\}$ with the agents playing a non-extensive (single round) game that is not strictly competitive – sum of all agents' utilities is not constant. Each agent X has a set of available actions denoted a^*_X, with actions $a^i_X \in a^*_X$ (whenever possible, we only write a_X). From this set, agent selects its action using its strategy. We suppose that all agents do the selection at one moment and therefore the selected actions are independent. The final state, *outcome* of the game $o(a_A, a_B, ...)$ is determined by strategies of the agents and determines both the individual agents' utilities $u_A(o), u_B(o), u_C(o), ...$ and the social choice function $u(o) = u_A(o)+u_B(o)+u_C(o)+...$, considered to represent the *social welfare* [4].

Utility is defined as *"a value which is associated with a state of the world, and which represents the value that the agent places on that state of the world"* by [8].

In the cooperative environment, all agents do share a single utility function.

DEFINITION 1. *We say that agent's A action a^{coop}_A is a* **cooperative action** *if it maximises social welfare:*

$$\texttt{coop}(a^{coop}_A) \Leftrightarrow u(a^{coop}_A, a_B, ...) = \max_{a^i_A \in a^*_A} u(a^i_A, a_B, ...)$$

The complete opposite is the self-interested environment, where the agents are trying to maximise their profit.

DEFINITION 2. *We say that agent's A action a^{si}_A is a* **self-interested action** *if it maximises agents' individual utility:*

$$\texttt{si}(a^{si}_A) \Leftrightarrow u_A(a^{si}_A, a_B, ...) = \max_{a^i_A \in a^*_A} u_A(a^i_A, a_B, ...)$$

In many contexts, the terms self-interestedness and competitiveness are considered to be synonymous. However, we consider the competitiveness to be more strict. In [3], self interestedness is defined as not taking the utility of the others into the consideration while maximizing own utility, while [6] requires the trust between competitors, allowing them to avoid globally undesirable outcomes. In the systems with carefully programmed mechanisms, the results are equivalent in both cases. However, in many real-world cases the total utility may decrease, even if each agent optimizes locally (a prison dilemma is an example of this situation).

In the competitive environment, agents select actions to maximize their own private utility, but they restrict their choice to the actions that at least conserve the social welfare.

DEFINITION 3. *We say that agent's A action a_A^{comp} is a* **competitive action** *provided that it maximizes individual profit while does not allow drop of the social welfare:*

$$\texttt{comp}(a_A^{comp}) \Leftrightarrow u_A(a_A^{comp}, a_B, ...) = \max_{a_A^i \in a_A^{**}} u_A(a_A^i, a_B, ...),$$

where $\forall a_A^i \in a_X^{**} : u(a_A^i, a_B, ...) \geq u(a_B, ...)$.

The expression $u(a_B, ...)$ represents hypothetical outcome of the community in the case when the agent A performs no action or is not in the community at all. This situation is illustrated later on Figure 1 by $\sim$ symbol.

Finally, let us try to define the concept of the adversarial action. The most intuitive definition would be that the adversarial action is such an action that is deliberatively preferred to another action that is equally achievable but has got higher social welfare utilities.

DEFINITION 4. *We say that agent's A action a_A^{adv} is an* **adversarial action** *if:* $\texttt{adv}(a_A^{adv}) \Leftrightarrow \exists a_A^i \in a_A^* :$

1. $u(a_A^{adv}, a_B, ...) \ll u(a_A^i, a_B, ...)$ *and*
2. $u(a_A^i, a_B, ...) - u(a_A^{adv}, a_B, ...) \gg u_A(a_A^{adv}, a_B, ...) - u_A(a_A^i, a_B, ...)$.

The definition 4 above states that the adversarial action a_A^{adv} selected by A from the set a_A^* hurts the social welfare without strong incentive. To make the formalism simpler, we have assumed that there is only single action a_A^{adv} of agent A that hurts the social welfare. There are several interesting points to consider in the general definition.

The first point is the non-emptiness of the set $a_A^* \setminus \{a_A^{adv}\}$ - we don't consider the behaviour with no alternative as adversarial.

Motivation and justification of the adversarial action is closely related to two relational operators used in the definition: $\ll$ and $\lesssim$. The first inequality $\ll$ signifies that the agent shall not cause significant harm to the common welfare, while the inequality $\lesssim$ means that the agent remains self-interested and it will not lose a significant part of its welfare to save the utility of other agents. The concept is illustrated by Fig. 1. In this context, it is important not to take our simplification of the game formalism literally and to consider only immediate payoff as the utility – in most systems, agents expect to encounter their partners again in the future and we suppose that the attitudes of their partners towards them and expected future profits are included in the utility u_X. Formally, we may pose:

DEFINITION 5. *We say that* ***action*** *a_A^j of agent A is* ***rationally adversarial*** *if it is both self-interested and adversarial. In the action is not self-interested and is adversarial, it is* ***irrationally adversarial****.*

2.1 Properties of Adversarial Action

2.1.1 Pareto-Optimality

In this context, we may mention the relationship between adversariality and Pareto-Optimality[1].

An outcome of an adversarial action is not Pareto optimal. Rationally adversarial action is not Pareto optimal in the situations

[1]Following [7], we denote as o^* a set of all achievable outcomes and we define: Outcome o is considered to be **Pareto optimal** if : (i) it is achievable (i.e. $o \in o^*$) and (ii) not majored by any other outcome $o' \in o^* \setminus \{o\}$, where we define majoring as: $\forall_{X \in Ag} u_X(o') \geq u_X(o)$ and $\exists_{X \in Ag} u_X(o') > u_X(o)$.

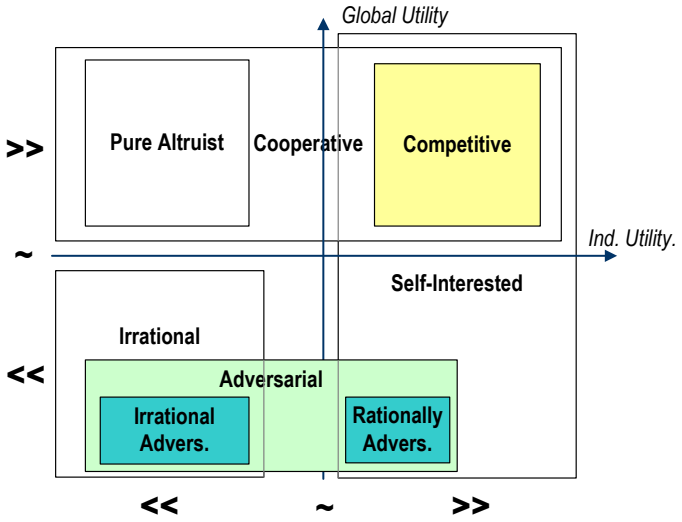


Figure 1: Classification of action with respect to global utility (social welfare) and individual utility of acting agent. The $\sim$ symbol corresponds to the situation where the agent A performs some neutral action or is not in the community at all, i.e. $u(a_B, ...)$.

where the agents may negotiate and transfer the utility - in such situations, the agents may always transfer enough utility to motivate the adversarial agent to behave cooperatively, therefore achieving socially acceptable outcome. When the utility is not transferable (e.g. indivisibility as defined in [5]), the set o^* is severely restricted and even an action that causes the overall social loss may be considered non-adversarial due to the lack of alternative. In the irrationally adversarial case, Pareto optimality does not hold neither, as the utility is lost both by adversarial agent and the society as a whole.

As stated above, an outcome of an adversarial action is not Pareto optimal in the situations:

- where the agents may negotiate and transfer the utility, and
- the action is irrationally adversarial – $u_A(a_A^{adv}) < 0$.

This property can be used mainly in the situation where the community consists of a coalition of mutually trusted actors and one agent whose adversariality is subject of investigation.

On the other hand, Pareto optimality as such doesn't preserve social welfare (due to the indivisibility), it only ensures that all agents behave rationally given the knowledge about the action of the others.

2.1.2 Implication of Adversarial Behaviour

We are interested in the impact of the adversarial action on the global social welfare of the community Ag. We can say that:

- decrease of social welfare *does not imply* existence of an adversarial behavior in the community,
- unnecessary decrease of social welfare *implies* existence of an adversarial behavior (intentional or unintentional) in the community, while
- existence of an adversarial behavior in the community *does not imply* decrease of social welfare.

3. EXAMPLE: UAV DECONFLICTION

In this example, we will illustrate rather abstract definitions of adversarial action provided above with the real example from domain of UAV (Unmanned Aerial Vehicle) flight-plan deconfliction. Let us operate two UAVs A and B. We have situation where the two UAVs are facing a collision and they individually deliberate about the actions $d(A)$ – the UAV A making the deconfliction manoeuver, the action $d(B)$ – the UAV B making the deconfliction manoeuver and $d(A, B)$ – the both UAVs making the deconfliction manoeuver. As the UAVs are different, they loss of their individual utility associated with the manoeuver is different. Let us assume:

- $mu_{\mathtt{d}(A)}(A) > mu_{\mathtt{d}(B)}(B) > mu_{\mathtt{d}(A,B)}(B) + mu_{\mathtt{d}(A,B)}(A)$
- $mu_{\mathtt{d}(A,B)}(A) > mu_{\mathtt{d}(A)}(A)$,
- $mu_{\mathtt{d}(A,B)}(B) > mu_{\mathtt{d}(B)}(B)$

The A UAV is smaller than the B UAV, which implies that it is cheaper for A to make the manoeuver than it is for B. However B making the manoeuver is even cheaper than sum of costs for both UAVs making the manoeuver. However, for each UAV it is cheaper to participate in a collective manoeuver than doing the manoeuver individually.

In the *cooperative environment* the UAVs with conflicting plans do minimize overall disruption and fuel consumption while solving deconfliction problems[2]:

- $A, B : \mathtt{d}(A) \succ \mathtt{d}(B) \succ \mathtt{d}(A, B) \succ \neg\mathtt{d}(A, B)$

Both the cooperative UAVs have the same strategy, that is the smaller UAV deconfliction is preferred to the bigger UAV deconfliction that is preferred to both plans deconfliction and which is preferred to a confliction.

In the *competitive environment* each UAV minimizes its own plan disruption and fuel consumption, but conserves overall welfare:

- $A : \mathtt{d}(B) \succ \mathtt{d}(A) \succ \mathtt{d}(A, B) \succ \neg\mathtt{d}(A, B)$
- $B : \mathtt{d}(A) \succ \mathtt{d}(B) \succ \mathtt{d}(A, B) \succ \neg\mathtt{d}(A, B)$

More specifically, if two UAVs would collide and only one evasive maneuver is necessary, it will try to make the other UAV divert from its course, but without compromising the security.

In the *self-interested environment* each UAV minimizes its own plan disruption and fuel consumption, regardless of the others.

- $A : \mathtt{d}(B) \succ \mathtt{d}(A, B) \succ \mathtt{d}(A) \succ \neg\mathtt{d}(A, B)$
- $B : \mathtt{d}(A) \succ \mathtt{d}(A, B) \succ \mathtt{d}(B) \succ \neg\mathtt{d}(A, B)$

In case of conflict, it only diverts from its course to protect its own safety.

In the adversarial environment the adversarial UAV causes a significant disruption of the other's plans, or even endangers them. Let us assume $mu_{\mathtt{d}(A)}(A) \ll mu_{\mathtt{d}(B)}(B)$ (e.g. A is pushing B away).

- $A : \mathtt{d}(B) \succ \neg\mathtt{d}(A, B) \succ \{\mathtt{d}(A, B), \mathtt{d}(A)\}$
- $B : \mathtt{d}(A) \succ \mathtt{d}(A, B) \succ \mathtt{d}(B) \succ \neg\mathtt{d}(A, B)$

If the B flying over an enemy area does not want to be pushed, the situation is as follows:

- $A : \mathtt{d}(B) \succ \neg\mathtt{d}(A, B) \succ \{\mathtt{d}(A, B), \mathtt{d}(A)\}$
- $B : \mathtt{d}(A) \succ \mathtt{d}(A, B) \succ \neg\mathtt{d}(A, B) \succ \mathtt{d}(B)$

[2]The symbol $\succ$ stands for collective choice preference

4. CONCLUSION

The problem of adversariality in the multi-agent systems is real. While the irrationally adversarial agents may be easy to identify, it may be much more difficult to identify the rationally adversarial behavior, especially if all the agents in the system are self-interested. In this context, the question of *bounded rationality* of agent's reasoning is crucial. To better illustrate the concept, we will cite several accepted causes for the emergence of the conflict between the rational actors. It is easy to realize that most of these causes can plausibly exist in the multi-agent system and shall be considered while designing autonomous agents: **Private information** of each agent is not available to the others, providing one of the causes of **miscalculation** about **capabilities or attitudes** of the other party. Such miscalculation may cause an adversarial behavior, as the agents will not be able to correctly estimate the utility function of the partners. Agents are often willing to **misrepresent** the reality about themselves, in order to obtain better payoff or negotiation position in the future.

5. ACKNOWLEDGEMENT

Effort sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-04-1-3044 and FA8655-04-1-3044-P00001. The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon[3]. We also gratefully acknowledge the support of the presented research by ARL project N62558-03-0819.

6. REFERENCES

[1] R. Axelrod. The evolution of strategies in the iterated prisoner's dilemma. In L. Davis, editor, *Genetic algorithms and simulated annealing*, pages 32–41. Research notes in AI, Pitman/Morgan Kaufmann, 1987.

[2] Gary S. Becker. Crime and punishment: An economic approach. *The Journal of Political Economy*, 76(2):169–217, 1968.

[3] Sviatoslav Brainov. The role and the impact of preferences on multiagent interaction. In *ATAL '99: 6th International Workshop on Intelligent Agents VI, Agent Theories, Architectures, and Languages (ATAL),*, pages 349–363. Springer-Verlag, 2000.

[4] Rajdeep K. Dash, Nicholas R. Jennings, and David C. Parkes. Computational-mechanism design: A call to arms. *IEEE Intelligent Systems*, 18(6):40–47, 2003.

[5] James D. Fearon. Rationalist explanations for war. *International Organization*, 49(3):379–414, 1995.

[6] D. Gambetta, editor. *Trust: Making and Breaking Cooperative Relations*. Basil Blackwell, 1990.

[7] Milan Mares. Fuzzy coalition structures. *Fuzzy Sets Syst.*, 114(1):23–33, 2000.

[8] Simon Parsons and Michael Wooldridge. Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 5(3):243–254, 2002.

[9] Eric A. Posner and Alan O'Neil Sykes. Optimal war and jus ad bellum, 2004.

[3]The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

Negotiating Efficient Outcomes over Multiple Issues

Sabyasachi Saha
Math & CS Department
University of Tulsa
sabyasachi-saha@utulsa.edu

Sandip Sen
Math & CS Department
University of Tulsa
sandip-sen@utulsa.edu

ABSTRACT

It is difficult to reach optimal outcomes in bilateral negotiations with multiple issues when the agents' preferences and priorities of the issues are not common knowledge. Self-interested agents often end up negotiating inefficient agreements in such situations. Some existing multiagent negotiation frameworks involve agents revealing their preferences to a trusted mediator. But in real-life situations, such a third party, trusted by both the agents may not be found. We design a protocol for bilateral multi-issue negotiation. This protocol guarantees envy-free and Pareto optimal agreement with minimum revelation of their preferences when both negotiators are rational and have the same ordinal preferences. This protocol also leads rational agents to envy-free and near-optimal solution in all cases.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Performance

Keywords

Negotiation, multi-issue, efficiency

1. INTRODUCTION

Negotiation is the most well-known approach for resolving conflicts in human and agent societies. With the rapid growth of E-commerce, research in automated negotiation has become increasingly important [4]. In automated negotiation, agents negotiate contracts on behalf of real-life negotiating parties they represent. An agent can negotiate resources and services with other agents or humans. Agents may bargain over a single issue of contention, but negotiations often involve bargaining over multiple issues [5], which may or may not be correlated. The issues are considered to be correlated if the utility of one issue depends on the consumption of any other issue(s). In this paper, we consider one-time negotiations with two agents negotiating over multiple, unrelated issues. We assume that the participating agents have no *a priori* knowledge about the preferences of the other agent. This situation is common in E-commerce negotiations, where the number and diversity of agents is so large that an agent may not have any estimate about the preferences of any particular agent it is negotiating with.

A negotiation process can be *distributive* or *integrative* in nature. In a *distributive* negotiation scenario, a gain for one participant implies a loss for the other. In *integrative* negotiation, agents can reach outcomes that are mutually beneficial. A single issue negotiation is usually a *distributive* negotiation. Multi-issue negotiations, on the other hand, where the ordinal and cardinal preferences are different, are integrative negotiations. In multi-issue negotiation, agents with known, divergent preferences can cooperate to reach agreements beneficial for both agents. But when the preferences are not known, using existing protocols, self-interested agents often fail to explore win-win possibilities and end up with inefficient agreements. By efficient or optimal solution, we refer to a solution which is *envy-free* and *Pareto optimal* [1].

Existing research shows that, if the agents are willing to reveal their complete preferences, it is possible to reach an efficient solution [3, 5]. If the agents, however, are unwilling to reveal their preferences at all, rational agents will end up with an equal split solution[1], which is not Pareto optimal unless both agents have identical preferences. So, there exists an efficiency-revelation trade-off. Our goal in this paper is to find out effective protocols for bilateral, multi-issue negotiation that can lead the participating agents to optimal or near-optimal outcomes while revealing only minimal information. Our work has some similarity with the work on contract negotiation by Sandholm [6] and Endriss *et al* [2] in the context of task allocation, where agents negotiate to reallocate tasks. But they want to identify, under complete information, necessary deals or exchanges required among agents to reach efficient allocation from an arbitrary initial allocation. We, on the contrary, want to find mechanisms that lead agents, without any prior information about opponent's preferences, to efficient outcomes without any third party intervention.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

[1]Each agent gets exactly half of each issue.

2. NEGOTIATION FRAMEWORK

We formally describe a representative negotiation scenario for allocation of available resources as a 3-tuple $\langle \mathcal{A}, R, \mathcal{U} \rangle$, where $\mathcal{A} = \{A_1, A_2\}$ is the set of agents, R is the resource whose allocation is being negotiated, and $\mathcal{U} = \{U_1, U_2\}$ is the set of utility functions, where U_i is the utility function of agent i. The resource $R = \{r_1, r_2, \ldots, r_H\}$ has $H \geq 2$ components and we assume that one continuously divisible unit of each component is available for allocation. Each component can be considered as a negotiation issue. Henceforth we will use the terms "component" and "issue" interchangeably. The negotiating agents must agree on the allocation of each component of the resource. An agent wants to obtain the entire unit of each components and some components are more important to it than others. Each agent has a complete preference order over the different components, which is not known to the other agent. Each agent can only assume that other agent is rational and maximizes its utility.

We now define some terms formally:

Outcome: An outcome O is an allocation of the resource components to different agents. Formally, $O : R \mapsto \mathbb{R}^{2\times H}_{[0,1]}$, where $\mathbb{R}^{2\times H}_{[0,1]}$ is a $2 \times H$ matrix, whose (i, j) cell, $x^O_{ij} \in [0, 1]$, presents agent i's allocation of the jth resource or component such that $x^O_{1j} + x^O_{2j} \leq 1$. If the agents cannot reach an agreement, they receive no share of the component, *i.e.*, $x^O_{ij} = 0$ for all i and j. Otherwise, $\sum_{i=1}^{2} x^O_{ij} = 1$, for $j \in \{1, \ldots, H\}$, *i.e.,* each component is exhaustively partitioned between the agents.

Total preference information: This is the relative importance of all the issues to a particular agent. For any agent i, total preference information is given by $\{w_{i1}, \ldots, w_{iH}\}$, where $w_{ij} \geq 0$ and $\sum_{j=1}^{H} w_{ij} = 1$. This is an agent's private information.

Utility: For agent i, utility of an outcome O, is defined as:

$$U^O_i = \sum_{j=1}^{H} w_{ij} * x^O_{ij}. \quad (1)$$

Pareto optimality: An outcome O is Pareto optimal, if there exists no other outcome O' such that all agent's utility in O' is at least as good as their utility in O and at least one agent's utility is more in O'. If outcome O is Pareto optimal, then there cannot exist any outcome O' such that, $U^{O'}_i \geq U^O_i$ for both $i = 1, 2$ and $U^{O'}_i > U^O_i$ for at least one i.

Envy-free: An outcome is envy-free, if and only if each agent's total utility for its own share is at least as much as its utility for the share assigned to the other agent [1].

3. PROPOSED NEGOTIATION PROTOCOL

In this section, we present the protocol that guarantees envy-free and Pareto optimal outcomes to two rational agents negotiating over multiple issues if their ordinal preferences are same. We refer to this protocol as the protocol leading to Pareto Optimal Solution through Equivalence (POSE).

3.1 POSE Protocol

In this protocol, two agents interact with a commonly observable random device. Two agents are negotiating over $H \geq 2$ components where each component has one continuously divisible unit. The protocol is as follows:

Step 1: Number the components from 1 to H and for each component the unit is initially split in equal halves between the two agents. Hence, each agent possesses 0.5 unit of all the components. Let, D^i_t is Agent A_i's current possession of component t. So, initially, $D^i_t = 0.5$, for all $t = 1 \ldots H$, and $i = 1, 2$.

Step 2: The random device chooses one of the two agents, S, to start. Let the set of components yet to be negotiated be G. Initially, $G = R$.

Step 3: Now, S will choose one of the remaining components from G, say C, and propose an exchange offer $E(C, Q_C)$. The offer means that S is ready to trade Q_C units of each of the $G - \{C\}$ components for one unit of C.

Step 4: The other agent, say S', can choose whichever of the options it prefers: either one unit of component C or Q_C units of each of the components in the set $G - \{C\}$.

For S', one unit of C produces $w_{S'C}$ utility and Q_C unit of each of the remaining $G - \{C\}$ components together produces $\sum_{j \in G-\{C\}} Q_C * w_{S'j}$ utility. S' will choose whichever one is more profitable to it. $D^{S'}_C$ and $D^S_C = 1 - D^{S'}_C$ are the amount or unit of component C currently possessed by agent S and S' respectively.

If S' prefers C, then it can earn up to $(1 - D^{S'}_C)$ unit of C in exchange of its own $\{(1 - D^{S'}_C) * Q_C\}$ unit of each component in $G - \{C\}$. But if $\{(1 - D^{S'}_C) * Q_C\} > D^{S'}_{a \in G-\{C\}}$, then the entire exchange is not possible[2]. In this situation S' can give $D^{S'}_{a \in G-\{C\}}$ unit of each component in $G - \{C\}$ and earn $\frac{D^{S'}_{a \in G-\{C\}}}{Q_C}$ unit of C in return. On the other hand if it chooses the other option, it will receive $\{(1 - D^S_C) * Q_C\}$ unit of each component in $G - \{C\}$ from S and S will receive $(1 - D^S_C)$ unit of C from S' and thereby own the entire unit of C. Similar to the earlier case, if $\{(1 - D^S_C) * Q_C\} > D^S_{a \in G-\{C\}}$, S' will receive $D^S_{a \in G-\{C\}}$ unit of each component in $G - \{C\}$ and give $\frac{D^S_{a \in G-\{C\}}}{Q_C}$ unit of C to S. D^i_a values are updated accordingly to present agent i's current possession of component a.

Step 5: The roles of agents S and S' are swapped and G is changed to $G - \{C\}$. If $|G| > 1$ return to Step 3, Otherwise stop. The final allocation is given by $x_{ij} = D^i_j$, where x_{ij} is the agent i's possession of component j.

In this protocol, agents only alternatively reveal their partial preference information.

[2]Since all the remaining issues are taken together for every exchange, for each agent, its possession of each of the issues yet to be negotiated will be same, i.e., for any agent S, D^S_a is same for all $a \in G - \{C\}$.

Proposition 1: *In POSE protocol rational agents produce an envy-free, Pareto-optimal solution if they have identical ordinal preferences*[3].

3.1.1 Rational strategy of agents in POSE

Now we analyze an agent's rational behavior when using the POSE protocol for negotiation. The negotiation strategy consists of two decisions. First, when an agent is proposing an exchange offer, denoted by E, which component, C, will it choose and what should be corresponding exchange ratio, Q_C. The other decision by an agent involves selecting between two options proposed by the other agent.

Statement 1: *When an agent has to select between two options, it will choose the one with higher utility.*

Proposition 2: *While proposing, an agent i, should choose the component C such that, $C = \arg\min_{j \in G}(w_{ij})$ and should propose a ratio Q_C, such that*

$$w_{iC} = \sum_{j \in G - \{C\}} w_{ij} * Q_C.$$

An agent proposes, according to its true preference, its least preferred option from the remaining issues.

3.1.2 An example

Let us illustrate the protocol with an example: two rational agents A_1 and A_2 are negotiating for a resource with three components. One unit of each of the three components are available for negotiation. Suppose ordinal preferences of both the agents are same and their total preference is given by $\{0.1, 0.4, 0.5\}$ and $\{0.2, 0.3, 0.5\}$ respectively. So, $w_{i1} \leq w_{i2} \leq w_{i3}$ for both the agents $i = 1, 2$. Initially each of the components is divided equally between the two agents, *i.e.,* each agent receives 0.5 unit of each component. Now, suppose, the random device chooses A_2 as the first agent (denoted by S in the POSE protocol). Then A_2 will choose the first issue to propose as this issue has the least relative weight to A_2 and it proposes an offer $E(1, \frac{0.2}{0.3+0.5} = 0.25)$ to A_1. So, A_2 is willing to trade 0.25 units of each of components 2 and 3 for one unit of component 1. For A_1, 0.25 unit of components 2 and 3 together is of higher utility than one unit of component 1. So, it will exchange its possession of 0.5 unit of component 1 for $0.25 * 0.5 = 0.125$ units of components 2 and 3 from A_2. So, after this exchange, the possession of components is $\{0, 0.625, 0.625\}$ for A_1 and $\{1, 0.375, 0.375\}$ for A_2 and the set of issues which are yet to be negotiated is $\{2, 3\}$. Now it is A_1's turn to propose and it chooses component 2, its least preferred issue among the remaining issues and offers $E(2, \frac{0.4}{0.5} = 0.8)$. A_2 prefers 0.8 units of component 3 than one unit of component 2. A_2 has 0.375 unit of component 2. So, it will relinquish 0.375 unit of component 2 and receive $0.375 * 0.8 = 0.3$ unit of component 3 from A_1. So, after this exchange the possessions of A_1 and A_2 are $\{0, 1, 0.325\}$ and $\{1, 0, 0.675\}$, respectively. The utility of A_1 and A_2 are 0.5625 and 0.5375, respectively. Now, the number of issues left to be negotiated is one. So, we stop this process and we claim that this allocation of the units is Pareto optimal as one agent's utility cannot be improved without decreasing the other agent's utility. In Figure 1, the corresponding utility pair is shown by point $P_1 = (0.5625, 0.5375)$.

[3]We exclude the proof of the propositions due to space limitation.

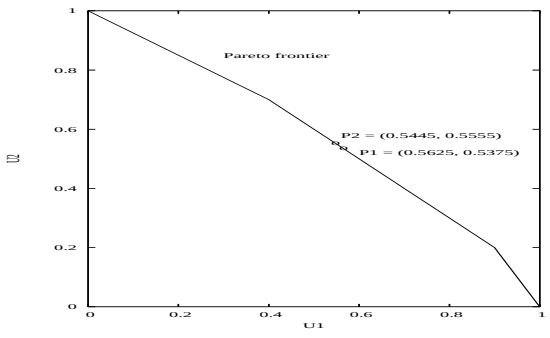


Figure 1: Bargaining solutions when weight of agent 1 is $\{0.1, 0.4, 0.5\}$ and weight of agent 2 is $\{0.2, 0.3, 0.5\}$.

If the random device chooses A_1 as the first agent instead of A_2 as in the above example, a similar process would lead the agents to the outcome where the possessions of A_1 and A_2 are $\{0, 1, 0.289\}$ and $\{1, 0, 0.711\}$, respectively, and the resultant utility pair is $P_2 = (0.5445, 0.5555)$, shown in Figure 1. Both the outcomes, P_1 and P_2, lie on the Pareto-frontier.

4. CONCLUSIONS AND FUTURE WORK

In multi-issue negotiation between two rational agents, there is a trade off between the quality of the agreement reached and the amount of information revealed. We proposed one protocol, where rational agents, without prior knowledge of the opponent preference, are guaranteed to reach a stable, envy-free, Pareto optimal solution if their ordinal preference rankings are identical. For arbitrary agent preferences, this protocol will lead rational agents to Pareto optimal solutions in most cases or otherwise very close to the Pareto optimal solutions while revealing minimal information.

Acknowledgments: This work has been supported in part by an NSF award IIS-0209208 and an NASA EPSCoR RIG.

5. REFERENCES

[1] S. J. Brams and A. D. Taylor. *Fair Division.* Cambridge University Press, 1996.

[2] U. Endriss, N. Maudet, F. Sadri, and F. Toni. On optimal outcomes of negotiations over resources. In *AAMAS-03*, pages 177–184, New York, NY, USA, 2003.

[3] Y. In and R. Serrano. Agenda restrictions in multi-issue bargaining. *Games and Economic Behavior*, 53:385–399, 2004.

[4] N. Jennings, P. Faratin, A. R. L. S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: prospects, methods and challenges. *International Journal of Group Decision and Negotiation*, 10(2):199–215, 2001.

[5] H. Raiffa. *The Art and Science of Negotiation.* Harvard University Press, Cambridge, MA, USA, 1982.

[6] T. W. Sandholm. Contract types for satisficing task allocation: I theoretical results, *AAAI Sprint Symposium*, 1998.

Managing Social Influences through Argumentation-Based Negotiation

Nishan C. Karunatillake[1], Nicholas R. Jennings[1], Iyad Rahwan[2], Sarvapali D. Ramchurn[1]

[1] School of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK.
{nnc02r,nrj,sdr}@ecs.soton.ac.uk

[2]Institute of Informatics
The British University in Dubai
P.O.Box 502216 Dubai, UAE.
irahwan@acm.org

ABSTRACT

Social in uences play an important part in the actions that an individual agent may perform within a multi-agent society. However, the incomplete knowledge and the diverse and con icting in uences present within such societies, may stop an agent from abiding by all its social in uences. This may, in turn, lead to con icts that the agents need to identify, manage, and resolve in order for the society to behave in a coherent manner. To this end, we present an empirical study of an argumentation-based negotiation (ABN) approach that allows the agents to detect such con icts, and then manage and resolve them through the use of argumentative dialogues. To test our theory, we map our ABN model to a multi-agent task allocation scenario. Our results show that using an argumentation approach allows agents to both ef ciently and effectively manage their social in uences even under high degrees of incompleteness. Finally, we show that allowing agents to argue and resolve such con icts early in the negotiation encounter increases their ef ciency in managing social in uences.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]

General Terms

Theory, Algorithms, Experimentation

Keywords

Argumentation-based Negotiation, Con ict Resolution.

1. SOCIAL ARGUMENTATION MODEL

In abstract, our framework consists of four main elements: (i) a *schema* for reasoning about social in uence, (ii) a set of *social arguments* that make use of this schema, (iii) a *language and protocol* for facilitating dialogue about social in uence, and (iv) a set of *decision functions* that agents may use to generate dialogues within the protocol (for a comprehensive formal representation of the framework refer to [3, 4]).

1.1 Social Influence Schema

The notion of *social commitment* [1] acts as our basic building block for capturing social in uence. In essence, a social commitment ($\mathrm{SC}_{\theta}^{x \rightarrow y}$) is a commitment by one agent x (termed the *debtor*) to another y (termed the *creditor*) to perform a stipulated action . As a result of such a social commitment, the debtor is said to attain an *obligation* toward the creditor, to perform the stipulated action. The creditor, in turn, attains certain rights (i.e., the right to demand or require the stipulated action, the right to question its non-performance, and the right to demand compensation to make good any losses suffered due to its non-performance). We refer to these as *rights to exert influence*. We extend this notion to capture social in uences resulting due to factors such as roles and relationships within a wider multi-agent society (i.e., those that rely on the structure of the society, rather than the speci c individuals who happen to be committed to one another). Speci cally, since most relationships involve the related parties carrying out certain actions for each other, we can view a relationship as an encapsulation of social commitments between the associated roles. Figure 1 formulates these notions as a schema of social in uence.

1.2 Social Arguments

The social in uence schema can be used to systematically identify social arguments that can be used to negotiate in the presence of social in uences. Speci cally, we identify two major ways:

Socially Influencing Decisions: In a social context, an agent can affect another agent's decisions by arguing about the validity of the other's social reasoning. Speci cally, agents can argue about the premises within the social in uence schema to undercut and rebut the justi cations to perform certain actions within a society.

Negotiating Social Influence: Agents can also use social in uences (i.e., obligations and rights) as additional parameters within their negotiations. Speci cally, here the agents use negotiation as a tool for trading social in uences .

1.3 Language and Protocol

To enable agents to express their arguments, we de ne two complimentary languages: the *domain language* and the *communication language*. The former allows the agents to express premises about their social context and the latter to construct arguments to engage in their discourse to resolve con icts. The protocol indicates the legal ordering of communication utterances and has six main stages: (i) *opening*, (ii) *conflict recognition*, (iii) *conflict diagnosis*, (iv) *conflict management*, (v) *agreement*, and (vi) *closing*. In operation, it is de ned as a dialogue game protocol which gives locutions rules, commitment rules, and structural rules for each of the locutions in the communication language.

1.4 Decision Making Functionality

The protocol described above gives agents a number of different options, at various stages, as to what utterances to make. For instance, after a proposal the receiving agent could either accept or reject it. After a rejection, the agent may choose to challenge this rejection, end the dialogue, or forward an alternative proposal. An

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

An agent a_i acting the role r_i
Leads it to be part of the relationship p
With another agent a_j acting the role r_j

A social commitment $SC_\theta^{r_i \rightarrow r_j}$ associated with p

- Leads to a_i attaining an obligation O toward r_j,
  Which subjects it to an influence of degree f
  To perform the action θ
- And, in turn, leads to a_j attaining the right R toward r_i
  To demand, question, and require the performance of action θ

Figure 1: Schema of Social Influence.

	r_0	r_1	r_2
r_0	0	1	0
r_1	1	0	1
r_2	0	1	0

(a) Rol-Rel map

	r_0	r_1	r_2
r_0	[0:0]	[200:0]	[0:0]
r_1	[400:100]	[0:0]	[200:600]
r_2	[0:0]	[700:200]	[0:0]

(b) Social commitment map

	r_0	r_1	r_2
a_0	1	0	0
a_1	0	1	1
a_2	0	1	0

(c) Ag-Rol map

Figure 2: Social Influence Model.

agent, therefore, still requires a decision mechanism for selecting a particular utterance among the available legal options. To this end, for each of the possible dialogue moves, we specify general decision making algorithms to give the agents that capability (see [3]).

2. ARGUMENTATION CONTEXT

To evaluate how our argumentation model can be used to manage social in uences, we require a computational context, in which a number of agents interact in the presence of social in uences and con icts arise as a natural consequence of these interactions. In abstract, the context is based on a multi-agent task allocation scenario where a collection of self-interested agents interact to obtain services to achieve a given set of actions (similar to that of [2]).

2.1 Modelling Social Influences

To map social in uences to this context we *first* embody a role-relationship structure into the multi-agent society by randomly linking a de ned number of roles to create relationships (see Figure 2(a) where 1 indicates that a relationship exists between the two related roles). *Second*, to this structure, we randomly specify a set of social commitments. We do so by de ning a decommitment penalty cost for each capability in each of the active edges (see Figure 2(b)). For instance, in Figure 2(b) the entry [400:100] in row 1, column 2 indicates that the role r_0 is committed to provide capabilities c_0 and c_1 to a holder of the role r_1. If the agent holding the role r_0 chooses to violate this commitment it will have to pay 400 and 100 (respectively for c_0 and c_1) if asked. *Finally*, we assign these roles to the actual agents using the mapping shown in Figure 2(c).

Since one of the aims in our experiments is to test how agents use argumentation to manage and resolve con icts created due to incomplete knowledge about their social in uences, we generate a number of settings by varying the level of knowledge seeded to the agents. Speci cally, we give only a subset of the agent-role mapping. Thus, a certain agent may not know all the roles that it or another agent may act. This lack of knowledge may, in turn, lead to con icts within the society, since certain agents may know certain facts about the society that others are unaware of. By controlling the level of missing knowledge, we generate an array of settings ranging from perfect knowledge (0% missing knowledge) to the case where agents are completly unaware of their social in uences (100% missing knowledge) within the society.

2.2 Agent Interaction

Agents within the system argue and negotiate with each other to nd willing and capable partners to accomplish their actions. In essence, an agent that requires a certain capability will generate and forward *proposals* to another selected agent within the community requesting it to sell its services in exchange for a certain reward. If

Algorithm 1 The ***negotiate()*** method.

```
1: [p_0, p_1, ..., p_max] ← generateProposals()
2: p ← p_0
3: isAccepted ← false
4: {Loop till either the agent agrees or the last proposal fails.}
5: while (isAccepted ≠ true || p ≤ p_max) do
6:     response ← PROPOSE(p)
7:     if (response = "accept") then
8:         isAccepted ← true
9:     else
10:        if (p ≠ p_max) then
11:            p ← getNextViableProposal()
12: return isAccepted
```

Algorithm 2 The ***argue()*** method.

```
1: H_o ← challenegeJustification() {Challenge for the opponent's justification}
2: H_p ← generateJustification() {Generate personal justification}
3: if (isValid(H_o) = false) then
4:     {Assert invalid premises of H_o}
5: else
6:     {Adopt premises of H_o into personal knowledge}
7: if (isValid(H_p) = false) then
8:     {Correct invalid premises of H_p within personal knowledge}
9: else
10:    {Assert H_p}
```

the receiving agent perceives this proposal to be viable and believes it is capable of performing it, then will *accept* it. Otherwise it will *reject* the proposal. In case of a reject, the original proposing agent will attempt to forward a modi ed proposal (Algorithm 1).

Agents detect con icts (such as the one highlighted in Section 2.1) by analysing the decommitment penalties paid by their counterparts for violating their social commitments. Once detected, agents attempt to resolve them by exchanging their respective justi cations. These would take the form of the social in uence schema and are then analysed to diagnose the cause of the con ict. If there are inconsistencies between them, social arguments are used to highlight these. If they are both valid, then each agent would point-out alternative justi cations via asserting missing knowledge (Algorithm2).

3. MANAGING SOCIAL INFLUENCES

To empirically evaluate how our ABN model allows agents to manage their social in uences, we de ne a number of different interaction strategies to allow the agents to manage con icts related to their social in uences and experiment their relative performance bene ts. The underlying motivation for these strategies is our social in uence schema, which gives the agents different rights; namely the right to demand compensation and the right to challenge non-performance of social commitments.

3.1 Demanding Compensation

If an agent violates a social commitment, one of the ways its counterpart can react is by exercising its right to demand compensation (see Section 1.1). This formulates our baseline strategy which extends our negotiation algorithm by allowing the agents to demand compensation in cases where negotiation fails (Algorithm 3). Once requested, the agent that violated its social commitment will pay the related the penalty. However, in imperfect information settings, a particular agent may violate a social commitment simply because it was not aware of it (i.e., due to the lack of knowledge of its roles or those of its counterparts). In such situations, an agent may pay a decommitment penalty different to what the other believes it should get, which may, in turn, lead to con icts. In such situations, our second strategy allows agents to use social arguments to argue about their social in uences and, thereby, manage their con icts (Algorithm 4). Our hypothesis here is that by allowing agents to argue about their social in uences we are providing them with a coherent mechanism to manage and resolve their con icts and, thereby, allowing them to gain a better outcome as a society. To this end, the former strategy acts as our control experiment and the latter as the test experiment (see Figure 3).

Algorithm 3 Claim-Penalty-Non-Argue strategy.

1: $isAccepted \leftarrow$ *negotiate*()
2: **if** $(isAccepted =$ **false**$)$ **then**
3: compensation $\leftarrow$ *demandCompensation*()

Algorithm 4 Claim-Penalty-Argue strategy.

1: $isAccepted \leftarrow$ *negotiate*()
2: **if** $(isAccepted =$ **false**$)$ **then**
3: compensation $\leftarrow$ *demandCompensation*()
4: **if** (compensation $<$ rightToPenalty) **then**
5: *argue*()

Observation 1: *The argumentation strategy allows agents to manage their social influences even at high uncertainty levels.*

The effectiveness of the the non-argue strategy falls more rapidly than the argue one. This is because the argue method allows agents to manage and resolve con icts of opinion that they may have about their social in uences. For instance, if a certain agent is unaware of a role that another acts, it may correct this through arguing with that agent. Thus, arguing allows agents to correct such gaps in their knowledge and, thereby, resolve any con icts that may arise as a result. In this manner, ABN allows the agents to manage their social in uences even at high uncertainty levels. Thereby, as a society, the agents can accomplish more of their actions and gain a higher total earnings value. The non-arguing approach, which does not allow them to argue about their social in uences, leaves such con- icts unresolved and reduces the population earnings as knowledge imperfections increase within the society.

Observation 2: *In cases of perfect information and complete uncertainty both strategies perform equally.*

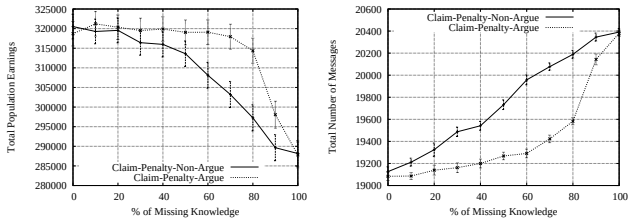
The reason for both strategies performing equally when there is perfect information (0 level) is because there are no knowledge imperfections. Thus, agents do not need to argue to correct con- icts simply because such con icts do not exist. The reason for both strategies performing equally when there is a complete lack of knowledge is because none of the agents in the society are aware of any social in uences (even though they exist). Thus, they are not able to detect any con icts or violations and, as a consequence, agents do not resort to arguing to manage such con icts. Therefore, when there is a complete lack of knowledge, the strategy that uses the argue performs the same as the non-argue one.

Observation 3: *At all knowledge levels, the argumentation strategy exchanges fewer messages than the non-arguing one.*

The reason for this is that, even though agents use some messages to argue and correct their incomplete knowledge, thereafter the agents use their corrected knowledge in subsequent interactions. However, if the agents do not argue to correct such knowledge imperfections, they negotiate more frequently since they cannot use their social in uence. Thus, this one-off increase of argue messages becomes insigni cant when compared to the increase in the propose, accept, and reject messages due to the increased number of negotiations.

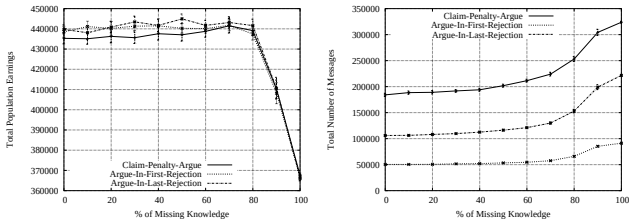
3.2 Questioning Non-Performance

When a particular social commitment is violated our social in uence schema also gives the agents the right to challenge and demand a justi cation for this non-performance. It is generally argued in ABN theory that allowing agents to exchange meta-information gives them the capability to understand each others' reasons and, thus, provides a more ef cient method of resolving con icts under uncertainty. In a similar manner, we believe that providing the agents with the capability to challenge and demand justi cations for violating social commitments allows the agents to gain a wider understanding of the internal and social in uences affecting their counterparts, thereby, providing a more ef cient method for managing social in uences in the presence of incomplete knowledge. Using this intuition as the underlying hypothesis, we use our



(a) Total Population Earnings (b) Total Number of Messages

Figure 3: Efficiency and Effectiveness of the Argue and Non-Argue strategies with 20 Agents and 3 Roles.



(a) Effectiveness (b) Efficiency

Figure 4: Efficiency and Effectiveness of the various argumentation strategies.

previous best strategy *Claim-Penalty-Argue* as the control experiment and design two other strategies (*Argue-In-First-Rejection* and *Argue-In-Last-Rejection*) to experiment with the effect of allowing the agents to challenge non-performance at different stages within the negotiation encounter. The former allows the agent to challenge after the receipt of the rst rejection and the latter after the last rejection. Figure 4 shows our results.

Observation 4: *The effectiveness of the various argumentation strategies are broadly similar.*

This is because all three strategies argue and resolve con icts even though they decide to argue at different points within the encounter. Thus, we do not expect to have a signi cant differences in the number of con icts resolved. Thus, the effectiveness stays the same.

Observation 5: *Challenging earlier in the dialogue, significantly increases the efficiency of managing social influences.*

The reason for this behaviour is based on how the agents use these reasons exchanged during the argue phase. In the *Claim-Penalty-Argue* strategy the main objective of arguing is to resolve the con- ict regarding the penalty value that should be paid. It does not however, attempt to nd out the reason for why its counterpart rejected its proposal. By challenging the reason for the rejection, the latter two strategies gain certain meta-information, which the agents constructively use in their subsequent interactions. For instance, if the counterpart rejected the proposal due to lack of capability, it can be excluded in future if the agent requires a capability which is equal or greater. Arguing in the rst rejection provides this information earlier in the negotiation, which, in turn, gives the agent more capacity to exploit such information (even in the present negotiation) than getting it in the last encounter.

4. REFERENCES

[1] C. Castelfranchi. Commitments: From individual intentions to groups and organizations. In *Proc. of ICMAS'95*, pages 41–48, San Francisco, CA, 1995.

[2] N. C. Karunatillake and N. R. Jennings. Is it worth arguing? In *Argumentation in Multi-Agent Systems*, LNAI 3366, pages 234–250, NY, USA, 2004.

[3] N. C. Karunatillake, N. R. Jennings, I. Rahwan, and T. J. Norman. Arguing and negotiating in the presence of social influences. In *Proc. of CEEMAS'05*, LNAI 3690, pages 223–235, Budapest, Hungary, 2005.

[4] N. C. Karunatillake, N. R. Jennings, I. Rahwan, and T. J. Norman. Argument-based negotiation in a social context. In *Proc. of ArgMAS'05*, pages 74–88, Utrecht, The Netherlands, 2005.

An Expressive Approach to Fuzzy Constraint Based Agent Purchase Negotiation

Miguel A. López-Carmona
Department of Automática, University of Alcalá
Escuela Politécnica, Ctra. NII,km. 33.600
28871 Alcalá de Henares (Madrid), Spain
miguellop@aut.uah.es

Juan R. Velasco
Department of Automática, University of Alcalá
Escuela Politécnica, Ctra. NII,km. 33.600
28871 Alcalá de Henares (Madrid), Spain
juanra@aut.uah.es

ABSTRACT

This paper presents a fuzzy constraint based model for bilateral multi-attribute agent purchase negotiations in competitive trading environments. Argumentation is used as a key mechanism to improve agreements, in contrast to other fuzzy constraint based models which are limited to quantitative offers and counter-offers. A set of locutions and decision mechanisms which fire them are specified, so that each agent may decide its degree of cooperation and expressiveness.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Distributed Artificial Intelligence, Problem Solving

General Terms

Algorithms, Theory, Experimentation

Keywords

Negotiation, Fuzzy Constraints, Strategies

1. INTRODUCTION

The use of fuzzy constraints is an efficient way of capturing requirements, and a useful mechanism for representing trade-offs between the possible values of attributes [1, 3]. Moreover, the solution space can be explored in a given exchange and so means that the search for an agreement is more efficient. The problem with previous approaches is that the ability of the seller agent to express its needs is limited [3]. As a consequence the joint decision process is not balanced. We develop a more expressive approach which lets the buyer agent value the degree of importance that each submitted constraint has, and lets the seller agent inform how constraints should be relaxed. We propose a general framework to implement these mechanisms, so that each agent may decide its degree of cooperation and expressiveness, which in turn may have effects on the speed of convergence and the quality of the agreement.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. DOMAIN KNOWLEDGE

Let $\mathcal{B}_{req} = (X, D, C^f, N_b)$ be the requirement model of the buyer agent, where $X = \{x_i | i = 1, ..., n\}$ is the set of attributes, $D = \{d_i | i = 1, ..., n\}$ is the set of domains of the attributes, and $C^f = \{R_i^f | i = 1, ..., m\}$ is the set of fuzzy constraints which expresses the requirements on the attributes. $N_b = \{\xi, \eta\}$ is the negotiation profile, where $\xi \in \{0, 1\}$ controls the use of purchase requirement valuations, and $\eta \in [0, 1]$ modulates the buyer's attitude regarding a relax requirement. The function that indicates how well a given constraint R_i^f is satisfied is the *satisfaction degree function*: $\mu_{R_i^f}(X) \rightarrow [0, 1], i = 1, ..., m$. The *overall (or global) satisfaction degree* of a potential solution p_X is defined as: $\alpha(p_X) = \oplus \left\{\mu_{R^f}(p_X) | R^f \in C^f\right\}$, where $\oplus$ is an aggregation from $[0, 1]^m$ to $[0, 1]$.

DEFINITION 1. *A purchase requirement is defined as* $\lambda_{\mathcal{B}_{req}} = \bigcap \left\{ R_i^{c(\sigma_j^i)} \right\}$, *where* $R_i^{c(\sigma_j^i)}$ *is the induced crisp constraint at cut level* $\sigma_j^i \in [0, 1]$. *A purchase requirement may be formed by a subset of fuzzy constraints, where* $\text{cons}(\lambda_{\mathcal{B}_{req}})$ *denotes the set of constraints associated with each* $\lambda_{\mathcal{B}_{req}}$.

DEFINITION 2. *Given a* $\lambda_{\mathcal{B}_{req}}$, *the potential overall satisfaction degree (posd) the buyer gets if this requirement is met is* $\alpha^{\lambda_{\mathcal{B}_{req}}} = \oplus \left\{\sigma_j^i\right\}$, *where the cut level applied to constraints which are not included in the* $\lambda_{\mathcal{B}_{req}}$ *takes value* 1.

DEFINITION 3. *Given a* $\lambda_{\mathcal{B}_{req}}^k$ *defined at instant* k, *a concession strategy may be defined as a mechanism which generates a new* $\lambda_{\mathcal{B}_{req}}^{k+1}$ *such that* $\alpha^{\lambda_{\mathcal{B}_{req}}^{k+1}} < \alpha^{\lambda_{\mathcal{B}_{req}}^{k}}$, *where* $\alpha^{\lambda_{\mathcal{B}_{req}}^{k}} - \alpha^{\lambda_{\mathcal{B}_{req}}^{k+1}}$ *is minimized.*

DEFINITION 4. *Given a* $\lambda_{\mathcal{B}_{req}}^k$, *a trade-off strategy may be defined as a mechanism which generates a new* $\lambda_{\mathcal{B}_{req}}^{k+1}$ *such that* $\alpha^{\lambda_{\mathcal{B}_{req}}^{k+1}} \geq \alpha^{\lambda_{\mathcal{B}_{req}}^{k}}$ *and* $\lambda_{\mathcal{B}_{req}}^{k+1} \neq \lambda_{\mathcal{B}_{req}}^{j} | \forall j < k + 1$.

DEFINITION 5. *Given a* $\lambda_{\mathcal{B}_{req}}^k$, *a purchase requirement valuation is defined as a set* $v_{\mathcal{B}_{req}} = \{v_i | v_i \in [0, 1], i \in \text{cons}(\lambda_{\mathcal{B}_{req}})\}$, *where* v_i *represents the degree of importance that the constraint* i *has for the buyer agent.*

Let $\mathcal{S}_{req} = (S, N_s)$ be the requirement model of the seller agent, which is defined by the set of products the seller holds and the profit that the seller agent gets if the product is sold, $S = \{s_j | s_j = (p_j, u_j), p_j = (a_{j1}, ..., a_{jn}), 0 \leq j \leq k\}$, where k is the total number of products; and a negotiation

profile $N_s = \{\psi, \beta\}$, where $\psi \in \{0, 1\}$ controls if the seller agent expresses its preferences for a specific relaxation of the buyer's demands, and $\beta \in [0, 1]$ modulates the seller's attitude regarding a purchase requirement valuation.

DEFINITION 6. *A sale offer is defined as* a product p_j.

DEFINITION 7. *A relax requirement is defined as a set* $\rho_{\mathcal{B}_{req}} = \{r_i | r_i \in [0, 1], i \in \text{cons}(\lambda_{\mathcal{B}_{req}})\}$, *where* r_i *represents the preference for constraint* i *to be relaxed.*

3. THE NEGOTIATION DIALOGUE

Our purchase negotiation dialogue is a four stage sequence [2]: *open dialogue* (L1-2), *negotiate* (L3-8), *confirm* (L9-10) and *close dialogue* (L11). A negotiation will involve a seller participant P_s who joins any dialogue initiated by a potential buyer P_b. These are the main locutions:

L3: willing_to_sell(P_s, P_b, p_j) indicates to the buyer P_b a willingness to sell a product p_j.
L4: desire_to_buy$(P_b, P_s, \lambda^k_{\mathcal{B}_{req}})$ requests to purchase a product which satisfies $\lambda^k_{\mathcal{B}_{req}}$.
L5: prefer_to_sell$(P_s, P_b, \lambda^k_{\mathcal{B}_{req}}, \rho_{\mathcal{B}_{req}})$ requests to relax $\lambda^k_{\mathcal{B}_{req}}$, and expresses which constraints are preferred to be relaxed by means of $\rho_{\mathcal{B}_{req}}$.
L6: prefer_to_buy$(P_b, P_s, \lambda^k_{\mathcal{B}_{req}}, \upsilon_{\mathcal{B}_{req}})$ requests to purchase a product which satisfies $\lambda^k_{\mathcal{B}_{req}}$. A purchase requirement valuation $\upsilon_{\mathcal{B}_{req}}$ is used.
L7: refuse_to_buy(P_b, P_s, p_j) expresses a refusal to purchase a product p_j.
L8: refuse_to_sell$(P_s, P_b, p_j | \lambda_{\mathcal{B}_{req}})$ expresses a refusal to sell a product p_j or products which satisfy the purchase requirement $\lambda_{\mathcal{B}_{\text{req}}}$.
L9: agree_to_buy(P_b, P_s, p_j) and
L10: agree_to_sell(P_s, P_b, p_j) express a commitment to buy or sell a product p_j.

4. THE DECISION MECHANISMS

According to the participants (Buyers **B** or Sellers **S**):
B1: Recognize Need initiates a purchase dialogue by means of a user interface. **Outputs:** have_need(θ), where θ is a category of products.
B2: Generate Purchase Requirement generates a new purchase requirement. **Outputs:** $\emptyset$, $\lambda^k_{\mathcal{B}_{req}}$.

There are two basic strategies: **(a)** add a new constraint when the first $\lambda_{\mathcal{B}_{req}}$ is generated or when a sale offer cannot be accepted, or **(b)** relax $\lambda^k_{\mathcal{B}_{req}}$. An empty set $\emptyset$ is generated as output if none of these strategies can be used. In case (b), if the received locution is a refuse_to_sell(), we define a meta strategy which consists in applying when possible the trade-off strategy, and otherwise the concession one, so the loss of posd is minimized. If the received locution is a prefer_to_sell(), a cooperative strategy may be applied:

DEFINITION 8. *Given* $\rho^k_{\mathcal{B}_{req}}$ *as a response to a* $\lambda^k_{\mathcal{B}_{req}}$, *a cooperative strategy is a mechanism which generates a new* $\lambda^{k+1}_{\mathcal{B}_{req}cooperative}$ *as a function of* $\rho^k_{\mathcal{B}_{req}}$ and $\lambda^k_{\mathcal{B}_{req}}$, *which must integrate the demands of the seller agent.*

This strategy is useful when a buyer agent has several alternatives to generate a new purchase requirement. Taking into account the demands of the seller agent, the speed of convergence and the joint utility may be increased. Obviously, both parameters must be requirements for the buyer agent. We propose the following algorithm to relax a purchase requirement.

Algorithm 1 (1) Obtain the set $\{\alpha^{\lambda^{k+1}_{\mathcal{B}_{req}}}_{R^f_i} | \forall i \in \text{cons}(\lambda^{k+1}_{\mathcal{B}_{req}})\}$, where $\alpha^{\lambda^{k+1}_{\mathcal{B}_{req}}}_{R^f_i}$ is the *posd* when R^f_i is relaxed; (2) Compute $\alpha^{\lambda^{k+1}_{\mathcal{B}_{req}cooperative}} = \odot(\{\alpha^{\lambda^{k+1}_{\mathcal{B}_{req}}}_{R^f_i} | \forall i \in \text{cons}(\lambda^{k+1}_{\mathcal{B}_{req}})\})$, where $\odot$ is an aggregation operator; (3) Obtain $_{feasible}\{\lambda^{k+1}_{\mathcal{B}_{req}}\}$ as the set of elements which satisfy $\alpha^{\lambda^{k+1}_{\mathcal{B}_{req}}}_{R^f_i} \geq \alpha^{\lambda^{k+1}_{\mathcal{B}_{req}cooperative}}$; (4) Relax the constraint obtained by means of the csf function:

$$\arg\Big(\max_{feasible\{\lambda^{k+1}_{\mathcal{B}_{req}}\}} \alpha^{\lambda^{k+1}_{\mathcal{B}_{\text{req}}}}_{R^f_i} + r_i * \eta\Big)$$

B3: Generate Purchase Requirement Valuation obtains the valuation of a $\lambda_{\mathcal{B}_{req}}$. **Outputs:** $\emptyset$, $\upsilon_{\mathcal{B}_{req}}$.

If $\xi = 0$ this mechanism returns an empty set $\emptyset$. Otherwise, a valuation of the purchase requirement must be obtained. A high valuation implies that the buyer agent prefers not to relax the constraint.

Algorithm 2 (1) Given a $\lambda^{k+1}_{\mathcal{B}_{req}}$ then obtain a vector with the *posd* for all the possible future purchase requirements: $\{\alpha^{\lambda^{k+2}_{\mathcal{B}_{req}}}_{R^f_i} | \forall i \in \text{cons}(\lambda^{k+2}_{\mathcal{B}_{req}})\}$; (2) Then apply the purchase requirement valuation function $prvf : \{\alpha^{\lambda^{k+2}_{\mathcal{B}_{req}}}_{R^f_i} | \forall i \in \text{cons}(\lambda^{k+2}_{\mathcal{B}_{req}})\} \rightarrow \upsilon_{\mathcal{B}_{req}}$. It represents a general framework to obtain the valuation of $\lambda_{\mathcal{B}_{req}}$.
B4: Consider Offers decides at a particular whether to: **(a)** accept an offer, when $\alpha(p_j) \geq \alpha^{\lambda^k_{\mathcal{B}_{req}}}$; **(b)** reject an offer, when for some reason it does not fall within a previously submitted $\lambda_{\mathcal{B}_{req}}$; or **(c)** generate a new $\lambda_{\mathcal{B}_{req}}$ when it does not exceed the actual *posd*. **Outputs:** accept_offer(p_j), reject_offer(p_j), generate_purchase_requirement(p_j).
B5: Consider Withdrawal generates the withdraw(θ) output. **Outputs:** withdraw(θ).
S1: Recognize Category assesses whether the seller has products to sell. **Outputs:** enter(θ), not_to_enter(θ).
S2: Assess Purchase Requirement assesses whether a proposed $\lambda_{\mathcal{B}_{req}}$ can be satisfied. **Outputs:** $\emptyset$, $\lambda^k_{\mathcal{B}_{req}}$, p_j.

Algorithm 3 (1) The seller agent searches for products which satisfy the purchase requirement; (2) If it cannot be satisfied and parameter $\psi = 0$, an empty set $\emptyset$ is returned; (3) If it cannot be satisfied and parameter $\psi = 1$ then $\lambda^k_{\mathcal{B}_{req}}$ is returned; (4) From the products which satisfy the buyer's requirement, choose those with the highest utility and which has been a sale offer a minimum number of times.
S3: Generate Potential Sale Offers selects products which the seller agent considers as good candidates for a future sale offer. **Outputs:** set S_p of products.

If a seller agent cannot satisfy a $\lambda_{\mathcal{B}_{req}}$, it may encourage the buyer agent to change its proposals. A good alternative to promote a convergence in the negotiation is to express how $\lambda_{\mathcal{B}_{req}}$ should be relaxed. This mechanism treats the first task in this process: the selection of products which the seller agent would prefer to sell. We have identified two main aspects when making this selection: *local utility*, which depends on the u_j parameter; and *viability*, which depends on the similarity between the product p_j and $\lambda^k_{\mathcal{B}_{req}}$, as well as on an estimate of the sale offer p_j buyer's valuation. The β parameter modulates both aspects. So we define the *prefer*

function which measures the goodness of a sale offer in terms of utility and viability as:

$$prefer(p_j) = \beta * u_j + (1 - \beta) * \hat{viability}(p_j, \lambda^k_{\mathcal{B}_{req}}, \upsilon_{\mathcal{B}_{req}})$$

Algorithm 4 (1) Take all the products in the catalogue and compute the function *prefer*; (2) Fix a threshold and select those products with a *prefer* value exceeding this threshold. This is the set S_p.

S4: Generate Relax Requirement obtains the relax-requirement $\rho_{\mathcal{B}_{req}}$. **Outputs:** $\rho_{\mathcal{B}_{req}}$.

Given a set of potential sale-offers S_p, it generates a $\rho_{\mathcal{B}_{req}}$ in order to encourage the buyer agent to buy one of the products contained in S_p.

Algorithm 5 (1) For each potential sale-offer see which constraints from $\lambda_{\mathcal{B}_{req}}$ are not satisfied; (2) Generate $\rho_{\mathcal{B}_{req}}$, where $r_i = 1$ when constraint R^f_i has not been satisfied for any product, and otherwise $r_i = 0$.

S5: Accept or Reject Offer returns accept(p_j) if the sale-offer accepted by the buyer agent still exists and otherwise reject(p_j). **Outputs:** accept(p_j), reject(p_j).

S6: Consider Withdrawal generates the withdraw(θ) output when invoked. **Outputs:** withdraw(θ).

5. OPERATIONAL SEMANTICS

Operational semantics provides a formal linkage between the dialogue locutions and the semantic mechanisms previously defined. We use the notation in [2] which defines $\langle P_x, \mathbf{K}, s\rangle$ to denote the mechanism with number K and with output s of agent P_x. The transition rules are:

TR1: $\langle P_b, \mathbf{B1}, have\_need(\theta)\rangle \xrightarrow{\mathbf{L1}} \langle P_s, \mathbf{S1}, .\rangle$
TR2: $\langle P_s, \mathbf{S1}, not\_to\_enter(\theta)\rangle \rightarrow \langle P_s, \mathbf{S1}, .\rangle$
TR3: $\langle P_s, \mathbf{S1}, enter(\theta)\rangle \xrightarrow{\mathbf{L2}} \langle P_b, \mathbf{B2}, .\rangle$
TR4: $\langle P_b, \mathbf{B2}, \emptyset\rangle \rightarrow \langle P_b, \mathbf{B5}, .\rangle$
TR5: $\langle P_b, \mathbf{B5}, withdraw(\theta)\rangle \xrightarrow{\mathbf{L11}} \langle P_s, \mathbf{S6}, .\rangle$
TR6: $\langle P_s, \mathbf{S6}, withdraw(\theta)\rangle \xrightarrow{\mathbf{L11}} \langle P_b, \mathbf{B5}, .\rangle$
TR7: $\left\langle P_b, \mathbf{B2}, \lambda^k_{\mathcal{B}_{req}}\right\rangle \rightarrow \langle P_b, \mathbf{B3}, .\rangle$
TR8: $\langle P_b, \mathbf{B3}, \emptyset\rangle \xrightarrow{\mathbf{L4}} \langle P_s, \mathbf{S2}, .\rangle$
TR9: $\left\langle P_b, \mathbf{B3}, \upsilon_{\mathcal{B}_{req}}\right\rangle \xrightarrow{\mathbf{L6}} \langle P_s, \mathbf{S2}, .\rangle$
TR10: $\langle P_s, \mathbf{S2}, \emptyset\rangle \xrightarrow{\mathbf{L8}} \langle P_b, \mathbf{B2}, .\rangle$
TR11: $\langle P_s, \mathbf{S2}, saleoffer(p_j)\rangle \xrightarrow{\mathbf{L3}} \langle P_b, \mathbf{B4}, .\rangle$
TR12: $\left\langle P_s, \mathbf{S2}, purchasereq(\lambda^k_{\mathcal{B}_{req}})\right\rangle \rightarrow \langle P_s, \mathbf{S3}, .\rangle$
TR13: $\langle P_s, \mathbf{S3}, S_p\rangle \rightarrow \langle P_s, \mathbf{S4}, .\rangle$
TR14: $\left\langle P_s, \mathbf{S4}, \rho_{\mathcal{B}_{req}}\right\rangle \xrightarrow{\mathbf{L5}} \langle P_b, \mathbf{B2}, .\rangle$
TR15: $\langle P_b, \mathbf{B4}, generatepurchasereq(p_j)\rangle \rightarrow \langle P_b, \mathbf{B2}, .\rangle$
TR16: $\langle P_b, \mathbf{B4}, acceptoffer(p_j)\rangle \xrightarrow{\mathbf{L9}} \langle P_s, \mathbf{S5}, .\rangle$
TR17: $\langle P_b, \mathbf{B4}, rejectoffer(p_j)\rangle \xrightarrow{\mathbf{L7}} \langle P_s, \mathbf{S2}, .\rangle$
TR18: $\langle P_s, \mathbf{S5}, accept(p_j)\rangle \xrightarrow{\mathbf{L10}} \langle P_b, \mathbf{B5}, .\rangle$
TR19: $\langle P_s, \mathbf{S5}, reject(p_j)\rangle \xrightarrow{\mathbf{L8}} \langle P_b, \mathbf{B2}, .\rangle$

6. A PURCHASE SCENARIO

A buyer desires to buy a product θ such that: $X_\theta = \{price, brand\}$, $D_\theta = \{[0, 30000], \{A, ..., J\}\}$, and $\oplus = min$. Figure 1 shows the buyer's preferences in a colored plane. The seller's catalogue is shown as a set of stars, circles and triangles representing products which give the seller agent a profit of 1.0, 0.5 and 0.1, respectively. The line describes the progress of the negotiation for both the inexpressive (dotted line) and expressive strategies. The progress of the negotiation in the first steps is the same in both strategies because all the potential sale offers are far from the buyer requirements, so the seller indicates the same preferences in the relax requirement. However, when $\lambda_{\mathcal{B}_{req}}$ is near to $(17000, E)$ and $(15000, D)$, the seller agent indicates that the brand constraint should be relaxed. For the buyer agent it is the same to relax the price or the brand constraint, so it relaxes the brand. The aggregated utility of the outcome is 1.4, while in the inexpressive strategy is 0.5.

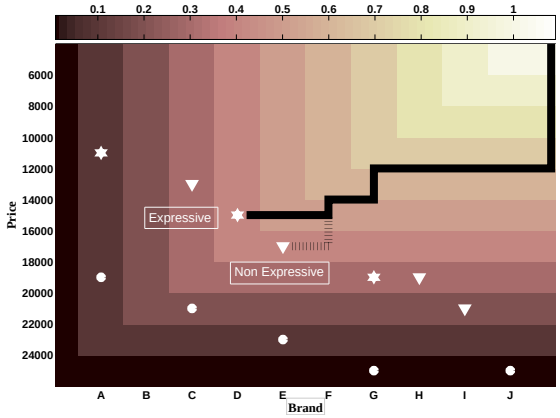


Figure 1: Expressive vs Inexpressive Strategies

7. CONCLUSIONS

This paper presents a model to support automated bilateral purchase negotiations using fuzzy constraints. All the internal decision mechanisms and the rules which govern interactions have been defined. In contrast to other fuzzy constraint models, in ours: (1) A buyer agent attends the seller's requirements in order to select the alternative from the set of trade-off proposals that is likely to benefit both agents. Typically this has been done employing an estimation of the similarity between an offer and the set of feasible counter-offers. The problem is that it is assumed that the potential offers are related among them, and this is not always the case. (2) Constraints can be valued in order to help the seller agent to make a more effective search. The purpose of this search is to select the more convenient potential sale offers in order to generate a balanced relax requirement. (3) Different attitudes for the seller and buyer agents can be easily modelled by means of the negotiation profiles.

8. ACKNOWLEDGEMENTS

This work has been supported by the Spanish Ministry of Science and Technology grant MCYT-TIC2003-09192-C11-05, and by the University of Alcalá grant UAH-PI-2003/001.

9. REFERENCES

[1] K. R. Lai and M. W. Lin. Modeling agent negotiation via fuzzy constraints in e-business. *Computational Intelligence*, 20(4):624–642, 2004.

[2] P. McBurney, R. M. V. Euk, S. Parsons, and L. Amgoud. A dialogue game protocol for agent purchase negotiations. *Autonomous Agents and Multi-Agent Systems*, 7:235–273, 2003.

[3] Xudong-Luo, N. R. Jennings, N. Shadbolt, Ho-Fung-Leung, and J. H. M. Lee. A fuzzy constraint based model for bilateral, multi-issue negotiations in semi-competitive environments. *Artificial Intelligence*, 148(1-2):53–102, 2003.

Symbolic Negotiation Revisited

Peep Küngas
Norwegian University of Science and Technology
Department of Computer and Information Science
Trondheim, Norway
peep@idi.ntnu.no

Mihhail Matskin
Royal Institute of Technology
Department of Microelectronics and Information Technology
Kista, Sweden
misha@imit.kth.se

ABSTRACT

In this paper we propose a formalism for symbolic negotiation. We regard symbolic negotiation as cooperative problem solving (CPS), which is based on symbolic reasoning and is extended with negotiation-specific rules. The underlying CPS formalism was previously presented in [3]. Here we extend the results and position symbolic negotiation according to other distributed problem solving mechanisms.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Multiagent systems; I.2.3 [Deduction and Theorem Proving]: Deduction; F.4.1 [Mathematical Logic]: Computational logic

General Terms

Algorithms, Theory

Keywords

Symbolic reasoning, partial deduction, cooperative problem solving, negotiation

1. INTRODUCTION

Symbolic negotiation is regarded in the field of computer science as a process, where parties try to reach an agreement on the high-level means for achieving their goals by applying symbolic reasoning techniques. Formalisation of symbolic negotiation could contribute both to the studies of human behaviour and multi-agent systems. In the former field the rational part of human reasoning could be modelled, while in the latter field self-organising systems with adaptive behaviour could be implemented.

Previously, Küngas and Matskin proposed [2] that symbolic negotiation could be formalised as partial deduction (PD) for linear logic (LL). In [3] a formalisation of PD for LL was presented and its soundness and completeness was proved. The paper regarded symbolic negotiation as interleaved cooperative problem solving (CPS) and plan modification. However, the paper formalised only the CPS part of the process. In this paper we extend the proposed CPS formalism with plan modification operators and then analyse symbolic negotiaton as a whole. We also formalise the coalition formation process and analyse its effect to CPS and symbolic negotiation. Anyway, due to the limited space we do not present proofs for propositions and theorems in this paper.

We would like to underline that from a computational point of view, we can regard CPS as AI planning and symbolic negotiation as plan reuse/repair. It has been shown [5] that from problem solving point of view in general neither planning from scratch nor plan repair has an advantage over each-other. Therefore we expect both CPS and symbolic negotiation to be computationally equivalent. Moreover, both CPS and symbolic negotiation lead to the same results as we prove in this paper.

However, compared to CPS, symbolic negotiation provides a more human-like way of problem solving, which can be more naturally followed by human participants. In addition, symbolic negotiation may encode a sort of search heuristics, which would make CPS computationally less demanding. These heuristics, however, are not discussed in this paper.

2. LINEAR LOGIC AND PARTIAL DEDUCTION

2.1 Linear logic

LL is a refinement of classical logic introduced by J.-Y. Girard to provide means for keeping track of "resources". In LL two assumptions of a propositional constant A are distinguished from a single assumption of A. This does not apply in classical logic, since there the truth value of a fact does not depend on the number of copies of the fact. Indeed, LL is not about truth, it is about computation.

We consider the !-Horn fragment of LL (HLL) [1] consisting of multiplicative conjunction ($\otimes$), linear implication ($\multimap$) and "of course" operator (!). In terms of resource acquisition the logical expression $A \otimes B \vdash C \otimes D$ means that resources C and D are obtainable only if both A and B are obtainable. After the sequent has been applied, A and B are consumed and C and D are produced.

While implication $A \multimap B$ as a computability statement clause in HLL could be applied only once, $!(A \multimap B)$ may be used an unbounded number of times. When $A \multimap B$ is

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

applied, then literal A becomes deleted from and B inserted to the current set of literals. If there is no literal A available, then the clause cannot be applied. In HLL ! cannot be applied to formulae other than linear implications.

In order to explain HLL and demonstrate some other features of LL, not presented in HLL, let us consider the following LL sequent from [4]—$(D \otimes D \otimes D \otimes D \otimes D) \vdash (H \otimes C \otimes (O \& S) \otimes !F \otimes (P \oplus I))$, which encodes a fixed price menu in a fast-food restaurant: for 5 dollars (D) you can get an hamburger (H), a coke (C), either onion soup O or salad S depending, which one *you* select, all the french fries (F) you can eat plus a pie (P) or an ice cream (I) depending on availability (restaurant owner selects for you). The formula $!F$ here means that we can use or generate a resource F as much as we want—the amount of the resource is unbounded.

2.2 Partial deduction

Here we describe basic definitions required to understand PD. Most of the definitions in this section are from [3], where they were used to define a CPS formalism. The presented rules have to read in bottom-up manner.

DEFINITION 1. *Resultant is a linear implication*

$$\vdash I \multimap_{\lambda a_1,\ldots,a_n.f} O, n \geq 0,$$

where f is a term representing a function, which generates O from I by applying potentially composite functions over $a_1, \ldots, a_n$.

DEFINITION 2. *Forward chaining PD step $\mathcal{R}_f(L_i)$ is defined as a rule*

$$\frac{B \otimes C \vdash G}{A \otimes C \vdash G}\ \mathcal{R}_f(L_i)$$

where L_i is a labeling of linear implication $\vdash A \multimap_{L_i} B$. A, B, C and G are multiplicative conjunctions.

DEFINITION 3. *Backward chaining PD step $\mathcal{R}_b(L_i)$ is defined as a rule*

$$\frac{S \vdash A \otimes C}{S \vdash B \otimes C}\ \mathcal{R}_b(L_i)$$

where L_i is a labeling of linear implication $\vdash A \multimap_{L_i} B$. A, B, C and S are multiplicative conjunctions.

PD steps $\mathcal{R}_f(L_i)$ and $\mathcal{R}_b(L_i)$, respectively, apply L_i in forward and backward chaining manner.

3. AGENTS AND COALITION FORMATION

3.1 Agent representation

An agent is presented with the following HLL sequent:

$$\Gamma; S \vdash G,$$

where Γ is a set of agent capabilities, S is the initial state and G is the goal state of the agent. Both S and G are multiplicative conjunctions of HLL formulae.

Every element of Γ is a linear implication $\vdash I \multimap O$, where I and O are formulae which are, respectively, consumed and generated when a particular capability is applied. It has to be mentioned that a capability can be applied only, if conjuncts in I form a subset of conjuncts in S. It should be also underlined that in order to achieve their goals, agents have to construct (and then execute) the following plan from the elements of Γ:

$$\vdash S \multimap G.$$

Each agent is assumed to have a single task only. Whenever in an application an agent is supposed to have several task, it would be modelled in our framework as several agents, each having a single task.

DEFINITION 4. *Agent's task is a resultant in PD.*

Modifications of an agent's task are derived through PD. We consider all possible derivations of a task as instances of the same task. Moreover, we constrain an agent to consider only one derivation at time during distributed CPS. This would ensure that instances of the same task would not be solved concurrently.

DEFINITION 5. *Resource is a formula of HLL.*

3.2 Agent coalitions

The purpose of agent coalition formation in our formalism is to merge several tasks into a larger one. Then agents become able to perform tasks, which they would not have been able to perform individually. Thereby coalition formation could be seen as a way to optimize problem solving globally.

DEFINITION 6. *Agent coalition $\mathcal{C}$ is a structure $\langle \mathcal{A}, \mathcal{T}, \mathcal{R}, \triangleright, [] \rangle$, where $\mathcal{A}$ is a set of participanting agents, $\mathcal{T}$ is a coalition task and $\mathcal{R}$ is a set of resources shared between agents. $\triangleright$ is a mapping from agents to resources ($\triangleright : \mathcal{A} \times \mathcal{A} \rightarrow \mathcal{R}^*$), which one agent would deliver to another if a solution for a coalition task is found. Finally, $[]$ is a mapping from a coalition task to agents in the coalition ($[] : \mathcal{T} \rightarrow \mathcal{A}^+$), which agreed to solve the task together by sharing their resources.*

We interpret a coalition as an agreement between agents to solve their tasks together. In a coalition agents can also exchange resources directly between themselves. Additionally it should be emphasised that we model tasks as (composite) actions consuming and producing resources. Coalitions are formed by applying rule *merge*.

DEFINITION 7 (COALITION FORMATION RULE). *Coalition formation rule merge is defined as follows:*

$$\frac{\vdash A \otimes B \multimap C \otimes D}{\vdash A \multimap C \quad \vdash B \multimap D}\ \textit{merge}$$

Rule *merge* combines 2 tasks X and Y to create a task with the intended meaning: "task X could be performed by one agent, if another agent performs task Y" and vice versa. From a distributed theorem proving point of view it means that tasks X and Y are combined together and their search spaces are merged. If a coalition includes all agents of a system, then problem solving is seen as solving a single task in a distributed manner. This contrasts with our general case, where multiple tasks are solved concurrently in a distributed way.

PROPOSITION 1. *Solution of a coalition task $\mathcal{T}$, which was achieved through PD, solves the tasks of all coalition members $[\mathcal{T}]$.*

DEFINITION 8 (SOUNDNESS OF COAL. FORMATION). *If a solution of coalition task $\mathcal{T}$ solves the tasks of all participating agents $[\mathcal{T}]$, then coalition formation is sound.*

PROPOSITION 2. *Coalition formation is sound with respect to PD.*

Coalition formation could be seen from problem solving point of view as merging several tasks into one. Thus instead of solving multiple tasks concurrently, with coalitions we would solve a composed task.

PROPOSITION 3. *If a set of tasks included in a coalition task is solvable through PD independently, then so is the composed coalition task.*

DEFINITION 9 (COMPLETENESS OF COAL. FORMATION). *If agents' tasks were solvable individually and their coalition task is solvable as well, then coalition formation is complete.*

PROPOSITION 4. *Coalition formation is complete with respect to PD.*

DEFINITION 10. *A coalition task is implicitly solved, if no PD steps are required to solve the coalition task.*

4. SYMBOLIC NEGOTIATION

In this section we define the symbolic negotiation process with respect to PD. We define also symbolic negotiation rules for operating over forward (Θ_f) and backward chaining (Θ_b). Additionally some other negotiation-specific definitions are given. Finally, we prove that symbolic negotiation is sound and complete.

DEFINITION 11. *Negotiation rule $\Theta_f(a_k/a_i)$ is defined as the the following operational rule:*

$$\frac{\begin{array}{c}\vdash ((((I \setminus O(a_i)) \cup I(a_i)) \setminus I(a_k)) \cup O(a_k)) \\ \multimap_{a_1,\ldots,a_{i-1},a_k,a_{i+1},\ldots,a_n} O\end{array}}{\begin{array}{c}\vdash I \multimap_{a_1,\ldots,a_{i-1},a_i,a_{i+1},\ldots,a_n} O \\ I(a_k) \subseteq I(a_i) \quad O(a_i) \subseteq O(a_k)\end{array}} \Theta_f(a_k/a_i)$$

DEFINITION 12. *Negotiation rule $\Theta_b(a_k/a_i)$ is defined as the following operational rule:*

$$\frac{\begin{array}{c}\vdash I \multimap_{a_1,\ldots,a_{i-1},a_k,a_{i+1},\ldots,a_n} \\ ((((O \setminus I(a_i)) \cup O(a_i)) \setminus O(a_k)) \cup I(a_k))\end{array}}{\begin{array}{c}\vdash I \multimap_{a_1,\ldots,a_{i-1},a_i,a_{i+1},\ldots,a_n} O \\ I(a_k) \subseteq I(a_i) \quad O(a_i) \subseteq O(a_k)\end{array}} \Theta_b(a_k/a_i)$$

$I(a)$ and $O(a)$ in the preceding rules represent respectively resources required to apply and achieved by applying capability a. I represents resources available for solving a task while O represents resources, which must be achieved by the task. The preceding negotiation rules replace one PD step application in a task derivation with another. Rule $\Theta_f(a_k/a_i)$ replaces forward chaining steps while $\Theta_b(a_k/a_i)$ replaces backward chaining steps. The symbolic negotiation rules replace a capability in the derivation with another capability which consumed the same amount or less and generated the same amount or more resources than the previous capability. Therefore negotiation rules allow to optimise a task of agent X by agent Y. This means that some PD steps of agent X in its task are substituted by Y, if associated capabilities are more efficient in terms of resource consumption and generation.

DEFINITION 13 (SYMBOLIC NEGOTIATION). *Symbolic negotiation is PD with coalition formation and symbolic negotiation rules.*

DEFINITION 14 (CPS). *Cooperative problem solving is PD with coalition formation.*

DEFINITION 15 (COMPUTATIONAL EQUIVALENCE). *Two processes are computationally equivalent with respect to PD, if they would construct the same set of resultants.*

THEOREM 1. *CPS and symbolic negotiation are computationally equivalent.*

According to Theorem 1 symbolic negotiation could be view as a special case of CPS, where PD is guided at operational level.

DEFINITION 16. *Symbolic negotiation is sound, if it is sound with respect to PD, and all resultants achieved through symbolic negotiation are achievable through CPS.*

DEFINITION 17. *Symbolic negotiation is complete, if it is complete with respect to PD, and all resultants achieved through CPS are achievable through symbolic negotiation.*

THEOREM 2 (SOUNDNESS OF SYMBOLIC NEGOTIATION). *Symbolic negotiation is sound.*

THEOREM 3 (COMPLETENESS OF SYMB. NEGOTIATION). *Symbolic negotiation is complete.*

Soundness and completeness of our CPS formalism are directly derived from Proposition 2, Proposition 4 and completeness/soundness results of PD presented in [3].

Acknowledgements

This work was partially supported by the Norwegian Research Foundation in the framework of Information and Communication Technology (IKT-2010) program—the ADIS project.

5. REFERENCES

[1] M. I. Kanovich. Linear logic as a logic of computations. *Annals of Pure and Applied Logic*, 67:183–212, 1994.

[2] P. Küngas and M. Matskin. Linear logic, partial deduction and cooperative problem solving. In *Proceedings of the First International Workshop on Declarative Agent Languages and Technologies (in conjunction with AAMAS 2003), DALT'2003, Melbourne, Australia, July 15, 2003*, volume 2990 of *Lecture Notes in Artificial Intelligence*, pages 263–279. Springer-Verlag, 2004.

[3] P. Küngas and M. Matskin. Partial deduction for linear logic—the symbolic negotiation perspective. In *Proceedings of the Second International Workshop on Declarative Agent Languages and Technologies (in conjunction with AAMAS 2004), DALT'2004, New York, USA, July 19, 2004*, volume 3476 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2005.

[4] P. Lincoln. Linear logic. *ACM SIGACT Notices*, 23(2):29–37, 1992.

[5] B. Nebel and J. Koehler. Plan reuse versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence*, 76:427–454, 1995.

A Multi-Issue Negotiation Protocol among Competitive Agents and its Extension to A Nonlinear Utility Negotiation Protocol

Takayuki Ito* and Mark Klein
Center for Coordination Science, Sloan School of Management,
Massachusetts Institute of Technology
3 Cambridge Center, NE20-336, Cambridge, MA 02142
{takayuki,m_klein}@mit.edu

ABSTRACT

Multi-issue negotiation mechanisms have been studied very widely and represent a promising eld because most negotiations in the real world are multi-issue negotiations. We model multi-issue negotiation with competitive agents. This re ects the real world in which people have to make agreements on constraints. A combinatorial winner determination mechanism is employed for solving a constraint satisfaction problem with competitive agents. The features of our protocol include Pareto e ciency on constraints. Our experimental results show that our proposed protocol is scalable with the number of agents, and that a heuristic method can achieve good utility. The above protocol is designed for linear utility domains. We also extend our protocol so that we can handle nonlinear utilities.

Categories and Subject Descriptors

I.2.11 [**Distributed Arti cial Intelligence**]: Multiagent systems; I.2.11 [**Distributed Arti cial Intelligence**]: Coherence and coordination; H.4.1 [**Office Automation**]: Groupware

General Terms

Algorithms

Keywords

Multi-issue negotiation, Constraint Satisfaction, Nonlinear Utility

1. INTRODUCTION

Multi-Issue Negotiation mechanisms have been studied very widely[1][7][3][2][6] and represent a promising eld because most negotiations in the real world are multi-issue negotiations. In particular, in real world, issues are constrained with respect to each other. Also, even in collaborative situation, to get an agreement, agents need to act competitively because of their self-interested nature.

In multi-issue negotiations, agents need to nd a solution that consists of concrete issue values and concrete constraints. While this is frequent in real world negotiation, existing constraint satisfaction research does not address the notion of candidates for constraints. In this paper, we assume agents are competitive in the sense that they try to get an agreement that can give them a higher utility. The multistage bargaining protocol we proposed here instantiates constraints by using a combinatorial auction-like protocol, and then determining the issue values.

Most existing work also assumes that agents are totally collaborative. Our work focuses on mainly competitive agents. The competitive assumption can hold in many situation in the real world. Even in the collaborative negotiation, we believe that there are competitive situations. If designers are in one of the engine companies, for example, they have to competitively negotiate to make a decision in terms of the engine.

In addition, we extend our protocol so that it can handle nonlinear utility domains[4][5]. For the linear utility domain, we allow agents to aggregate the utilities of the issue-values by simple linear summation. However, in the real world, such aggregations are unrealistic. For example, we cannot just add up the value of a car's tyre and the value of a car's engine. First, we do not have any guarantees that such utilities can be added up. Second, there can be another constraint among the issues. For example, there can be a constraint between a car's color and engine. If a sports car does not have a gray body variant and there are only engines for sports cars, then we cannot select a gray body and a sports-type engine. In the real world, such constrains can exist in any situation. Therefore, in this paper, we extend our protocol to handle nonlinear utilities.

The rest of the paper is organized as follows. First we propose a bargaining protocol that achieves a desirable solution in multiple issue negotiations. Second we demonstrate the experimental results for a simple example. Then, we extend our protocol so that the protocol can handle nonlinear utilities. Finally concluding remarks are given in the nal section.

*Visiting from Nagoya Institute of Technology, Japan

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS2006 Hakodate, Hokkaido, Japan
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. A COMPETITIVE COMBINATORIAL OPTIMZIATION NEGOTIATION

2.1 Multiple Issues Negotiations

A set of agents $a_1, a_2, \ldots, a_l$. Multiple issues, and constraints among issues.Each agent a_i has an evaluation value function $v_{a_i}(x_k)$ on each issue value x_k and an evaluation value function $v_{a_i}(c_k)$ on constraints. Agent a_i's utility is de ned as $u_{a_i} = \sum v_{a_i}(x_k) + \sum v_{a_i}(c_k) \quad c$, where $\sum v_{a_i}$ means the total of the evaluation values for issue values and constraints, and c is the payment for the auction. Agents try to maximize their utility. In this paper, we assume agents are competitive. This means that we cannot compute the entire optimal social welfare by making all agents reveal their utility functions.

2.2 Negotiation Mechanism

We adopt auction-like negotiations to determine constraints and issue values since agents are competitive.

(Stage 1 : Determining the constraints.) Each agent a_i makes bids on bundles of constraint instances. A bundle consists of one or more constraint instances that do not con- ict with each other. The mechanism selects a combination of bundles of constraint instances that maximizes social welfare i.e. maximizes the utilities of the accepted constraints. The mechanism announces the accepted constraints.

(Stage 2 : Determining the issue values.) Players make bids on the values of the issues. Auctions are held for each issues separately. We employ a 2nd price auction. The mechanism selects the combination of issue values that maximize social welfare for the issue values and satis es all constraints.

In this protocol, agents rst determine the constraints, and then determine issue values. In stage 1, a combinatorial auction maximizes social welfare with respect to constraints. In stage 2, a 2nd price auction maximizes social welfare with respect to issue values.

The proposed protocol is Pareto E cient. This is because the constraints are selected using a combinatorial optimization that maximizes social welfare in terms of the constraints. Then, the maximum utility issue values are selected so that these values can satisfy the selected constraints.

We propose a winner-based value assignment method for our negotiation protocol. Utilities on constraints can be related to utilities on issue values since satisfying a constraint totally depends on the issue values. We can expect that a user has high utilities for issue values that satisfy the related constraints. This means that a winner of constraints in stage 1 may have high utilities for the issue values that satisfy the constraints. Using this observation, we heuristically modi ed the stage 2 in the protocol as follows:

(Stage 2' : Determining the issue values by stage 1 winners.) Winners make bids on the values of the issues. Auctions are held for each issue separately. We employ a 2nd price auction. The mechanism selects a combination of issue values that maximizes the winners' social welfare with respect to issue values and satis es all constraints. Since the original stage 2 is a combinatorial optimization problem, it requires substantial computation power. The winner-based value assignment method can e ectively reduce this computation cost.

2.3 An Example

Here, we present an example of our negotiation protocol. Figure 1 outlines the problem. There are two constraints and three issues. Each constraint has two instances, $\{>, <\}$. Each issue has issue values, $\{1, 2, \ldots, 20\}$.

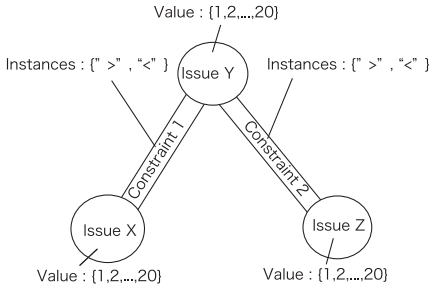


Figure 1: An example of the proposed protocol

We assume there are three agents. Also, we assume one coordinator agent who holds auctions and determines payments. In stage 1, each agent submits combinatorial bids on bundles of constraint instances. Each agent must solve a combinatorial problem to compute consistent bundles that do not con ict with each other. In this example, possible bundles include $\{<,\}, \{>,\}, \{,<\}, \{,>\}, \{>,>\}, \{<,>\}, \{>,<\}, \{<,<\}$. Each agent submits one or more bundles with their evaluation values.

Based on these submitted bundles, the coordinator agent searches for the combination of bundles that maximize social welfare and do not con ict with each other. The social welfare of a bundle is the sum of the evaluation values for the constraints in the bundle. We say there is a con ict in a bundle when there is no solution for the constraints in a bundle. The winners are determined.

The coordinator agent determines the payments for each winner, and announces the accepted constraint instances. For this example, we assume that the bundle $\{<,<\}$ is accepted.

For each issue, agents submit maximum evaluation values for issue values that satisfy the accepted constraints $\{<,<\}$. We do not need to assume agent's evaluation values are linear over issue values. For example, one agent can have the following evaluation values : \$20 for issue value 1, \$30 for issue value 2, and \$20 for issue value 3. Each agent searches for the maximum utility combination of evaluation values, and submits it.

The coordinator agent holds a second price auction for each issue, and determines the winner and the payment. This auction is strategy proof since the winner cannot manipulate his payment. Finally, the decided issue values and constraint instances are announced to all agents.

3. EXPERIMENTAL RESULTS

3.1 Definition of Agent's evaluation values

We conducted experiments using the example in the previous section. The negotiation problem is shown in Figure 1. We observed how the proposed negotiation protocols can obtain social surpluses when varying the number of agents.

For each experiment, we de ned the agent's evaluation values. There are 20 issue values and the maximum evaluation value is 10. We assume that each agent's evaluation value function has a single peak. The height of the peak

shows the maximum evaluation value for that issue. Each agent has evaluation value functions for each issue.

Evaluation values for constraints are de ned using the following equation: $v_{a_i}(c_k) = \max_S \sum_{x_{kj} \in S} u_{a_i}(x_{kj})$, where S is a set of values that satisfy constraint c_k. This shows that our domain is a linear utility domain. We describe how to extend it to nonlinear utility domains in the next section.

Figure 2 shows experimental results for a comparison in a competitive case. The vertical axis shows the obtained social surplus. The horizontal axis shows the number of agents. Even though the number of agents is large, we could determine winners since we assume just 2 items. The "CA-based protocol" is the combinatorial optimization based negotiation protocol proposed in section 2. The "CA & Winner-based Protocol" is the combinatorial optimization based negotiation protocol with the winner-based assignment method proposed From this graph, we can see that the CA & Winner-based protocol can work well even when the number of agents is 100; it obtains about %80 of the social surplus of the CA-based protocol.

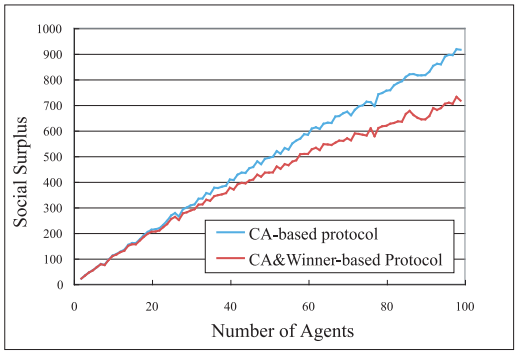


Figure 2: Experimental Results

4. A STEP TOWARD A NON-LINEAR UTILITY NEGOTIATION PROTOCOL

In the previous sections, we proposed a negotiation protocol for linear utility domains. In linear utility domains, we can sum up utilities among issues and constraints. However, in the real world, such aggregations are problematic. We cannot de ne the sum of the utilities for a car's color and a car's engine. Also, in the real world, we cannot assume that issues are independent. Multiple issues are almost always interdependent. In the previous sections, we assume that each agent calculates his utility for a constraint as the sum of the issue values that satisfy the constraint. This makes our domain a linear utility domain. In this section we relax this assumption so that our protocol can be applied to nonlinear utility domains.

First we slightly change the domain setting. A utility function is given that maps a contract (i.e. a set of issue values) to a utility value for each agent. Every agent has a set of constraints it wants to be satis ed. They can be binary constraints (e.g. $X > Y$) as well as unary constraints (e.g. $x = 3$). The binary constraints are what create the nonlinearity. Every constraint has a weight W. The utility of a constraint is W if the constraint is satis ed, and 0 if the constraint is not satis ed. The utility of a contract is the sum of the weights for every constraint that is satis ed by that contract The point is that the weights of the constrains are now the aggregation of the utilities of the issues.

Based on the above setting, the negotiation protocol can be modi ed as follows: A bid is a set of constraints. Agents calculate the utility of a bid by rst nding all contracts that satisfy all the constraints in the bid, and then returning the value of the best contract in that set. Agents conduct the VCG auction. Since each agent has to solve a constraint satisfaction problem, the computational cost is not low. However, we can handle nonlinear utility using this protocol.

The above protocol can generate multiple-optima utility functions that have valleys (for contracts that satisfy few or none of the constraints) as well as hills and mountains (for contracts that satisfy some or all of the constraints). Note that it is possible, even with this approach, to get a single optimum utility function, just depending on the particular constraints that apply for a given situation. But if we have a large enough space of possible contracts, as well as enough higher-order (binary or higher) constraints, it is likely that there will be multiple optima. In this case, in order to get the optimal contract, we have to have a method suitable for nonlinear optimization, e.g., stochastic search. In future work, we are going to construct such a negotiation protocol with competitive agents.

5. CONCLUSIONS

In this paper, we proposed an combinatorial optimization-based negotiation protocol among competitive agents. First we modeled a multi-issue negotiation with competitive agents. We proposed a variant of the auction-like negotiation protocol, in which we focus only on winners to reduce the size of the search space. The experimental results show the winner-based negotiation protocol can achieve good social surplus. In addition, we investigated the nonlinear utility model. The nonlinear utility model can represent more realistic negotiations. As a future work we are now investigating the nature of nonlinear utility negotiations.

6. REFERENCES

[1] M. Barbuceanu and W.-K. Lo. Multi-attribute utility theoretic negotiation for electronic commerce. In *Proc of AMEC2000*, 2000.

[2] S. S. Fatima, M. Wooldridge, and N. R. Jennings. An agenda-based framework for multi-issue negotiation. *Artificial Intelligence*, 152:1–45, 2004.

[3] C. M. Jonker and V. Robu. Automated multi-attribute negotiation with e cient use of incomplete prefenrece information. In *Proc. of AAMAS2004*, 2004.

[4] M. Klein, P. Faratin, H. Sayama, and Y. Bar-Yam. Negotiating complex contracts. *Group Decision and Negotiation*, 12(2):58–73, 2003.

[5] R. J. Lin and S. T. Chou. Bilateral multi-issue negotiations in a dynamic environment. In *Proc. of AMEC V*, 2003.

[6] X. Luo, N. R. Jennings, N. Shadbolt, H. fung Leung, and J. H. man Lee. A fuzzy constraint based model for bilateral, multi-issue negotiations in semi-competitive environments. *Artificial Intelligence*, 148:53–102, 2003.

[7] L.-K. Soh and X. Li. Adaptive, con dence-based multiagent negotiation strategy. In *Proc. of AAMAS2004*, 2004.

Robotics

Multi-Robot Learning with Particle Swarm Optimization

Jim Pugh and Alcherio Martinoli
Swarm-Intelligent Systems Group
École Polytechnique Fédérale de Lausanne
1015 Lausanne, Switzerland
{jim.pugh,alcherio.martinoli}@epfl.ch

ABSTRACT

We apply an adapted version of Particle Swarm Optimization to distributed unsupervised robotic learning in groups of robots with only local information. The performance of the learning technique for a simple task is compared across robot groups of various sizes, with the maximum group size allowing each robot to individually contain and manage a single PSO particle. Different PSO neighborhoods based on limitations of real robotic communication are tested in this scenario, and the effect of varying communication power is explored. The algorithms are then applied to a group learning scenario to explore their susceptibility to the credit assignment problem. Results are discussed and future work is proposed.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics—*Autonomous vehicles*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Experimentation

Keywords

particle swarm optimization, unsupervised learning, multi-robot systems

1. INTRODUCTION

Designing even simple behaviors for robots that are efficient and robust can be very difficult for humans; it is often not hard to implement a rudimentary controller that accomplishes the task, but achieving optimal performance can be very challenging. Unsupervised robotic learning allows for automated design of efficient, robust controllers, which saves much design time and effort. Unsupervised learning is also useful for allowing robots to adapt to situations where the task/environment is unknown beforehand or is constantly changing.

Genetic Algorithms (GAs) are a very common method of accomplishing machine learning and optimization. Candidate solutions to a problem are modeled as members of a population, and breeding (selection and crossover) and mutation are applied to "parents" (high performing solutions) in the population to generate "children" (new candidate solutions). GA can be used to shape an Artificial Neural Network (ANN) controller by using the parameter set as the weights, and the evaluative function as a measure of the performance of a desired robot behavior.

Particle Swarm Optimization (PSO) is a promising new optimization technique which models a set of potential problem solutions as a swarm of particles moving about in a virtual search space. The method was inspired by the movement of flocking birds and their interactions with their neighbors in the group. PSO can also be used to evolve ANN robotic controllers.

Both GA and PSO use groups of interacting virtual agents in order to achieve their optimization. In collective robotics, groups of robots interact to accomplish their goals. It may therefore be possible to implement these algorithms in a parallel distributed fashion for learning in multi-robot systems. Each robot would be responsible for several virtual agents, which it would need to evaluate at each iteration. After each set of evaluations, the robots would communicate to share the fitness information needed to progress to the next iteration of the algorithm. By running the algorithms in this fashion, we would need no external supervisor to oversee the learning process, and the speed of learning could be significantly improved, as many robots evaluating in parallel would decrease the number of required controller evaluations and therefore decrease the total learning time.

In the local neighborhood version of PSO, each particle only needs to be aware of the state of a small subset of particles in the population in order to update itself at each iteration. It may therefore be possible to implement PSO in a distributed manner where communication from any given node would only be necessary with several other nodes, making it a very scalable parallel approach. In contrast, the GA population manager must have knowledge of the entire population in order to implement standard breeding techniques, which prevents the same scalable technique from being applied.

In this paper, we explore the effectiveness of using a modified version of PSO on groups of realistically simulated robots performing distributed unsupervised learning. At the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

maximum group size, the number of robots is set equal to the number of particles in the PSO population, allowing each robot in the group to manage a single unique particle. We test how the performance is affected if we adapt the standard PSO neighborhood structure to more closely model what is possible in a real robot group with limited communication abilities. Section 2 provides some background on GA, PSO, unsupervised robotic learning, and multi-robot learning. Section 3 examines how the effectiveness of distributed unsupervised learning is affected by the number of robots in the group. Section 4 analyzes how the learning performance is affected by different neighborhood structures based on the limitations of robotic communication when each robot contains a single particle. Section 5 focuses on one such neighborhood structure and tests the effect of varying the communication range of the robots. Section 6 applies the algorithms to a group learning task, to see how affected they are by the credit assignment problem and to see how the communication-based neighborhoods fare in different scenarios. Section 7 discusses the implications of the results and suggests future work, and Section 8 concludes.

2. BACKGROUND

Genetic algorithms were originally developed in the 1960s by John Holland. The algorithms are inspired by evolution, where the fittest members of a population tend to reproduce more often than the less fit members. Candidate solutions are modeled as a population of "chromosomes". At each iteration of the algorithm, a new population is generated from the previous one. Selection of the parents of the new generation is implemented using one or more of several schemes, such as elitism (using only the top performing members of the population), Roulette Wheel sampling (stochastically choosing parents with weight proportional to performance), and rank selection (ranking chromosomes from best to worst and stochastically choosing parents with weight proportional to the rank). After parents have been chosen, crossover between the parents can occur with some probability (each chromosome is split into two, and children use one part from one parent and the other part from the other). This allows positive aspects from different chromosomes to be merged into a single chromosome. Last, mutation is applied, where each element of the chromosome may have its value randomly changed with some probability. This provides a random local search, which allows solutions to continue to improve beyond the genetic diversity that was available in the original population ([6], [15]).

The original PSO method was developed by James Kennedy and Russel Eberhart ([9], [3]). Every particle in the population begins with a randomized position ($x_{i,j}$) and randomized velocity ($v_{i,j}$) in the n-dimensional search space, where i represents the particle index and j represents the dimension in the search space. Candidate solutions are optimized by flying the particles through the virtual space, with attraction to positions in the space that yielded the best results. Each particle remembers the position at which it achieved its highest performance ($x^*_{i,j}$). Each particle is also a member of some neighborhood of particles, and remembers which particle achieved the best overall position in that neighborhood (given by the index i'). This neighborhood can either be a subset of the particles (local neighborhood), or all the particles (global neighborhood). For local neighborhoods, the standard method is to set neighbors in a pre-defined way (such as using particles with the closest array indices as neighbors modulo the size of the population, henceforth known as a "ring topology") regardless of the particles' positions in the search space. The equations executed by PSO at each step of the algorithm are

$$\begin{aligned} v_{i,j} &= w \cdot (v_{i,j} + pw \cdot rand() \cdot (x^*_{i,j} - x_{i,j}) \\ &\quad + nw \cdot rand() \cdot (x^*_{i',j} - x_{i,j})) \\ x_{i,j} &= x_{i,j} + v_{i,j} \end{aligned}$$

where w is the inertia coefficient which slows velocity over time, pw is the weight given to the attraction to the previous best location of the current particle and nw is the weight given to the attraction to the previous best location of the particle neighborhood. $rand()$ is a uniformly-distributed random number in $[0, 1]$.

PSO has been shown to perform as well as or better than GA in several instances. Eberhart and Kennedy found PSO performs on par with GA on the Schaffer f6 function [3, 9]. In work by Kennedy and Spears [10], a version of PSO outperforms GA in a factorial time-series experiment. Fourie showed that PSO appears to outperform GA in optimizing several standard size and shape design problems [5].

Unsupervised learning describes learning scenarios where there is no external entity which decides upon the training set inputs for the learning agent(s). Rather, inputs are generated dynamically as the agents interact with their environment. This is as opposed to supervised learning, where the inputs are generated/collected first and then used repeatedly. In supervised learning, the accuracy of the system at each iteration is usually decided by an external "teacher" evaluating the system output. The pre-defined inputs are split into two separate sets, one for training the system and the other for testing the performance. Supervised learning tends to be easier than unsupervised, as the data does not change between iterations of the algorithm and can be preselected to avoid using unusual or particularly noisy data points. However, supervised learning is not possible in situations where the input data to the system depends on the current state of the learning agent; this is the case for online robotic learning, since the robot's movements affect what its sensors will perceive.

Evolutionary algorithms have been used extensively for unsupervised learning of robotic behavior. A good survey of the work is given in [12]. More specifically, standard GA has been shown to be effective in evolving simple robotic controllers [4], and modified noise-resistant versions of both GA and PSO were shown to achieve very good performance on simulated unsupervised robotic learning, outperforming the standard versions of the algorithms [18].

In collective robotics, many desired behaviors result in strong interactions between robots in the group, and the actions of one robot can significantly impact the performance of another. If a behavior is being learned and each robot is evaluating a different controller, this can give rise to the "credit assignment" problem, where robots do not know whether a good/bad fitness score was due to its own performance or to that of other robots. This effect can be particularly pronounced in cases where robots do not explicitly share their intentions through communication channels and can severely hamper the learning process. The credit assignment problem can arise in two different scenarios. The

first is when robots are learning individual behaviors, but the performance of their behavior can be impacted by the actions of other robots in the group, causing an inaccurate fitness evaluation. The second is when robots are learning a group behavior with a single collective fitness. Often, it is not easy to decompose the fitness value to know which robots positively affected the performance and which didn't. An elegant way to bypass the credit assignment problem in these cases is to use "homogenous" learning (as opposed to "heterogeneous" learning), where all the robots simultaneously evaluate the same controller and therefore contribute equally to the collective score on average. While this drastically slows the learning process, it is the only way of achieving good results in some highly stochastic scenarios, such as those investigated in Hayes et al. [7].

Multi-robot learning has been used and explored in various ways. Matarić studied mechanisms to encourage individual agents in a group to act in ways to help the group performance [11]. Multi-robot learning using several methods in a wide variety of scenarios has been explored ([2], [20]). Techniques for increasing individual learning speed via multi-robot learning were studied in [8] and [13]. A modified version of a genetic algorithm has been embedded onto a 2-robot system to allow for distributed parallel learning [17]. Particle swarm optimization has thus far not been used for learning in scalable multi-robot systems.

3. VARYING THE ROBOTIC GROUP SIZE

In Pugh et al. [18], unsupervised learning was used to teach robots obstacle avoidance behavior for both a single robot and two robots co-learning. We wish to expand this to test unsupervised learning on much larger robotic groups, where the evolving candidate solutions are distributed throughout the group to achieve faster learning.

3.1 Experimental Setup

We use the noise-resistant GA and PSO algorithms from [18]. GA uses elitism to select the best half of the population as the parent set, and then applies Roulette Wheel sampling to replenish the missing chromosomes. PSO uses a local neighborhood in a ring topology with one neighbor on each side. At every iteration, these algorithms reevaluate their previous best locations and parent sets for PSO and GA, respectively, combining the new fitness value with previous ones to get a more accurate measure of the actual fitness. Although this requires twice as many fitness evaluations at each iteration as their standard counterparts, this technique prevents noisy fitness evaluations from severely disrupting the learning process and gives much better results given the same amount of computational time.

We modify the noise-resistant PSO algorithm from its original form slightly: when updating the neighborhood best particle for particle i ($x^*_{i'}$), the neighborhood best is only changed if the new neighborhood best particle ($x^*_{i''}$) has higher fitness than the previous best location of the current particle. In other words

$$x^*_{i'} = x^*_{i''} \text{ only if } fitness(x^*_{i''}) > fitness(x^*_i)$$

It should be noted that this excludes the previous best location of a particle from ever being its neighborhood best. We found that this modification dramatically improved the performance of the learning over the standard update method. Although we are not certain why this is the case, it may be that the new technique encourages diversity by ensuring that every particle will have two distinct locations to which it is attracted.

The parameters for the algorithms are given in Table 1.

Table 1: GA and PSO Parameters for Unsupervised Learning

GA		PSO	
Population Size	20	Population Size	20
Crossover Probability	0.6	pw	2.0
Mutation Probability	0.15	nw	2.0
Mutation Range	[-5.0, 5.0]	w	0.6

We use Webots, a realistic simulator, for our robotic simulations [14], using the Khepera robot model [16]. The robot(s) operate in a 2.0 m x 2.0 m square arena (see Fig. 1). The robotic controller is a single-layer discrete-time artificial neural network of two neurons, one for each wheel speed, with sigmoidal output functions. The inputs are the eight infrared proximity sensors (six in front, two in back), as well as a recursive connection from the previous output of the neuron, lateral inhibitions and bias values (see Fig. 2), giving us 22 weights total. Sensors have a maximum range of 5.0 cm, and sensor output varies linearly from 0.0 at maximum range to 5.12 at minimum range (0.0 cm) with 10% noise. Slip noise of 10% is applied to the wheel speed. The time step for neural updates is 128 ms. We use the fitness function used in [18]. The fitness function is given by:

$$F = V \cdot (1 - \sqrt{\Delta v}) \cdot (1 - i)$$
$$0 \leq V \leq 1$$
$$0 \leq \Delta v \leq 1$$
$$0 \leq i \leq 1$$

where V is the average absolute wheel speed of both wheels, Δv is the average of the difference between the wheel speeds, and i is the average activation value of the most active proximity sensor over the evaluation period. These factors reward robots that move quickly, turn as little as possible, and spend little time near obstacles, respectively. The terms are normalized to give a maximum fitness of 1. The evaluation period of the fitness tests for these experiments is 480 steps, or approximately 60 seconds. Between each fitness test, the position and bearing of the robots are randomly set by the simulator to ensure the randomness of the next evaluation.

We test for robot group sizes of 1, 2, 5, 10, and 20 for 100 iterations of each algorithm. Since learning is being done in parallel, this has a significant effect on the simulated time needed ($\sim$67 hours for 1 robot compared to $\sim$3 hours for 20). In the case of 20 robots, each robot effectively contains a single candidate solution in the algorithm population.

3.2 Results

A comparison of the average fitnesses can be seen in Fig. 3. The progress of the average population fitness throughout the learning process for a 20-robot group can be seen in Fig. 4. There is no significant change in the performances of the algorithms for different robot group sizes, indicat-

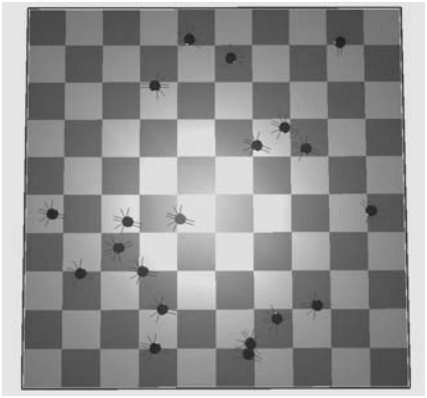


Figure 1: Robot arena with Khepera robots. Lines protruding from Kheperas represent proximity sensors.

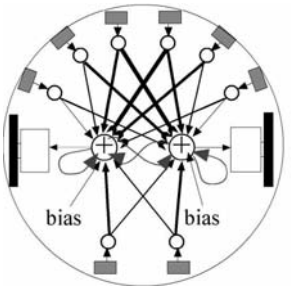


Figure 2: Depiction of the artificial neural network used for the robot controller. Grey boxes represent proximity sensor inputs and white boxes on the sides represent the motor outputs. Curved arrows are recurrent connections and lateral inhibitions.

ing that this technique is quite scalable. Although GA has initially faster convergence, the performance is noticeably lower than that of PSO for all group sizes. This was due to GA converging to poor solutions a large fraction of the time. A likely cause of this is the small population size (20 agents here as opposed to 60 in [18]), which does not provide enough genetic diversity for GA in this scenario, while PSO, though slower, is able to converge well with much smaller population sizes.

4. COMMUNICATION-BASED NEIGHBORHOODS

In multi-robot scenarios, communication range is often limited. Untethered robots have a very limited amount of available energy at their disposal, and it is important to conserve this by restricting transmission power. Also, if communication range is too large, interference between signals can decrease the rate at which data can be sent. If we distribute particles in a PSO population between robots and use the standard PSO local neighborhood model, robots may be required to share information with other robots that are far from their position. Therefore, to realistically model a scalable multi-robot system, particle neighborhoods should be set in such a way that robots are not required to communicate with other robots outside of some close proximity.

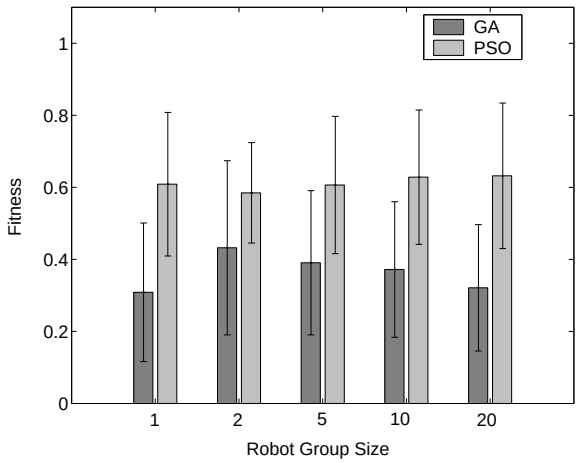


Figure 3: Average of final best performances over 20 evolutions for GA and PSO with different robotic group sizes. Error bars represent standard deviation across evolutionary runs.

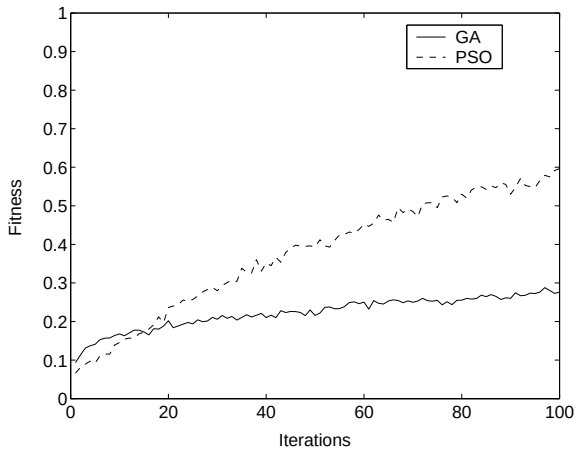


Figure 4: Average performance of population over 20 evolutions for GA and PSO with 20-robot groups.

4.1 Experimental Setup

We propose two such models for PSO neighborhoods to emulate realistic robot communication.

Model 1: Each robot contains one particle. At the end of each fitness evaluation, the robot selects the two robots closest to it, and uses their particles as its neighborhood for the next iteration of the algorithm. This maintains the same number of particles in the neighborhood, but allows for the neighbors to change over the course of the learning. As the physical location of the robots is independent of the particle indices, this should be roughly equivalent to randomly choosing two neighbors at each iteration of the algorithm, especially since obstacle avoidance behavior should result in a uniformly random distribution of robots within the environment.

Model 2: Each robot contains one particle. At the end of each fitness evaluation, the robot selects all robots within a fixed radius r, and uses their particles as its neighborhood for the next iteration of the algorithm. This results in a

variable number of neighbors, as the robot may be close to very few or very many robots randomly. However, it is perhaps more realistic than Model 1, since for very sparse robot distributions, there may be fewer than two other robots in close proximity at times.

We compare the performance of the original neighborhood topology to the two new models, using $r = 40$ cm, for a group of 20 robots. We use the setup previously described.

4.2 Results

A comparison of the average fitnesses is shown in Fig. 5. Both new neighborhood models achieve slightly better fitness than the original. This suggests that random neighborhood selection at each iteration is marginally superior to the fixed ring topology. The good performance of Model 2 indicates that the effectiveness of learning is not tied to keeping strictly two neighbors at each iteration. The success of these models shows that we can accomplish distributed unsupervised learning in a realistic multi-robot system.

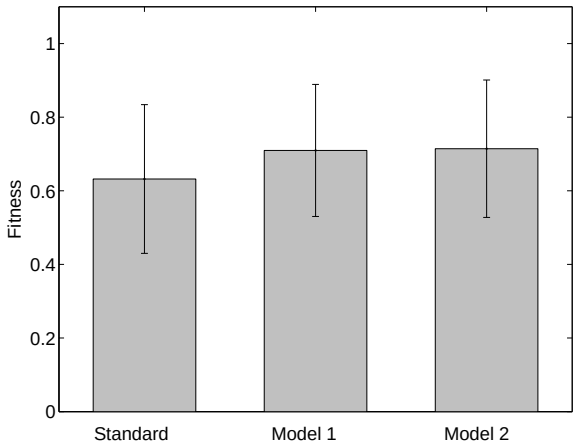


Figure 5: Average of final best performances over 20 evolutions for different neighborhood models. Error bars represent standard deviation across evolutionary runs.

5. VARYING COMMUNICATION RANGE

We now explore the effects of varying the communication range used in Model 2. This could be accomplished in a real robotic system by varying the output power of the transmission. It is useful to know the trade-off between output power and learning performance.

5.1 Experimental Setup

We use communication ranges of 10 cm, 20 cm, 40 cm, 80 cm, and 160 cm. The expected number of robots within communication range are given in Table 2, assuming a uniformly random distribution of robots within the arena. We therefore go from almost no interparticle communication to almost full interparticle communication.

5.2 Results

The average fitnesses for different communication ranges can be seen in Fig. 6. The progress of the average population fitness throughout the learning process for 10 cm, 40 cm, and 160 cm can be seen in Fig. 7. Both very high and very low communication ranges achieve fairly poor performance, while the intermediate ranges all achieve fairly good results.

Table 2: Expected Number of Neighboring Particles

r (cm)	Expected Number of Neighbors
10	0.14
20	0.54
40	2.0
80	6.5
160	16

Failure of low communication range is due to not enough information being exchanged between particles; particles end up almost exclusively using their own personal best position for learning, which causes extremely slow convergence. In the case of very high communication range, the initial convergence of the population was faster than with the shorter communication ranges, but it would often prematurely converge on a solution which did not have particularly high performance. This indicates that a global neighborhood is actually detrimental to finding very good solutions, and we therefore gain no benefit whatsoever by expanding our communication range beyond a certain point.

Both communication ranges of 40 cm and 80 cm (corresponding to average neighborhood sizes of 2.0 and 6.5 particles respectively) achieved very high fitness. Even a communication range of 20 cm, corresponding to 0.54 neighbors on average, achieved good fitness. The success of all these suggests that the effectiveness of the algorithm is not highly dependent on choosing an exact neighborhood size, making the algorithm parameters quite flexible. This is an important feature, as the communication range with real robots can vary due to obstruction and environmental effects.

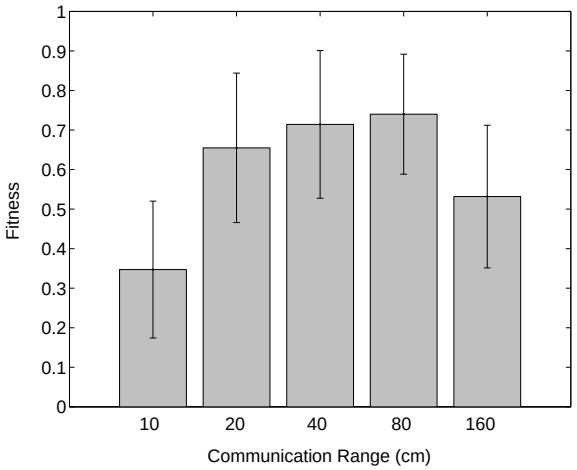


Figure 6: Average of final best performances over 20 evolutions for different communication ranges in Model 2. Error bars represent standard deviation across evolutionary runs.

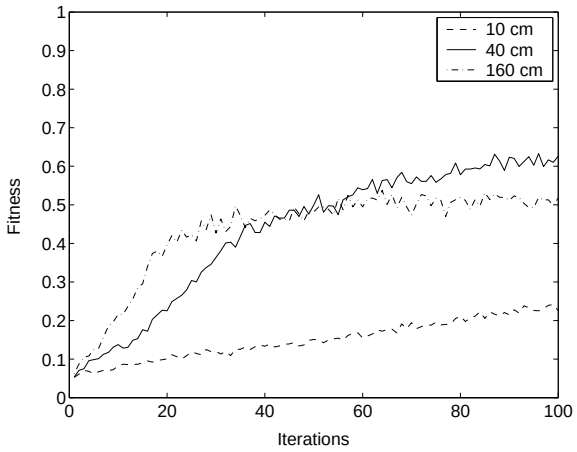


Figure 7: Average performance of population over 20 evolutions for 10 cm, 40 cm, and 160 cm communication range in Model 2.

6. GROUP LEARNING AND CREDIT ASSIGNMENT

Obstacle avoidance is a largely single-robot behavior. The observations and actions of other robots do not impact a robot's performance, except in having to avoid robots which move into its path. We wish to explore how susceptible our algorithms are to the credit assignment problem by evolving aggregation, a behavior whose success is highly dependent on the coordinated actions of many agents in the group.

6.1 Experimental Setup

We endow the Khepera robots with the ability to detect the relative positions of other nearby robots. This measurement is completely independent of any global coordinate system, and is given solely by where other robots are from the detecting robot's point of view (for example, range, the distance to the other robot, and bearing, the angular offset from the detecting robot's forward direction). This is a capability common to many robots working in collective scenarios, and can be accomplished with fairly simple systems (e.g., [19]). We add zero-mean Gaussian noise to the range and bearing, with range noise standard deviation equal to 10% of the range value and bearing noise standard deviation equal to 0.1 radians. We assume our relative positioning system is not susceptible to occlusions.

We expand the inputs to our artificial neurons to include relative positioning information. Because the number of robots within relative positioning range may vary, we use the center of mass of all detected robots as the input values. This is represented as x and y, where x is the forward-back displacement of the center of mass and y is the left-right displacement from the robot's point of view. This increases our total neural weights to 26.

The fitness value we use for this scenario is given by

$$F(i) = \frac{rob_{RP}(i)}{rob_{tot}}$$

where $F(i)$ is the fitness of robot i, $rob_{RP}(i)$ is the number of robots within relative positioning range of i, and rob_{tot} is the total number of robots. Therefore, a robot is rewarded for having the maximum number of other robots within relative proximity range at the end of a run.

The progress of evolving aggregation behavior may be susceptible to the first type of credit assignment problem described in Section 2, where robots use individual fitness values which can be impacted by the actions of other robots in the group, causing inaccurate evaluations. We therefore wish to compare the performance of our normal heterogeneous algorithms to homogenous algorithms, an established method of overcoming the credit assignment problem where all robots use the same controller at each evaluation. We generate a group fitness for each run by averaging all the individual fitness values obtained:

$$F_g = \frac{1}{rob_{tot}} \sum_i F(i)$$

Because the individual and group fitness functions are well-aligned, this allows us to compare their performances in a very fair manner. While homogenous learning will drastically slow the algorithm speed since we can no longer evaluate controllers in parallel, it will immediately provide a very noise-free estimation of the effectiveness of the solution, something which may not be available in the heterogeneous case (e.g., a robot may have a very good controller, but achieves poor performance because no other robot is aggregating well).

We use an unbounded arena with 20 Khepera robots for our setup. At each evaluative run, the Kheperas are distributed randomly in a 2m x 2m square. The evaluation lasts 10 simulated seconds, and the fitness is measured at the end. Robots are capable of sensing other robots within 80 cm of them. The arena can be seen in Fig. 8.

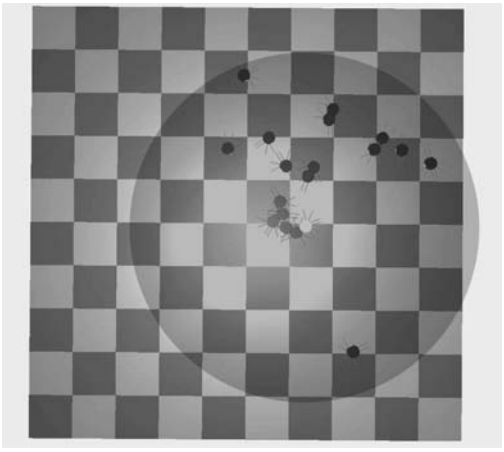


Figure 8: Aggregation arena with Khepera robots aggregating. The dimmed circle represents the relative positioning range of the white robot. All robots within this range are detectable.

We use our noise-resistant GA and PSO (with ring topology) algorithms and homogenous versions of these algorithms (HGA and HPSO, respectively), as well as the Model 1 and Model 2 neighborhood versions of the PSO algorithm. All algorithms have the same parameters used previously. Model 2 uses $r = 40$ cm. We run 100 iterations of the heterogeneous algorithms. Because we are using 20 robots, homogenous versions of the algorithms progress 20 times slower than the heterogeneous versions. We therefore run only 5 iterations

of these algorithms to match the number of evaluative runs.

6.2 Results

The final performance of all algorithms can be seen in Fig. 9. The progression of GA, PSO, HGA, and HPSO over the evolution can be seen in Fig. 10. All algorithms achieved good results in this scenario. For GA, the heterogeneous algorithm performed slightly worse than the homogenous version, while heterogeneous PSO performed as well as homogenous PSO on average, though with a higher standard deviation. This suggests that PSO may be less susceptible to the credit assignment problem than GA. However, as very few iterations were performed with the homogenous algorithms, it is likely HPSO would be able to achieve superior performance in longer runs. If we observe the progression of the algorithms throughout the learning process, GA initially improves more quickly, but levels off, while PSO continues to improve throughout. GA could therefore be preferable to PSO if we are only able to run very few iterations.

While the best final solutions from homogenous GA and PSO achieved very similar performances, the average population fitness for HGA was much higher than the average population fitness for HPSO throughout the learning process. This was observed in [18] and is likely due to the GA population containing much less variation than the PSO population, as the local neighborhood in PSO maintains diversity and no mechanism is present in GA to accomplish this. This diversity is likely what allows PSO to continue improving after the GA population converges on some solution.

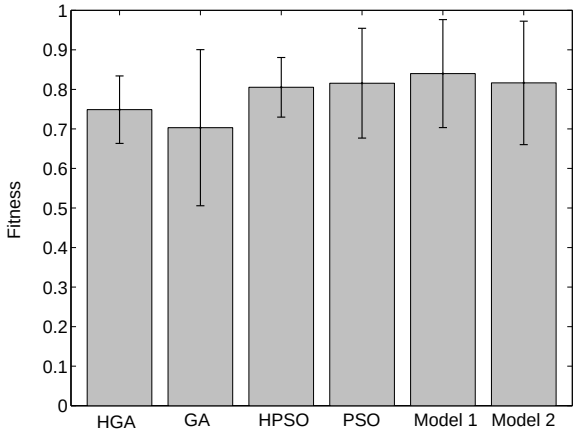


Figure 9: Average of final best performances in aggregation over 20 evolutions. HGA and HPSO are homogenous versions of the GA and PSO algorithms. Model 1 and Model 2 are PSO with neighborhoods described in Section 4. Error bars represent standard deviation across evolutionary runs.

Model 1 and Model 2 neighborhoods again achieved performances comparable to the standard ring topology, in spite of the very different distribution of robots in aggregation (high performing robots will be clustered together, while low performing robots may have traveled very far from the group center). The fact that these models continue to perform well is a good indication that neighborhoods based on limited communication capabilities of real robots can continue to function in a variety of scenarios.

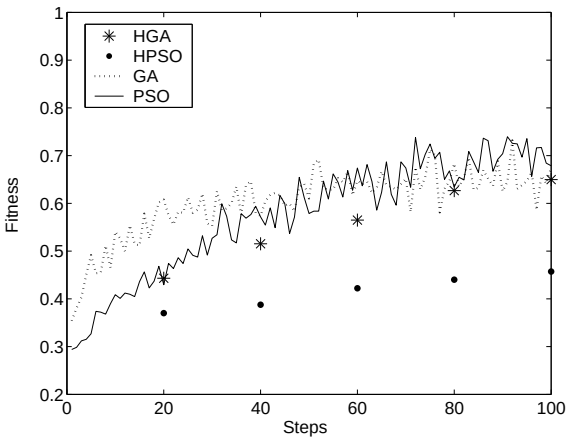


Figure 10: Average performance of population in aggregation over 20 evolutions. HGA and HPSO are the homogenous versions of the GA and PSO algorithms.

7. DISCUSSION AND OUTLOOK

Although PSO far outperformed GA in Section 2 of this paper, we suspect this is due almost exclusively to the small population size used. Indeed, the good performance of GA in Section 6 indicates that it can perform comparatively to PSO, and even better for fewer iterations. However, because of the different ways in which the population is managed, we would need to modify GA much more heavily in order to allow it to function distributedly with limited communication. Although this is certainly possible (e.g., as was done in [17]), the fundamental changes to the algorithm structure make it much more likely that we will lose the useful dynamics of the algorithm, as compared to PSO, where very few modifications are needed.

By increasing the robot group size from 1 to 20 robots, we were able to decrease the behavior learning time by a factor of 20. However, using a particle population size of 20, there is no easy to further decrease the time while maintaining only local interactions. It may be possible to use 40 robots to simultaneously evaluate the 20 new particles and reevaluate 20 previous best particles, but this would require a global supervisor to manage the assignments of candidate solutions to different robots, as each robot is no longer fully in charge of a particle. Therefore, further increasing the number of robots using only local interactions would only allow us to increase the size of the population. It has yet to be explored how increasing the population size could effect the convergence time.

In our model, the progress of all the robots was synchronized (i.e., fitness evaluations began and ended at the same time). In real-world multi-robot scenarios, this often isn't the case. Therefore, it would not make sense to exchange particle information only at the end of an evaluation, as the difference in time between robots could cause major delays. A simple alternative would be to exchange particle information on the previous evaluation during the evaluation itself. This is a rather minor modification to the algorithm, and we predict it will not significantly impact the performance. In fact, robots moving about during the evaluation would likely be exposed to more robots in close proximity, which

may allow them to further decrease their communication range while maintaining the same number of neighbors.

The model we use for communication in this paper is omnidirectional, immune to obstruction and error free. This corresponds to using radio transmissions in an open noiseless environment. In the real world, many other communication methods may be preferred or required. Infrared transmissions are often directional and could be blocked by other robots or by environmental obstacles. Radio could be blocked by large obstacles in some environments. Many types of communications may be susceptible to errors in noisy environments. The performance of the algorithm in these scenarios is thus far unexamined.

The scenario we used for testing susceptibility to the credit assignment problem only explored a small portion of the problem. It would be interesting to apply the algorithms to other scenarios, such those requiring specialization amongst robots or having less well-aligned group and individual fitness functions.

8. CONCLUSION

A modified version of the Particle Swarm Optimization algorithm was tested for unsupervised learning in groups of robots. The algorithm maintained good performance for groups of robots of various sizes. In the case of assigning a single unique particle to each robot, the performance was further improved by using PSO neighborhoods based on the limited communication abilities of real-world robots. Varying the communication range demonstrated that there is no benefit to communicating farther than a certain distance, and that the algorithm maintains high performance over a large variation of range. Applying the algorithm to a group learning task showed that it is able to overcome the credit assignment problem and that communication-based neighborhoods can perform well for non-uniform robot distributions. Implications of the results are examined and future research is suggested.

9. ACKNOWLEDGEMENTS

Jim Pugh and Alcherio Martinoli are currently sponsored by a Swiss NSF grant (contract Nr. PP002-68647).

10. REFERENCES

[1] Antonsson E. K, Zhang Y., & Martinoli A. "Evolving Engineering Design Trade-Offs". Proc. of the ASME Fifteenth Int. Conf. on Design Theory and Methodology, September 2003, Chicago, IL.

[2] Balch, T. *Behavioral diversity in learning robot teams.* PhD Thesis, College of Computing, Georgia Institute of Technology, 1998.

[3] Eberhart, R. & Kennedy, J. "A new optimizer using particle swarm theory" Proc. of the Sixth Int. Symposium on Micro Machine and Human Science, MHS '95, 4-6 Oct 1995, pp. 39-43.

[4] Floreano, D. & Mondada, F. "Evolution of Homing Navigation in a Real Mobile Robot" Systems, Man and Cybernetics, Part B, IEEE Transactions on, Vol. 26, No. 3, Jun 1996, pp. 396-407.

[5] Fourie, P. C. & Groenwold, A. A. "The particle swarm optimization algorithm in size and shape optimization" Struct. Multidisc. Optim., 2002, Vo. 23, pp. 259-267.

[6] Goldberg, D. E. *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[7] Hayes, A. T., Martinoli, A. & Goodman, R. M. "Swarm Robotic Odor Localization: Off-Line Optimization and Validation with Real Robots", Special Issue on Biological Robots, D. McFarland, editor, Robotica, 2003, Vol. 21, pp. 427-441.

[8] Kelly, I. D. & Keating, D. A. "Faster learning of control parameters through sharing experiences of autonomous mobile robots" Int. Journal of System Science, 1998, Vol. 29, No. 7, pp. 783-793.

[9] Kennedy, J. & Eberhart, R. "Particle swarm optimization" Neural Networks, 1995. Proceedings., IEEE International Conference on, Vol.4, Iss., Nov/Dec 1995, pp. 1942-1948.

[10] Kennedy, J. & Spears, W. M. "Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator" in Proceedings of IEEE International Conference on Evolutionary Computation, Anchorage, May 1998, pp. 78-83.

[11] Matarić, M. J. "Learning to Behave Socially" In Proc. of the 3rd Int. Conf. on Simulation and Adaptive Behaviors - From animals to animats 3, 1994, pp. 453-462.

[12] Matariíc, M. J. & Cliff, D., "Challenges in evolving controllers for physical robots", Robot. and Autonomous Syst., 1996, Vol. 19, No. 1, pp. 6783.

[13] Matarić, M. J. "Learning in behavior-based multi-robot systems: Policies, models, and other agents" Special Issue on Multi-disciplinary studies of multi-agent learning, Ron Sun, editor, Cognitive Systems Research, 2001, Vol. 2, No. 1, pp. 81-93.

[14] Michel, O. "Webots: Professional Mobile Robot Simulation" Int. J. of Advanced Robotic Systems, 2004, Vo. 1, pp. 39-42.

[15] Mitchell, M. *An Introduction to Genetic Algorithms.* MIT Press, Cambridge, MA, 1996.

[16] Mondada, F., Franzi, E. & Ienne, P. "Mobile robot miniaturisation: A tool for investigation in control algorithms" Proc. of the Third Int. Symp. on Experimental Robotics, Kyoto, Japan, October, 1993, pp. 501-513.

[17] Nehmzow, U. "Learning in multi-robot scenarios through physically embedded genetic algorithms" In Proc. of the 7th Int. Conf. on the Simulation of Adaptive Behavior: From animals to animats, 2002, pp. 391-392.

[18] Pugh, J., Zhang, Y. & Martinoli, A. "Particle swarm optimization for unsupervised robotic learning" Swarm Intelligence Symposium, Pasadena, CA, June 2005, pp. 92-99.

[19] Pugh, J. & Martinoli, A. "Relative Localization and Communication Module for Small-Scale Multi-Robot Systems", Proc. of the IEEE International Conference on Robotics and Automation, Miami, Florida, USA, May 15-19, 2006.

[20] Stone, P. *Layered Learning in Multi-Agent Systems.* PhD Thesis, School of Computer Science, Carnegie Mellon University, 1998.

Multi-Model Motion Tracking under Multiple Team Member Actuators

Yang Gu
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
guyang@cs.cmu.edu

Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
mmv@cs.cmu.edu

ABSTRACT

Autonomous robots need to track objects. Object tracking relies on predefined robot motion and sensory models. Tracking is particularly challenging if the robots can actuate on the object to be tracked, as the motion can become highly discontinuous and nonlinear. We have previously developed a successful tracking approach that switches among target motion models as a function of one robot's actions. In this paper, we consider the object to be effected by a team of agents. We contribute on our team-based tracking method that can use a dynamic multi-motion model based on a team coordination plan. We present the multi-target multi-model probabilistic tracking algorithm in detail and present empirical results both in simulation and in a human-robot Segway soccer team. The team coordination plan allows the robot to much more effectively track mobile targets.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics—*Sensors, Autonomous vehicles, Kinematics and dynamics*

General Terms

Algorithms, Experimentation, Human Factors

Keywords

multi-model, motion modeling, tracking, actuator

1. INTRODUCTION

Autonomous robot agents need to be able to track moving objects. (e.g. [11]). When tracking is performed by a robot executing specific tasks acting over the target being tracked, such as a Segway RMP soccer robot grabbing and kicking a ball, the motion model of the target becomes dependent on the robot's actions [9]. The robot's tactic provides valuable information in terms of the target behavior. A tactic-based motion modeling and tracking in such scenarios has been introduced in [7].

However, the Segway RMP soccer robot operates in such a environment that all the players on the field can actuate over the ball, namely grab and kick the ball according to the rules, which makes the motion model of the ball even more complex. When the robot is joining a game as a member of a human-robot team, the team coordination knowledge provides further information that can be incorporated into the motion modeling and tracking process. Furthermore, the approach in [7] does not perform well once incorrect measurements originating from clutter or false alarms exist, which causes multiple hypothesis of the tracked target. Recently, a hybrid approach for online joint detection and tracking for multiple targets was proposed [10]. This approach does not assume the knowledge of true targets (without clutter) is given. It first uses a deterministic clustering method that searches for regions of interest (ROIs) based on the observations and monitors these ROIs for target detection, then performs multi-target tracking by Sequential Monte Carlo (SMC) methods. In this paper, based on their approach, we present an extension to the tactic-based tracking scheme introduced in [7] to solve a plan-dependent multi-target multi-hypothesis tracking problem.

The paper is organized as follows. We first give a brief description of the Segway RMP soccer robot. Next we show the system dynamics. We introduce play-based motion model which incorporates the team coordination knowledge. We then describe the multi-target multi-model tracking algorithm, leading to our experimental results, related work, and conclusions.

2. SEGWAY RMP SOCCER ROBOT

The Segway platform is unique due to its combination of wheel actuators and dynamic balancing [5]. Segway RMP, or Robot Mobility Platform, provides an extensible control platform for robotics research. It imbues the robot with the novel characteristics of a fast platform and travel long ranges, able to carry significant payloads, able to navigate in relatively tight spaces for its size, and provides the opportunity to mount sensors at a height comparable to human eye level [12].

In our previous work, we have developed a Segway RMP robot base capable of playing Segway soccer. We briefly describe the two major components of the control architecture, the sensor and the robot cognition, which are highly related to our multi-model motion tracking.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2.1 Vision Sensor and Infrared Sensors

The goal of vision is to provide as many valid estimates of targets as possible. Tracking then fuses this information to track the most interesting targets (the ball and the team member, in this paper) of relevance to the robot. We use one pan-tilt camera as the vision sensor. We do not discuss the localization of the robot in the sense that a lot of soccer tasks (known as tactics and plays described in section 2.2) can be done by the Segway RMP robot without localization knowledge. Also we use global reference for position and velocity in this paper which means it is relative to the reference point where the robot starts to do dead reckoning.

We have equipped each robot with infrared sensors to reliably detect the objects located in the catchable area of the robot. Its measurement is a binary value indicating whether or not an object is there. In most cases, this is the blind area of the pan-tilt camera. Therefore, the infrared sensor is particularly useful when the robot is grabbing the ball.

2.2 Robot Cognition

A control architecture, called Skills-Tactics-Plays, was proposed in [4] to achieve the goals of responsive, adversarial team control. The key component of STP is the division between single robot behavior and team behavior.

A play, P, is a fixed team plan consisting of a set of applicability conditions, termination conditions, and N roles, one for each team member. Each role defines a sequence of tactics $T_1, T_2, \cdots$ and associated parameters to be performed by that role in the ordered sequence. Assignment of roles to team members is performed dynamically at run time. Upon role assignment, each robot i is assigned its tactic T_i from the current step of the sequence for that role.

A tactic, T, encapsulates a single robot behavior. Each robot i executes its own tactic as created by the current play P. A tactic T_i determines the skill state machine SSM_i to be executed by the robot i.

A skill, S, is a focused control policy for performing some complex action. Each skill is a member of one, or more, skill state machines $SSM_1, SSM_2, \cdots$. Each skill S determines what skill it transitions to S' based upon the world state, the time skill S has been executing for, and the executing tactic for that robot.

We construct the robot cognition using a similar architecture. Plays, tactics, and skills, form a hierarchy for team control. Plays control the team behavior through tactics, while tactics encapsulate individual robot behavior and instantiate actions through sequences of skills. Skills implement the focused control policy for actually generating useful actions. Figure 1 shows the $SSMs$ and transitions for an example tactic: *CatchKickToGoal*, which contains six skills. Each node in the figure is a skill and the edges show the transition between skills.

Segway soccer is a team sport, and therefore the building of our game strategy required not only execution of single robot behavior , but also coordination with the team member, the human player. The current coordination is simple and basically based upon two fixed plays for offensive and defensive situation respectively. Our offensive play is shown as follows, in which the termination condition is either play aborted or the situation changed (a turn-over of ball possession announced by the referee). There are two roles in this play, one passes the ball to the other who positions down field and waits for receiving a pass. Once the positioning

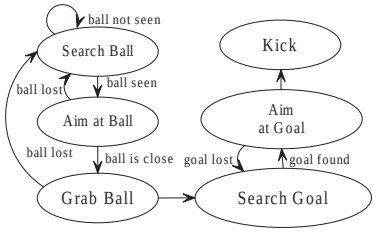


Figure 1: Skill state machines (SSMs) for an example tactic: *CatchKickToGoal*.

is done (e.g the other is closer to the opponent goal), the passing is performed.

```
PLAY Naive Offense
APPLICABLE offense
DONE aborted OR !offense
ROLE 1
  pass 2
  none
ROLE 2
  position_down_field
  receive_pass
  none
```

Our current coordination is purely observation based. Each player assigns role from his own eyeshot without communication. For example, should the robot think the team member is closer to the ball, the robot (ROLE 2) would choose to position and receive the ball from its team member (ROLE 1). Furthermore, the robot knows which side gains possession of the ball from the referee announcement, therefore it tells offensive from defensive situation clearly and thus it has deterministic idea of which play the team is using. The robot makes an assumption that its team member is performing the same game play as itself. The robot infers what tactic the team member is executing from the team play. For example, after receiving the ball from the team member, as a passer the robot would assume the team member go forward to a tactically advantageous position to receive a pass. The predefined play for team coordination helps motion modeling, which will be further discussed in section 3.

3. PLAY-BASED MOTION MODELING

In this section, we take a multi-target tracking problem as a detailed example to show the extension of the tactic-based motion modeling method in general when the team coordination knowledge (play) is incorporated. First we give an introduction of the environment and targets under the Segway soccer setup. Second, we describe detailed motion models for both the ball and the team member. Third, we extend the tactic-based motion modeling to the play level when both the ball and the team member are included into the tracking. We show how we model the play-dependent interactions between the team member, the robot and the ball and set up a base for giving the multi-model tracking algorithm in the next section. Although we present the Segway soccer domain as an example, the formalism is general.

3.1 Tracking Scenario

Many tracking scenarios involve multiple moving targets. In a Segway soccer game, we need to track the ball, the hu-

man team member and the two opponents. Each team is identified by their distinct color. The ball is orange [13]. An observation from the sensors might consist of multiple measurements due to the moving objects and the clutter. Generally the clutter has similar color as the targets we are interested in and it causes multiple hypothesis for the true targets. Therefore, we construct two multi-target trackers in the system, for the ball and the team member respectively. We use two separate trackers instead of one because we can differentiate the ball with the team member thanks to the color-based vision system. We use multi-target tracker instead of single target tracker because we can keep track of the true target and the false positive at the same time without losing any of them and perform ball(or team member) recognition from a pool of tracked objects later.

3.2 State Space and Dynamics

Let $\mathbf{x}_t = [\mathbf{x}_{1,t}^T, \mathbf{x}_{2,t}^T, \cdots, \mathbf{x}_{K_t,t}^T]^T$, denote a combined target state vector for K_t unknown and time-varying number of targets. The general parameterized system process for the kth target $\mathbf{x}_{k,t}$ at time t is given by:

$$\mathbf{x}_{k,t} \quad = \quad \mathbf{f}_k(\mathbf{x}_{k,t-1}, \mathbf{u}_{k,t-1}, \mathbf{v}_{k,t-1}) \tag{1}$$

where f_k is the parameterized state transition functions for the kth target; $\mathbf{x}, \mathbf{u}$ are the state and input vectors; $\mathbf{v}$ is the process noise vectors of known statistics. Given the number of targets at $t-1$ and t, the state transition can be represented as follows:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, K_t, K_{t-1})$$

$$= \begin{cases} p_0(\mathbf{x}_{K_t,t}) \prod_{k=1}^{K_{t-1}} p(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}), if\ K_t = K_{t-1}+1 \\ \prod_{k=1}^{K_t} p(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}), if\ K_t = K_{t-1} \\ \prod_{k=1,k\neq k^*}^{K_{t-1}} p(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1}), if\ K_t = K_{t-1}-1 \\ 0,\ otherwise \end{cases} \tag{2}$$

The prior $p(\mathbf{x}_{k,t}|\mathbf{x}_{k,t-1})$ can be evaluated from (1). The function $p_0(\mathbf{x}_{K_t,t})$ is the initial distribution of a new target state and k^* is a target that vanishes at t. We assume at most one target appears or vanishes in each time step. The number of targets is obtained deterministically (see section 4.1 for the details).

3.3 Observation Model

We assume that measurements are independent of each other and either originate from true targets or clutter. Furthermore, each target generates at most one measurement in each time step but may not be detected by the sensor.

Let $\mathbf{z}_t = [\mathbf{z}_{1,t}^T, \mathbf{z}_{2,t}^T, \ldots, \mathbf{z}_{M_t,t}^T]$ denote an observation vector for M_t measurements. The observation equation for modeling the cth measurement originating from the kth target is given as:

$$\mathbf{z}_{c,t} \quad = \quad \mathbf{h}_k(\mathbf{x}_{k,t}, \mathbf{n}_{c,t}) \tag{3}$$

where $\mathbf{n}_{c,t}$ is a zero-mean observation noise with a known covariance Σ_n. The measurement originating from a clutter is modelled to be uniformly distributed within the entire visible region (observation volume V) of the camera. We use a deterministic method to perform measurement-to-target association. Let α_t denote the association vector which indicate the measurement-to-target assignment, whose ith element $\alpha_{i,t}$ is set to j if the ith measurement originates from the jth target, or zero if it originates from clutter. Let N_{C_t} denote the number of clutter points, and Ω_D denote the set of measurements indices corresponding to the detected targets, the likelihood function for measurement $\mathbf{z}_t$ can be written as:

$$p(\mathbf{z}_t|\mathbf{x}_t, K_t, \alpha_t) = (\frac{1}{V})^{N_{C_t}} \prod_{l\in\Omega_D} p(\mathbf{z}_{l,t}|\mathbf{x}_{\alpha_{l,t},t}) \tag{4}$$

where $p(\mathbf{z}_{l,t}|\mathbf{x}_{\alpha_{l,t},t})$ is evaluated from (3).

3.4 Ball Motion Modeling

In our Segway RMP soccer robot environment, we define five models to model the ball motion (for the rest of this paper, for simplicity, we use $\mathbf{x}_t$ to represent the ball state, and use $\mathbf{x}'_t$ to represent the team member state).

- *Free-Ball*, *Robot-Grab-Ball*, *Robot-Kick-Ball*. We use the same models as *Free-Ball*, *Grab-Ball* and *Kick-Ball* introduced in [7] respectively.
- *Human-Grab-Ball*. The ball is held by the team member. we can infer the ball position similarly if we know the team member position well.
- *Human-Kick-Ball*. The ball is kicked by the team member and it is supposed to be either a pass to the robot or a shoot at the goal .

In general, we will have n motion models $m_1, m_2, \cdots, m_n$.

3.5 Team Member Motion Modeling

We define four models to model the human team member's motion.

- *Random Walk*. The team member is wondering in the field. So the state at the new time is the state at the current time with some additive zero-mean (assumed Gaussian) noise.
- *Holding Ball*. The team member is holding the ball without moving and waiting for the robot to receive the ball. Should the robot know the ball position well, it can infer the team member position by the ball position in a similar way as *Robot-Grab-Ball* for ball motion modeling.
- *Accelerating (Decelerating)*. The team member dashes (stops) to obtain (lose) a velocity in a short time.
- *Positioning*. The team member is going to a predefined tactical position with a constant speed. This case happens mostly after the team member passing the ball to the robot and moving down the field toward opponent's goal.

3.6 Play Based Model Transitions

Given the knowledge of the team coordination plan (the play P_{t-1} at time $t-1$), the robot can infer what tactic the team member is executing (T'_{t-1}), which provides valuable information about the motion model of the team member (m'_t). Both the robot and the team member act over the ball in a Segway soccer game. The motion model of the ball (m_t) is therefore affected by what tactic the robot (T_{t-1}) and the team member (T'_{t-1}) are executing.

From the previous subsection, we know that the model index m determines the present model being used. For our team member tracking example, $m'_t = i, i = 1, \cdots, 4$. In our

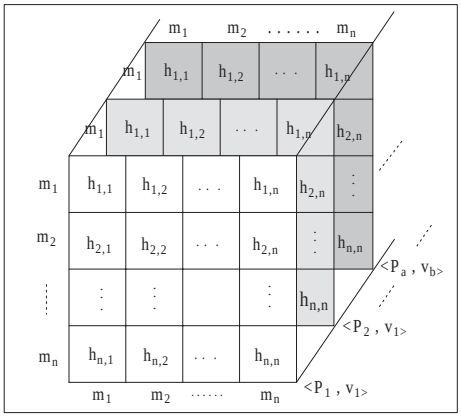


Figure 2: Play-Based motion modeling, where $m_1, m_2, \cdots, m_n$ are n models, P_a is the team play, v_b is the additional information. $h_{i,j}$ is the transition probability from model m_i to model m_j given m_i, and $\langle P_a, v_b \rangle$. Each layer in the graph is conditioned on a particular combination of the play executed and the additional information obtained.

approach, it is assumed that the team member motion model index, m'_t, conditioned on the previous tactic executed T'_{t-1} by the team member, and other useful information v'_t (such as ball state), is governed by an underlying Markov process, such that, the conditioning parameter can branch at the next time-step with probability.

$$p(m'_t = i | m'_{t-1} = j, T'_{t-1}, v'_t) = h'_{i,j} \tag{5}$$

where $i, j = 1, \cdots, N_{m'}$. Since T'_{t-1} can be determined by P_{t-1}, we get

$$h'_{i,j} = p(m'_t = i | m'_{t-1} = j, P_{t-1}, v'_t) \tag{6}$$

Since we can draw $p(m'_t = i | m'_{t-1} = j)$ in an $N_{m'} \times N_{m'}$ table, we can create a table for Equation 6 with a third axis which is defined by the tuple $\langle P_a, v_b \rangle$ as shown in Figure 2. Here the play P_a, is the primary factor that determines whether m_i transits to m_j and what the transition probability is, while the information v_b determines the prior condition of the transition. Each layer in the graph is conditioned on a particular combination of the tactic executed and the additional information obtained.

For our ball tracking example, $m_t = i, i = 1, \cdots, 5$. Similarly,

$$h_{i,j} = p(m_t = i | m_{t-1} = j, T_{t-1}, T'_{t-1}, v_t) \tag{7}$$

where $i, j = 1, \cdots, N_m$. Since T_{t-1}, T'_{t-1} can be determined by P_{t-1}, we get

$$h_{i,j} = p(m_t = i | m_{t-1} = j, P_{t-1}, v_t) \tag{8}$$

Suppose the current team play is the *Naive Offense* in Section 2.2, we can obtain the corresponding motion model transitions for both the ball and the team member using the play-based method (Figure 3).

4. MULTI-MODEL MOTION TRACKING

In this section, we first describe the clustering algorithm we used to continuously monitor the appearance and disappearance of regions of interest (ROIs) on the field. We then

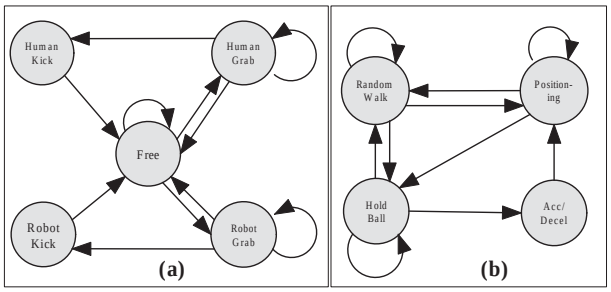


Figure 3: Object motion modeling based on the play: *Naive Offense*. Each node is a model. Models transit to one another according to the predefined probabilities (not shown in the figure). (a) Ball motion model. (b) Human team member motion model.

use dynamic Bayesian networks to represent the whole system. We give the detailed sequential Monte Carlo methods of tracking in a multi-model multi-hypothesis scheme finally.

4.1 Clustering Algorithm

The idea of this algorithm is to group a set of ROIs, $\mathcal{S}_t = \{S_t^{(j)}\}_{j=1}^{J_t}$, within a buffer of observations $\mathcal{Z}_t = \{\mathbf{z}_{t'}\}_{t-\tau}^{t}$, where τ is the length of the sliding window [10]. If the current targets are widely separated, the measurements originating from them will be clustered in the locations where the targets had visited from $t - \tau$ to t. Therefore the distance between two measurements within two successive time steps can be taken as the clustering criteria. Let $d_{c,l}(t'+1, t')$ denote the normalized distance between the cth and lth measurements of $\mathbf{z}_{t'+1}$ and $\mathbf{z}_{t'}$.

$$d_{c,l}(t'+1, t') = e_{c,l}^T(t'+1, t')\Sigma_e^{-1} e_{c,l}(t'+1, t') \tag{9}$$

where $c \in \{1, \ldots, M_{t'+1}\}$ and $l \in \{1, \ldots, M_{t'}\}$, $e_{c,l}(t'+1, t') = \mathbf{z}_{c,t'+1} - \mathbf{z}_{l,t'}$ and $\Sigma_e = 2\Sigma_n$. For all measurements of $\mathbf{z}_{t'+1}$, we have a set of normalized distances $\{d_{c,l}(t'+1, t')\}_{c=1}^{M_{t'+1}}$. The c^*th measurement $\mathbf{z}_{c^*,t'+1}$ will be grouped with $\mathbf{z}_{l,t'}$ in the same cluster $S_t^{(j)}$ if

$$d_{c^*,l}(t'+1, t') = argmin\{d_{c,l}(t'+1, t')\}_{c=1}^{M_{t'+1}} \tag{10}$$

and

$$d_{c^*,l}(t'+1, t') \leq \epsilon_z \tag{11}$$

where ϵ_z is a threshold. If none of the measurements of $\mathbf{z}_{t'+1}$ can be grouped with $\mathbf{z}_{l,t'}$, the search procedure continues and uses $\mathbf{z}_{t'+b}, 1 < b \leq \tau$ until (10) and (11) are both satisfied. Meanwhile the algorithm propagates through time to group other measurements in the remaining observations. Since the detected regions obtained via (10) and (11) possibly arise from clutter points, another threshold is set to examine the number of measurements in the region to eliminate the false positives. That is, if the number of measurements in the jth region $S_t^{(j)}$ is less than τ_{min}, defined as the minimum number of clustered measurements in a region required to identify a target, it is discarded; otherwise, the jth region is classified as originating from a target. Once a set of ROIs $\mathcal{S}_t = \{S_t^{(j)}\}_{j=1}^{J_t}$ is obtained, it is necessary to determine which region belongs to which existing active track or a

new track deterministically. Let $\gamma_t = [\gamma_{1,t}, \gamma_{2,t}, \cdots, \gamma_{K_t,t},]^T$ denote the track-to-region association vector. $\gamma_{k,t}$ is used to indicate the association between the ROIs and the active tracks. $\gamma_{k,t}$ is evaluated as j if track k can be associated with $S_t^{(j)}$, otherwise it is zero. Refer to [10] for more details of the clustering algorithm.

4.2 DBN Representation

Following the play-based motion model, we can use dynamic Bayesian networks (DBNs) to represent the whole system for team member and ball tracking in a natural and compact way as shown in Figure 4 and Figure 5 respectively. In the two graphs, the system state is represented by variables (play P, tactic T, infrared sensor measurement s, ball state $\mathbf{x}$, ball motion model index m, vision sensor measurement of ball $\mathbf{z}$, team member state $\mathbf{x}'$, team member motion model index m', vision sensor measurement of team member $\mathbf{z}'$), where each variable takes on values in some space. The variables change over time in discrete intervals, so that e.g. $\mathbf{x}_t$ is the ball state at time t.

Furthermore, the edges indicate dependencies between the variables. For instance, in Figure 5 the ball motion model index m_t depends on $m_{t-1}, T_{t-1}, T'_{t-1}, s_t$ and $\mathbf{x}_{t-1}$, hence there are edges coming from the latter five variables to m_t. For the rest of this section, we give the ball-tracking algorithm following Figure 5. The team-member-tracking algorithm can be obtained similarly following Figure 4.

4.3 Importance Sampling Function

We use the sequential Monte Carlo method to track the motion model m and the multi-target state $\mathbf{x}$. Particle filtering is a general purpose Monte Carlo scheme for tracking in a dynamic system [6]. It maintains the belief state at time t as a set of particles $p_t^{(1)}, p_t^{(2)}, \cdots, p_t^{(N_s)}$, where each $p_t^{(i)}$ is a full instantiation of the tracked variables $\{p_t^{(i)}, w_t^{(i)}\}$, $w_t^{(i)}$ is the weight of particle $p_t^{(i)}$ and N_s is the number of particles. In our case, $p_t^{(i)} = \langle \mathbf{x}_t^{(i)}, m_t^{(i)} \rangle$. To make our notation more concrete, a particular particle $p_t^{(i)}$, which is tracking K_t multi-target state vector $\mathbf{x}_t$ and motion model m_t, is given as [8]:

$$p_t^{(i)} = \begin{pmatrix} m_{1,t}^{(i)} & m_{2,t}^{(i)} & \cdots & m_{K_t,t}^{(i)} \\ x_{1,t}^{(i)} & x_{2,t}^{(i)} & \cdots & x_{K_t,t}^{(i)} \\ y_{1,t}^{(i)} & y_{2,t}^{(i)} & \cdots & y_{K_t,t}^{(i)} \\ \dot{x}_{1,t}^{(i)} & \dot{x}_{2,t}^{(i)} & \cdots & \dot{x}_{K_t,t}^{(i)} \\ \dot{y}_{1,t}^{(i)} & \dot{y}_{2,t}^{(i)} & \cdots & \dot{y}_{K_t,t}^{(i)} \end{pmatrix} \tag{12}$$

We sample the ball motion model following the ball-tracking DBN as below:

$$m_{k,t}^{(i)} \sim p(m_{k,t} | m_{k,t-1}^{(i)}, \mathbf{x}_{k,t-1}^{(i)}, s_t, T_{t-1}, T'_{t-1}) \tag{13}$$

Note that T_{t-1} and T'_{t-1} are inferred deterministically from P_{t-1} instead of sampling. Conditioned on the ball motion model $m_{k,t}^{(i)}$, we then use the importance function introduced in [10] to sample ball state $\mathbf{x}_{k,t}^{(i)}$:

$$\mathbf{x}_{k,t}^{(i)} \sim \begin{cases} q_D(\mathbf{x}_{k,t} | m_{k,t}^{(i)}, \mathbf{x}_{k,t-1}^{(i)}), & k \notin \mathcal{K}_{S_t} \\ q_{DS}(\mathbf{x}_{k,t} | m_{k,t}^{(i)}, \mathbf{x}_{k,t-1}^{(i)}, S_t^{(j)}), & k \in \mathcal{K}_{S_t} \end{cases} \tag{14}$$

where $k \in \mathcal{K}_{S_t}$ are those tracks with $\gamma_{k,t} = j$ and $j \in \Omega_D$, and $q_D(\cdot)$ and $q_{DS}(\cdot)$ are the proposal functions for $x_{k,t}$

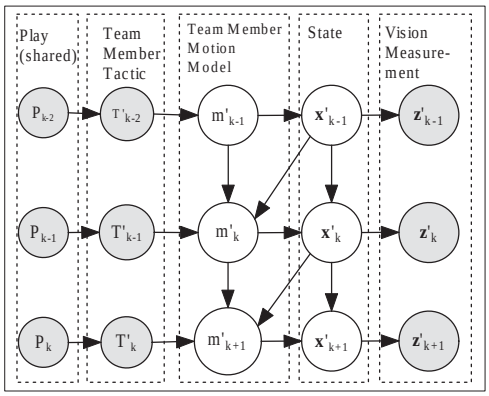


Figure 4: A dynamic Bayesian network for team member tracking with a Segway RMP robot. Filled circles represent deterministic variables which are observable or are known as the tactic or the play that the robot is executing.

without and with an associated ROI $S_t^{(j)}$, given as follows, repectively,

$$q_D(\mathbf{x}_{k,t} | m_{k,t}^{(i)}, \mathbf{x}_{k,t-1}^{(i)}) = p(\mathbf{x}_{k,t} | m_{k,t}^{(i)}, \mathbf{x}_{k,t-1}^{(i)}) \tag{15}$$

$$q_{DS}(\mathbf{x}_{k,t} | m_{k,t}^{(i)}, \mathbf{x}_{k,t-1}^{(i)}, S_t^{(j)}) =$$

$$\mu p(\mathbf{x}_{k,t} | m_{k,t}^{(i)}, \mathbf{x}_{k,t-1}^{(i)}) + (1-\mu) q(\mathbf{x}_{k,t} | m_{k,t}^{(i)}, \mathbf{x}_{k,t-1}^{(i)}, S_t^{(j)}) \tag{16}$$

where $0 \le \mu \le 1$ and $q(\cdot)$ is a uniform sampling from the associated ROI $S_t^{(j)}$. If $\mu = 1$, the importance sampling function is reduced back to the dynamic prior. If $\mu = 0$, all particles are generated from the data-dependent importance function. If $0 < \mu < 1$, this proposal combines the dynamic prior and the current ROIs to generate representative particles.

4.4 Birth, Death and Update Moves

Assuming that there are K_t^b ROIs that cannot be associated with any existing track, we will initiate a new track in each time step from one of these regions instead of initiating K_t^b tracks simultaneously in order to fit the birth move with the assumed system process model in (2). When an existing track cannot be associated with a region at a given time, the target being tracked by the tracker may have disappeared or temporarily experience a short period of data loss. Thus we may remove the track for the target only if it has failed to associate with any ROI with τ_d time steps. Refer to [10] for the detailed algorithm of birth move and death move.

In the update move, there is no change in terms of the number of ROIs. We only need to update the target states with a common value of number of targets $K_t^{(i)} = K_{t-1}^{(i)}$, using the sequential importance sampling method as follows:

$$[\{\mathbf{x}_t^{(i)}, m_t^{(i)}, w_t^{(i)}\}_{i=1}^{N_s}] =$$
$$\boldsymbol{SIS}[\{\mathbf{x}_{t-1}^{(i)}, m_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_s}, \mathbf{z}_t, s_t, T_{t-1}, T'_{t-1}]$$

01 **for** $i = 1 : N_s$
02 **for** $k = 1 : K_t$
03 draw $m_{k,t}^{(i)} \sim p(m_{k,t} | m_{k,t-1}^{(i)}, \mathbf{x}_{k,t-1}^{(i)}, s_t, T_{t-1}, T'_{t-1})$.
04 **if** track k has corresponding ROI $S_t^{(j)}$
05 draw $\mathbf{x}_{k,t}^{(i)} \sim q_{DS}(\mathbf{x}_{k,t} | m_{k,t}^{(i)}, \mathbf{x}_{k,t-1}^{(i)}, S_t^{(j)})$

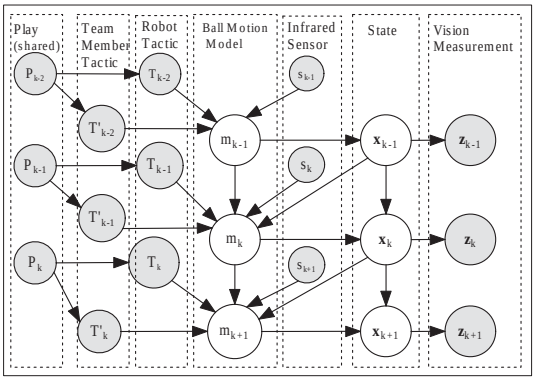


Figure 5: A dynamic Bayesian network for ball tracking with a Segway RMP robot.

```
06        else
07          draw x_{k,t}^{(i)} ~ q_D(x_{k,t} | m_{k,t}^{(i)}, x_{k,t-1}^{(i)})
08        end if
09      end for
10      set w_t^{(i)} = w_{t-1}^{(i)} · p(z_t | x_t, K_t, α_t)
11    end for
12    Calculate total weight: w = Σ[{w_t^i}_{i=1}^{N_s}]
13    for i = 1 : N_s
14      Normalize: w_t^i = w_t^i / w
15    end for
16    Resample.
```

The inputs of the algorithm are samples drawn from the previous posterior $\langle \mathbf{x}_{t-1}^{(i)}, m_{t-1}^{(i)}, w_{t-1}^{(i)} \rangle$, the present vision and infrared sensory measurement $\mathbf{z}_t, s_t$, the robot's tactic T_{t-1}, and the team member's tactic T'_{t-1}. The outputs are the updated weighted samples $\langle \mathbf{x}_t^{(i)}, m_t^{(i)}, w_t^{(i)} \rangle$. In the sampling algorithm, first, a new ball motion model index, $m_t^{(i)}$, is sampled according to (13) at line 03. Then given the model index, and previous ball state, a new ball state is sampled according to (14) at line 05/07. According to (4), the importance weight of each sample is given by the likelihood of the vision measurement given the predicted new ball state at line 10. Finally, each weight is normalized and the samples are resampled. Then we can estimate the ball state based on the mean of all the $\mathbf{x}_t^{(i)}$. Though we are trying to eliminate the clutter from the beginning of tracking (clustering algorithm), due to the property of the multi-target tracker, further recognition process might be done in order to figure out which tracked target is the true ball. Similarly the state of the team member $\mathbf{x}'_t$ can be obtained from the team member tracker.

5. EXPERIMENTAL RESULTS

From previous work we knew the initial speed and accuracy of the ball velocity after a kick motion. We profiled the system and measurement noise as well. In this section, we evaluate the effectiveness of our tracking system in both simulated and real-world tests.

5.1 Simulation Experiments

Because it is difficult to know the ground truth of the target's position and velocity in the real robot test, we do the simulation experiments to evaluate the precision of tracking.

Table 1: The average RMS error of position estimation and velocity estimation from human trackers and ball trackers.

Motion Model	Single Model	Multiple Model
Human Position Est RMS (m)	0.0030	0.0014
Human Velocity Est RMS (m/s)	0.42	0.025
Ball Position Est RMS (m)	0.0028	0.0017
Ball Velocity Est RMS (m/s)	0.4218	0.0597

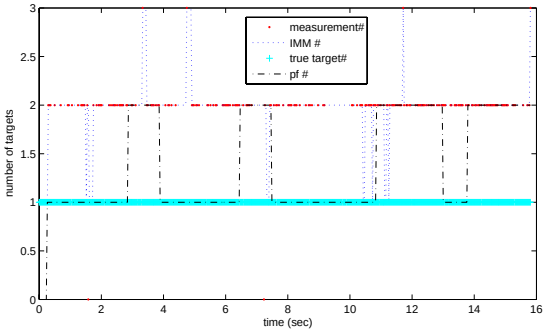


Figure 6: A comparison of the target detection performance between the proposed method and the IMM tracker when only one target exists with surrounding clutters.

Experiments are done following the *Naive Offense* play, in which the robot acts as the receiver and the human team member acts as the passer. Noises are simulated according to the model we profiled in previous work. In the beginning, the team member holds the ball. After a fixed amount of time, the ball is kicked towards the robot, and the team member moves forward to a predefined location.

We implement both a single model tracker and a play-based multi-model tracker for the ball and the team member. We simulate the experiment for 50 runs, and then compare the performance of the two trackers with different implementations. The average RMS error of position estimation and velocity estimation are shown in Table 1. The results show that the play-based multi-model scheme performs much better than the single model especially in terms of velocity estimation. Because with the play-based motion model, when the ball is being kicked, most particles evolving using the transition model determined by the play will change its motion model $m_t^{(i)}$ from *Free-Ball* to *Human-Kick-Ball*, and a velocity will be added to the ball accordingly.

5.2 Multi-Target Tracking Test

In this test, one Segway RMP robot is tracking one or more balls on the field with *SearchBall* tactic. We would like to compare solely the target detection performance between the proposed method and the IMM tracker. A scenario with K_t $(0 \leq K_t \leq 3)$ balls appearing and disappearing at different times and there are a set of false positives at fixed position in the surroundings.

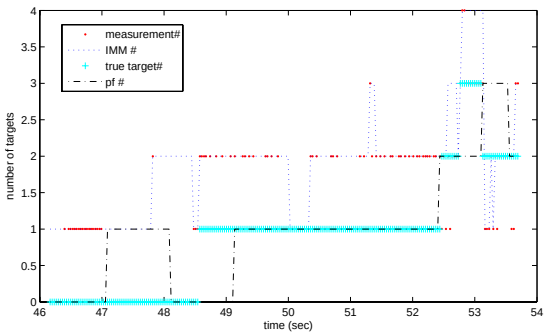


Figure 7: A comparison of the target detection performance between the proposed method and the IMM tracker when multiple targets exist with surrounding clutters.

When estimating the number of targets, 3600 particles are used in the proposed method. Figures 6-7 summarize the results. In both figures, the dots show the number of measurements at a given time. The dotted line represents the number of the targets tracked by the IMM tracker. The dashdotted line represents the number of targets tracked by the multi-model multi-target tracker proposed in this paper. The crosses show the true number of the targets at any given time. As shown in the figure, the IMM tracker is sensitive to the number of measurements, while our approach is more robust and consistent to high clutter density. Since the detection is basically performed on the clustering of the observations and the association between the detected ROIs and the existing tracks, it is computational low-cost. Therefore it is also practical for real-time multi-target detection and tracking.

5.3 Team Cooperation Test

We do experiments on the Segway RMP soccer robot executing the offensive play and coordinating with the human team member. The test setup is demonstrated in Figure 8, in which the digits along the lines show the sequence of the whole strategy, the filled circle at position B represents the robot, the unfilled circle at position E represent an opponent player, and the shaded circle represent the human team member.

When each run begins, the human team member is at position A . With this team cooperation plan (play), the robot chooses the tactic *CatchKickToTeammember* to execute, in which the robot starts with the skill *Search-Ball*. When the robot finds the ball, the team member passes the ball directly to the robot and chooses a positioning point to go to either at C or D. The robot grabs the ball after the ball is in the catchable area and is detected by the infrared sensor (skill *Grab-Ball*). Next the robot searches for the team member holding the ball with its catcher (skill *Search-Teammember*). After the robot finds the team member, the robot kicks the ball to its team member (skill *KickToTeammember*) and the team member shoots at the goal, completing the whole offensive play. Each run ends in one of the following conditions.

- succeed if the human receives the ball from the robot or the human does not receiver the ball but the pass can be considered as a "good" one.
- fail if the robot is in searching for the ball or the team member for more than 30 seconds.
- fail if the ball is outside the field before the robot catches it.

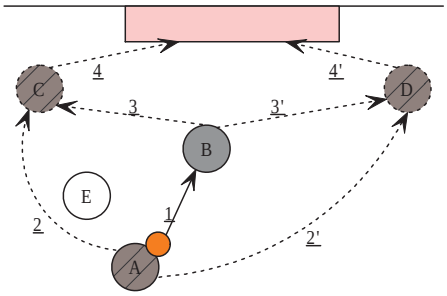


Figure 8: A demonstration of a naive team cooperation plan in offensive scenario. The digits along the lines show the sequence of the whole plan. The filled circle at position B represents the robot. The unfilled circle at position E represent an opponent player. The shaded circle represent the human team member.

Table 2: The average time taken over all the successful runs.

Motion Model	Single Model	Multi-Model
Mean Time (sec)	33.4	22.6

In the experiment over 15 runs, the robot with single model trackers fails 5 of the total. While the robot with play-based multi-model trackers fails 2 of the total. We also keep track of the mean time taken in all the successful runs. We list the result in Table 2. Using play-based multi-model tracking saves 32.3% time in terms of completing the whole play over single model tracking. During the experiment, we note that when using the single model tracking, most time were spent on searching the team member. Incorporating the team cooperation knowledge known as play into the team member motion modeling greatly improves the accuracy of the team member motion model and therefore avoids taking time in searching a lost target from scratch.

6. RELATED WORK

Tracking moving targets using a Kalman filter is the optional solution if the system follows a single model, **f** and **h** in Equation 1 and 3 are known linear functions and the noise **v** and **n** are Gaussians [1]. Multiple model Kalman filters such as Interacting Multiple Model (IMM) are known to be superior to the single Kalman filter when the tracked target is maneuvering [3]. For nonlinear systems or systems with non-Gaussian noises, a further approximation is introduced, but the posterior densities are therefore only locally accurate and do not reflect the actual system densities.

Since the particle filter is not restricted to Gaussian densities, a tactic-based motion modeling method is proposed

in [7]. Based on that approach, we further introduce the play-based motion modeling method when team coordination knowledge is available. Another related approach was proposed to track a moving target using Rao-Blackwellised particle filter [9], in which a fixed transition table was used between different models. Our transition model is dependent on the play that the robot is executing and the additional information that matters. This model can be flexibly integrated into our existing STP architecture.

There have been different strategies in multi-target tracking. In order to handle the data association and tracking problem, the classical Joint Probabilistic Data Association Filter (JPDAF) adopts the methods like the extended Kalman Filter (EKF) for multi-target state estimation, whose tracking performance is known to be limited by the linearity of the data models [2]. Another approach known as sequential Monte Carlo methods are able to perform well even when the data models are nonlinear and non-Gaussian. However, almost all of these methods assume that the knowledge of true targets (without clutter) is given, which is not applicable in the field that Segway RMP soccer robots operates in.

Recently, a hybrid approach for online joint detection and tracking for multiple targets was proposed [10]. This approach does not rely on the clutter-free assumption. In this paper, based on their approach, we present a play-based multi-target tracking algorithm, which incorporates tactic information to eliminate the false alarms and to improve resampling efficiency. Compared to our method, first, existing techniques consider less complex dynamic systems where only one part of the state space is non-linear. In contrast, our approach estimate a system where multiple components are highly non-linear (Segway RMP robot motion, ball motion, team member motion). Second, most existing techniques examine their performance with simulated experiments, while we test our approach in real robot experiments. Third, our approach goes beyond existing techniques by incorporating team coorperation information into the tracking process which further improves the performance.

7. CONCLUSIONS AND FUTURE WORK

Motivated by the interactions between a team and the tracked target, we contribute a method to achieve efficient tracking through using a play-based motion model and combined vision and infrared sensory information. This method gives the robot a more exact task-specific motion model when executing different tactics over the tracked target (e.g. the ball) or collaborating with the tracked target (e.g. the team member). Then we represent the system in a compact dynamic Bayesian network and use particle filter to keep track of the motion model and target state through sampling. The empirical results from the simulated and using the real robot agent show the efficiency of the multi-model tracking over single model tracking.

If the teammate is a human, not a robot, the certainty that the teammate is executing the expected play or tactic could be reduced. That is, the human teammate could fail to execute the desired play or tactic. Future work will take such uncertainty into account. A better human team member modeling (for example, include intercepting the moving ball, mark a player, covering the goal) will also help. Another interesting work is to know how the performance of the presented method is affected by the presence of tactics of the team member that are not exactly determined in the team coordination plan.

8. ACKNOWLEDGMENTS

We would like to thank the members of the CMBalance Segway soccer team for their help with developing the infrastructure for the Segway robots. In particular, we thank Brett Browning for the vision and initial control systems and Brenna Argall for the game skills.

This work was supported by United States Department of the Interior under Grant No. NBCH-1040007. The content of the information in this publication does not necessarily reflect the position or policy of the Defense Advanced Research Projects Agency (DARPA), US Department of Interior, US Government, and no official endorsement should be inferred.

9. REFERENCES

[1] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, Feb. 2002.

[2] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*. Academic Press, Inc, 1988.

[3] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc, 2001.

[4] B. Browning, J. Bruce, M. Bowling, and M. Veloso. STP: Skills, tactics and plays for multi-robot control in adversarial environments. *IEEE Journal of Control and Systems Engineering*, 219:33–52, 2005.

[5] B. Browning, J. Searock, P. E. Rybski, and M. Veloso. Turning segways into soccer robots. *Industrial Robot*, 32(2):149–156, 2005.

[6] A. Doucet, N. D. Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

[7] Y. Gu. Tactic-based motion modelling and multi-sensor tracking. *Proceedings of Twentieth National Conference on Artificial Intelligence*, 2005.

[8] C. Kreucher, K. Kastella, and A. O. H. III. Multi-target sensor management using alpha-divergence measures. pages 209–222, 2003.

[9] C. Kwok and D. Fox. Map-based multiple model tracking of a moving object. *Proceedings of eight RoboCup International Symposium*, July 2004.

[10] W. Ng, J. Li, S. Godsill, and J. Vermaak. A hybrid approach for online joint detection and tracking for multiple targets. *IEEE Aerospace Conferences*, 2005.

[11] D. Schulz, W. Burgrad, and D. Fox. People tracking with mobile robots using sample-based joint probabilistic data association filters. *International Journal of Robotics Research*, 22(2), 2003.

[12] J. Searock, B. Browning, and M. Veloso. Turning Segways into Soccer Robots. In *Proceedings of IROS'04*, Sendai, Japan, September 2004.

[13] M. Veloso, B. Browning, P. Rybski, and J. Searock. Segwayrmp robot football league rules. Technical report, http://www.cs.cmu.edu/ robosoccer/segway/, 2005.

Multi-Agent Strategic Modeling in a Robotic Soccer Domain

Andraz Bezek, Matjaz Gams
Department of Intelligent Systems
Jožef Stefan Institute
{andraz.bezek, matjaz.gams}@ijs.si

Ivan Bratko
Faculty of Computer and Information Science
University of Ljubljana
bratko@fri.uni-lj.si

ABSTRACT

This paper presents an algorithm for multi-agent strategic modeling (MASM) applied in a robotic soccer domain. It transforms a multi-agent action sequence into a set of strategic action descriptions in a graphical and symbolic form. By using hierarchically ordered domain knowledge, the algorithm is able to generate graphic and symbolic strategic action descriptions together with corresponding rules at different levels of abstraction. The method was evaluated on the RoboCup Soccer Server Internet League data.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning - *Concept learning*

General Terms

Algorithms, Experimentation.

Keywords

Multi-agent strategic modeling, strategy, RoboCup.

1. INTRODUCTION

Multi-agent modeling deals with the following task[15]: How can external observations of multi-agent systems be used to analyze, model, and direct agent behavior? Analysis of such systems must thus capture complex world state representation and asynchronous agent activities. Instead of studying pure numerical data researchers often use knowledge-level structures, such as rules and decision trees. Although being extremely useful when characterizing state space, they lack the ability to clearly represent temporal state changes occurred by agent actions. To comprehend simultaneous agent actions and complex changes of state space, most often a combination of graphical and symbolical representation performs better in terms of human understanding.

Multi-agent strategic modeling (MASM) is a process that is able to discover a strategic activity from raw multi-agent action data. A strategic activity is a time-limited multi-agent activity that exhibits some important or unique domain-dependant characteristics.

This paper addresses the problem of graphically and symbolically representing strategies of multi-agent systems (MAS). Our goal was to create an algorithm that would discover strategic agent behavior and enable humans to understand and study the underlying behavioral principles. The task is to translate multi-agent action sequence and observations of a noisy, dynamic, complex and multivariate world state into a graph-based and rule-based representation. By using hierarchically ordered domain knowledge the algorithm should be able to generate strategic action descriptions and corresponding rules at different levels of abstraction. More abstract descriptions also tend to be more noise resistant.

Our approach is applied in a Multi-Agent Strategy Discovering Algorithm (MASDA) and tested on a RoboCup Simulated League domain (RoboCup) [12], a multi-agent domain where two teams of 11 agents play simulated soccer games. The physical 2D soccer is simulated with additional noise when calculating forces on objects. Continuous time is approximated with discrete cycles. All agents can move and act independently as long as they comply with soccer rules. Agents can freely communicate with each other, but as in real world their visual and hearing perception is distance-limited. The domain is at the same time a complex MAS and easily understandable by humans due to its soccer-related content.

An example of a sequence of RoboCup actions is presented in Figure 1. The four frames show a successful direct-attack action. The players are presented as filled circles with a pointer indicating body direction. The attacking team is black while the defending team is gray. The ball is a white circle with black border; its trace is drawn throughout the sequence. Lines represent traces of movement. The action sequence in Figure 1 is typically described by humans as: "The left team's left forward player dribbles from the left half of the middle third into the penalty box. He then passes a ball to the center forward in the center of the penalty box, where the receiver successfully shoots into the right part of the goal." This sequence is graphically presented in Figure 2 consisting of **player roles,** player actions and factual ***domain-related descriptions***, such as positions of involved players.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

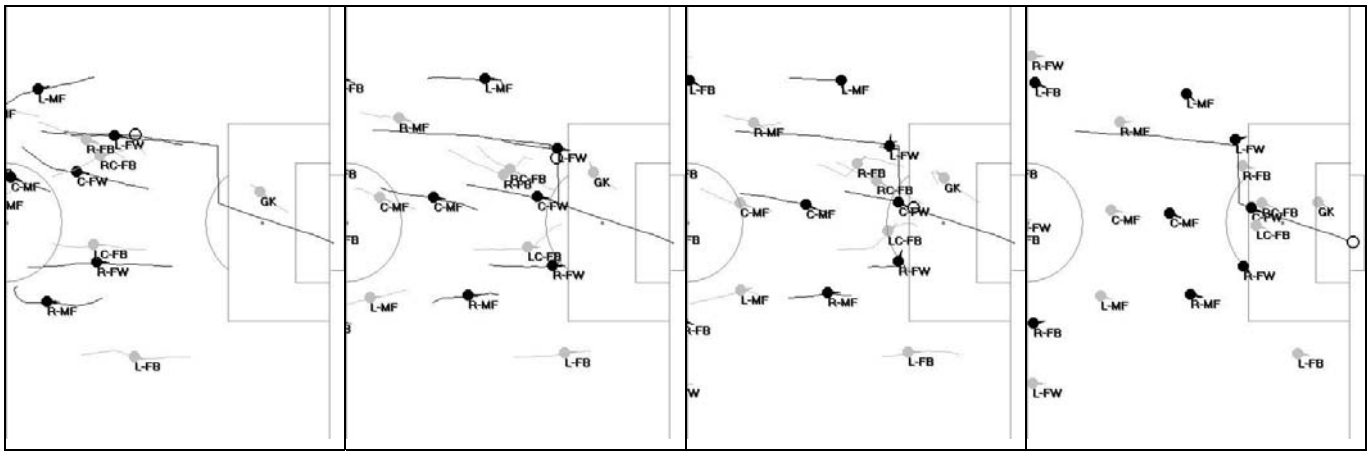


Figure 1: Four frames from a RoboCup game showing a successful direct-attack action sequence.

The diagrammatic action representation is commonly used to help understanding spatial and temporal activity, as shown in Figure 2. Since the aim of MASM is to discover strategic activity and to describe this activity in a human-understandable form, MASDA uses a similar representation, with a diagram, action sequence and rules that describe factual domain features.

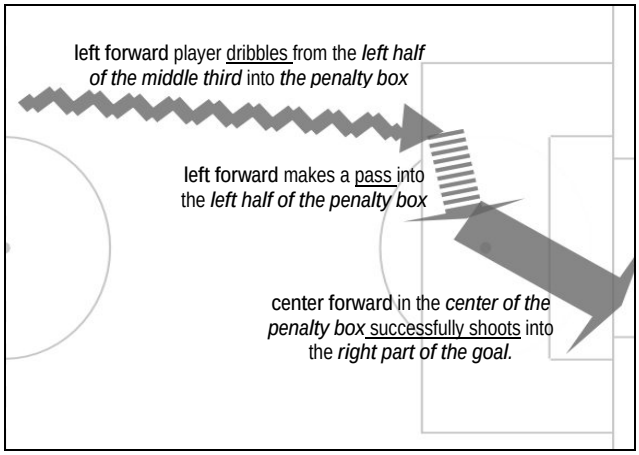


Figure 2: An example of a soccer strategy: a left side direct attack.

This paper is organized as follows. Section 2 reviews the related work. Section 3 presents MASDA. Section 4 describes an example of a generated strategy. In Section 5 an evaluation of the described method is given, followed by discussion in Section 6.

2. RELATED WORK

The task of graphically representing agent actions has been addressed by several multi-agent graphical models, most notably by multi-agent influence diagrams [8] and influence diagram networks [15]. However, all these are concerned with agent decision making, and not agent modeling, and are therefore not suitable for representing observed strategic agent behavior.

The work [11] from Intille deals with the related task of recognizing multi-agent actions in the American football domain. However, the focus in that work is on recognition, using a database of plans, knowledge about football, computation of perceptual features from the trajectories, and propagation of uncertain reasoning, the recognition system developed can determine if the example of a game play is in fact an example of a "p51curl" play. The drawback with their approach is that they explicitly make use of a database of plans and cannot discover new behavior. The paper [5] from Hirano and Tsumoto presents a novel method for finding interesting pass patterns from soccer game records. Taking two features of the pass sequence – temporal irregularity and requirements for multiscale observation – into account, they have developed a comparison method of the sequences based on multiscale matching. The method is used with hierarchical clustering to produce clusters of spatial pass patterns. Experimental results soccer game records demonstrate that the method is able to discover some interesting pass patterns that may be associated with successful goals. This work is concerned primarily with spatial pass patterns while out process also captures important domain-specific knowledge.

There has been extensive research about quantitative modeling in the RoboCup domain. Unfortunately, the problem of constructing human-readable representations of multi-agent activity has received little interest. The work from Raines et al. [13] focused on an automated assistant for analyzing agent teams. Their system ISAAC performs post-hoc, off-line analysis of teams using agent-behavior traces in the domain. The analysis is performed bottom-up, using data mining and inductive learning techniques. ISAAC is able to discover interesting patterns based on user provided features, thus combining the analytical capabilities of the machine with the domain knowledge of the user. Their system is similar to ours in terms of domain knowledge and rule induction. However, our approach explicitly exploits abstraction and provides graphic representation for detected strategies. Work from Kuhlmann et al. [9] deals with the problem of formulating advice-giving as a learning problem. They are able to learn and predict agent behavior from past observations and automatically generate advice to improve its team performance. Although their work concentrates on advice giving task, they successfully construct rules that predict the next significant action events. Our work is

similar in terms of rule induction; however we focus towards increased human comprehension and visual representation as opposed to purely classification accuracy. Work from Kaminka et al. [7] addresses the problem of unsupervised autonomous learning of the sequential behaviors of agents. They translate observations of the complex world state into a time-series of recognized atomic behaviors. The time series are then analyzed to find repeating or statistically significant subsequences characterizing each team. Their work addresses only frequent action subsequence matching while we construct graph-based action model that provides spatial, sequential, and temporal information about agent actions. Kaminka et al. [6] later introduced a behavior-based approach to plan recognition, in which hierarchical behaviors are used to model the observed robots. Efficient symbolic algorithms for answering key queries about observed robots are also presented. However, their approach explicitly utilizes a hierarchical behavior library while our process creates graph-based action models on-the-fly. The work from Sukthankar and Sycara [16] presents a cost minimization approach to the problem of human behavior recognition in a military domain. Using full-body motion-capture data acquired from human subjects, the system is able to recognize the behaviors that a human subject is performing. Low-level motion classification is performed using support vector machines (SVMs) and a hidden Markov Model (HMM); output from the classifier is used as an input feature for the behavior recognizer. Given the dynamic and highly reactive nature of the domain, their system is able to handle behavior sequences that are frequently interrupted and often interleaved. While their approach can identify different behaviors using behavior library our approach detects strategic activity without prior strategic knowledge.

3. THE ALGORITHM

In this section we present MASDA, which is capable of discovering strategic multi-agent activity and describe it graphically and symbolically. The algorithm can be presented as a three-step process: a preprocessing step, a graphical-description generation step and a symbolic-description generation step. Each step is further split into smaller tasks, as presented here, with data structures in parenthesis:

- Step 1: Data preprocessing:
  - Detection of actions in raw data (attributes)
  - Action sequence generation (symbolic sequence)
  - Introduction of domain knowledge (hierarchical concepts)
- Step 2: Graphical description creating:
  - Action graph creation (action graph)
  - Abstraction process (abstract action graph)
  - Strategy selection (graph path)
- Step 3: Symbolic description learning:
  - Action description generation (action description)
  - Rule data generation (set-valued attributes)
  - Rule induction (rules)

The output from each step is the input for the next one. In the first two tasks of the first step MASDA detects actions in raw multi-agent data and generates corresponding action sequences. Detailed description of preprocessing [1] is omitted due to lack of space. The domain knowledge is introduced (task 3 of step 1) in Section 3.1. The process of generating graphical descriptions (step 2) is described in Section 3.2. In Section 3.3 we describe the process of symbolic descriptions generation (step 3).

3.1. Introduction of Domain Knowledge

MASDA exploits domain knowledge through taxonomies of domain concepts. *Taxonomy T* is a hierarchical representation of domain concepts. A concept $x \in T$ is an ancestor of a concept $y \in T$, $x \leftarrow y$, if it denotes a more general concept than the concept y. Hierarchically ordered domain knowledge allows MASDA to travel up and down in the hierarchy to produce more or less abstract descriptions. Specifically, MASDA makes use of:

- a taxonomy of agent roles (T_{roles}), shown in Figure 4
- a taxonomy of agent actions ($T_{actions}$)
- taxonomies of binary domain features ($T_{features}$)

Given a hierarchically ordered structure, we can define the *depth of a concept*. It is the length of the shortest path from the root to the given concept. Consequently, the smaller the depth is, the more abstract the concept is. The root concept has a depth 0.

As agents in MAS can change roles and thus change their behavior [10], agent roles are assigned dynamically during agent activity. Similarly, each detected agent action is assigned its corresponding hierarchical representation. While agent roles and actions are used to describe the activity of agents the domain features are used to describe the truth of some particular domain feature in a domain state space. In our case, domain features are only defined for moving entities - the ball or an agent. They are used to describe agent positions, their relation to the ball and their states. For example, in a RoboCup domain, the feature *HasBall* is true only for the agent that controls the ball (E.g.: *agent1:HasBall*=true) and is false for all other agents (E.g.: *agent2:HasBall*=false). A part of ball-related domain feature taxonomy is shown in Figure 3.

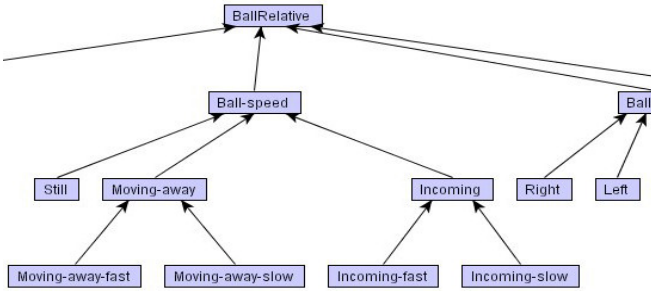


Figure 3: A part of ball-related soccer domain features.

We defined the following three domain taxonomies:

- Position-related taxonomy defines concepts describing whether the ball or an agent is in certain spatial region of the playing field. For example: *left-wing, center-of-the-field, right-wing, left-half, right-half, defending-third, defending-third-danger-zone, defending-third-penalty-box, defending-third-goal-box, middle-third, center-circle, attacking-third, attacking-third-danger-zone,* , etc.
- Speed-related taxonomy defines the speed of an agent and ball: *stationary, moving, fast*, *medium*, and *slow*.

- Ball-related taxonomy defines relations of players regarding the ball, as shown in Figure 3. For example: *has-ball*, *left, right, front, back, incoming, moving-away, incoming-fast, incoming-slow*, *moving-away-fast, moving-away-slow, still, immediate, near, far*, etc.

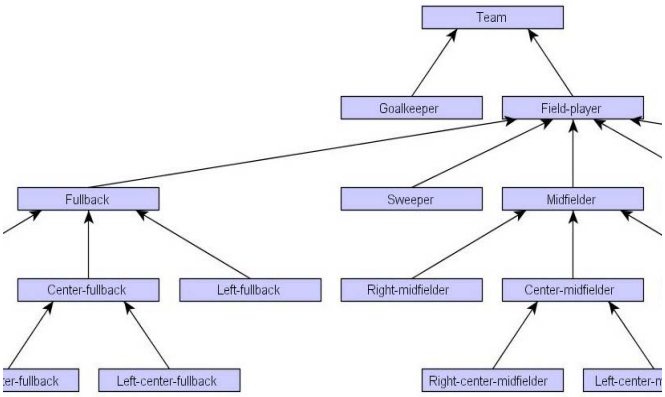


Figure 4: A part of role concept taxonomy.

3.2. Graphical representation

In this section, we describe the process of constructing graphical action representation from the observed action sequence that was generated in the preprocessing step.

3.2.1 Action Graph Representation

An *action graph* (AG) is a directed graph, where connections correspond to actions and nodes represent state space at the beginning of each action. Terminal actions (i.e. the last actions in action sequences) are represented as connections to terminal nodes, which are nodes with no outgoing connections and define the state space at the end of action For example, an action sequence (α, β, γ) is represented as an AG:

$$a \xrightarrow{\alpha} b \xrightarrow{\beta} c \xrightarrow{\gamma} c_{terminal}$$

Node positions correspond to agent positions in a domain space. Each node is also assigned a common action- and role-parent (see Table 1) of outgoing actions from action and role taxonomies. This enables MASDA to generate abstract descriptions of agent role and actions. Each connection also holds start times and durations of the represented actions.

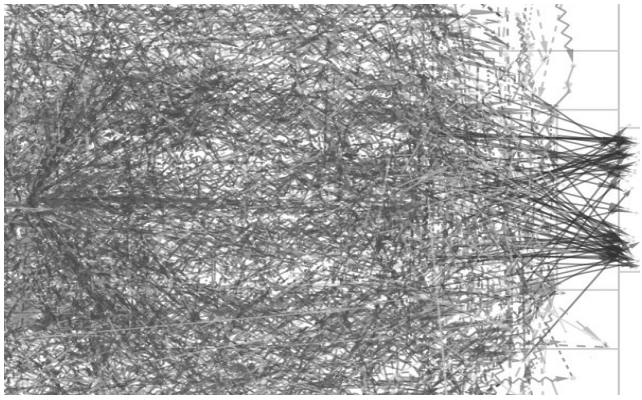


Figure 5: An example action graph.

A more detailed description of action graphs and the construction process is given in [1]. An example action graph, obtained from actions in 10 RoboCup games, is presented in Figure 5.

3.2.2 The Abstraction Process

Complex action graphs with many nodes are difficult to visualize, and are thus difficult to comprehend by humans. An approach to overcome this problem is to reduce the number of nodes. However, one would like to reduce action graph nodes in a manner consistent with captured action concepts.

This can be accomplished with hierarchical clustering of graph nodes. By merging the nearest two graph nodes, we generate a new node that represents common role and action concepts (i.e. the most specific common ancestor of both concepts). The rationale behind the merge process is that actions, frequently occurring near in domain space, define some important strategic concepts. The definition of 'near' thus solely governs how the abstraction process combines action descriptions. Although the distance function is strictly domain-dependent it must utilize conceptual distance between actions, agent roles, and domain positions. Formally, the distance between nodes a and b is a function *distance*(a, b), which must comply with the following three distance axioms [3]:

1. Non-negative definiteness: $\forall$ a, b:
   $dist(a, b) \geq 0 \wedge dist(a, b) = 0$ iff a = b
2. Symmetry: $dist(a, b) = dist(b, a)$
3. Triangular inequality:
   $\forall$ a, b, c: $dist(a, b) \leq dist(a, c) + dist(c, b)$

We defined the distance between graph nodes as a weighted sum of distances between node positions and graph distances between role and action concepts. The weights were assigned by trial and error to produce meaningful results. A more in depth description of the distance function and the whole abstraction process is described in [2].

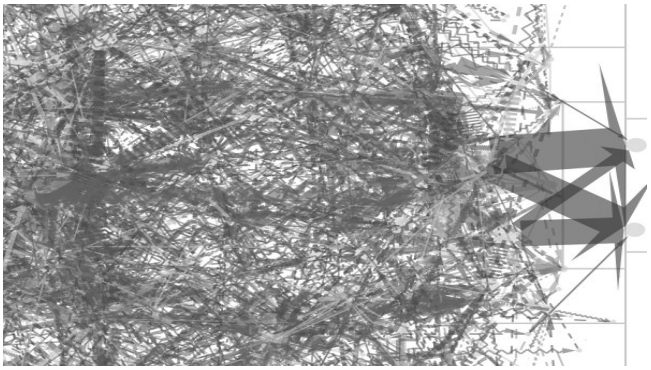


Figure 6: An abstract action graph (AAG_8).

The clustering process results in an *abstract action graph* (AAG), which is an action graph where graph nodes represent more than one agent action. Abstract action graph describes agent behavior in a more abstract way than the original action graph. An abstract action graph, where minimal distance between nodes is greater or equal than some value *dist*, is labeled AAG_{dist}. Such graph can be created with repeated merging of nearest nodes until the minimal distance between nodes is grater than *dist*. An *action description*

of a node in AAG_{dist} is a combination of a node position, corresponding action, and role concepts and a parameter *dist*. AAGs with greater value *dist* represent actions in a more abstract way than AAGs with a lower *dist* value. Therefore, the value of parameter *dist* can be regarded as the degree of abstraction of an AAG. An example abstract action graph AAG_8 is presented in Figure 6. The abstraction process can be stated in a pseudo-code as follows:

```
Input:
- action graph: G
- node distance function: distance(n1, n2)
- target abstraction value: dist

Algorithm:
while (true) { // repeat forever
  //find the nearest nodes
  find nodes a,b ∈ G: min(distance(a, b))
  // check if nodes are near enough
  if (distance(a, b) < dist) { // merge nodes
    create new node c ∈ G
    c.action = common_parent(a.action, b.action)
    c.role = common_parent(a.role, b.role)
    c.instances = a.instances ∪ b.instances
    c.pos = avg(position of agents in
                c.instances)
    remove a, b from G
  }
  else {
    break // stop the abstraction process
  }
}

Output:
- abstract action graph: Gdist
```

3.2.3 Detection of Strategies

When dealing with the observed multi-agent activity it is difficult to decide whether the observed activity represents an instance of a strategic plan or is it simply a result of agent activity based on some local agent decisions. However, multiple and similar agent activities observed over a longer time period tend to be more strategic than those emerged as local decisions, which can be regarded as noise. Therefore we consider repeated and similar action activity as a strategic activity.

In MASDA, similar actions are represented as a single node in an abstract action graph. The amount of similarity is regulated with the value of abstraction parameter *dist*. Multi-agent activity is represented as a path in an AAG. Hence, repeated activity can be discovered by finding adequate number of complete action instance sequences in an AAG path.

An example of a discovered path, together with traces of its action instances, is shown in Figure 7. Although not all actions continue throughout the whole path, there are three action sequences in this example that span the whole path. As this path exhibits repeated and similar agent activity, it is therefore considered as a representation of a strategic activity.

3.3. Symbolic description generation

This section describes the process of generating symbolic description from the obtained graphic representation of strategic activity.

3.3.1 Action Description Generation

We showed that path nodes from the obtained graphic representation of strategic activity represent a sequence of strategic actions. A symbolic action description is generated by visiting the nodes in a path and extracting role and action concepts together with node position and abstraction value. For example, an obtained strategic action description sequence presented in Table 1 corresponds to AAG_8 path shown in Figure 7. The sequence is strategic because it describes a strategic soccer situation - a successful shot. Again, it is expected that a strategy generated from AAG_{dist} with a greater *dist* value is more abstract that the one generated from AAG_{dist} with lower *dist* value.

Action descriptions can be used to create a pattern to detect strategic activity. This is particularly useful when observing unseen agent activity. Namely, if opponent activity matches a starting part of previously constructed strategic action description, it is reasonable to conclude that agents will follow this strategic activity. Consequently, agents can anticipate further opponent actions and devise a suitable counter-activity.

LTeam.FW: Long-dribble	LTeam.FW: Pass-to-player	LTeam.FW: Succesful-shoot	LTeam.FW: Succesful-shoot (End)
(x: 2.3, y: -11.9) *dist*: 8	(x: 39.2,y: -10.8) *dist*: 8	(x: 41.0, y: -4.9) *dist*: 8	(x: 53.6, y:4.5) *dist*: 8

Table 1: A strategic action description from AAG_8

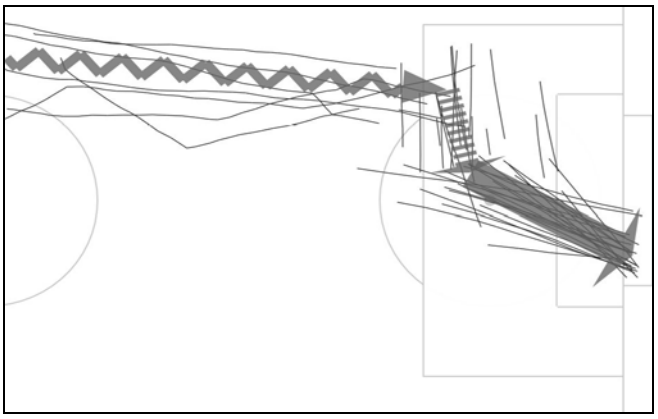


Figure 7: Graphical representation of strategic activity.

3.3.2 Dataset Generation and Rule Induction

As each node defines unique action concept it can be used to generate rules that describe this specific agent action. We generate data suitable for rule induction algorithms as follows. Positive examples are action instances in the target node and negative examples are instances in nearby nodes (i.e. near misses). For each time step we generate all pairs (*agent role, domain feature*), written as *role:feature*, and store only the true ones. However, the standard attribute-value approach is not suitable here. Namely, agents dynamically change roles, and thus it is impossible to generate all *role:feature* pairs for all time steps. Consider an example with only positive examples describing an action X. Internal data representation is presented in Table 2 with its corresponding attribute-value representation shown in Table 3.

	agentA			agentB		
time	role	feature1	feature2	role	feature1	feature2
1	L-FW	T	T	C-MF	F	T
2	L-FW	T	F	C-MF	T	T
3	C-MF	F	T	R-FW	T	T
4	C-MF	F	T	R-FW	T	F

Table 2: Internal data representation

	L-FW		R-FW		C-MF	
time	feature1	feature2	feature1	feature2	feature2	feature2
1	T	T	/	/	F	T
2	T	F	/	/	T	T
3	/	/	T	T	F	T
4	/	/	T	F	F	T

Table 3: Attribute-value data representation with roles instead of agent names

Because *role:feature* pairs are not defined for all times, this representation suffers from poor expressiveness. A more appropriate approach is to define *set-valued attributes* that are attributes whose domains are sets instead of single values. With this approach, each feature corresponds to one set-valued attribute while the values are sets of agent roles, whose corresponding *agent:feature* pair is true. The example data is presented in Table 4.

time	feature1	feature2
1	{L-FW}	{L-FW, C-MF}
2	{L-FW, C-MF}	{C-MF}
3	{R-FW}	{R-FW, C-MF}
4	{R-FW}	{C-MF}

Table 4: Set-valued attribute data representation

Due to hierarchically ordered nature of role concepts, it is possible to automatically generate common parent role concepts, as shown in Table 5. This allows rule inducer to induce rules that describe agent activity in a stricter and more abstract manner. Namely, a rule generated from example shown in Table 4 is *IF (C-MF:feature2)* $\Rightarrow$ *X* while rule generated from example in Table 5 is *IF (FW:feature1* $\wedge$ *C-MF:feature2)* $\Rightarrow$ *X*. The latter rule exhibits more abstract role concept (FW instead of L-FW and R-FW) and is more specific, as it has two features instead of only one as in the previous rule.

time	feature1	feature2
1	{L-FW, FW}	{L-FW, C-MF}
2	{L-FW, C-MF, FW}	{C-MF}
3	{R-FW, FW}	{R-FW, C-MF}
4	{R-FW, FW}	{C-MF}

Table 5: Set-valued attribute data representation with generated common parent role concepts (FW)

4. OBTAINED RESULTS

We generated data for each action obtained from nodes in a discovered strategic path. We used SLIPPER [4], a set-valued rule inducer, with fixed values of parameters to generate corresponding rules. In a typical experiment, 10 games were given as input and several tens of strategic action descriptions were constructed each time. For a chosen description we constructed rules for AAG_1 to AAG_{20}.

The example in Table 6 presents automatically constructed rules generated from AAG_8 action description (see Figure 8).

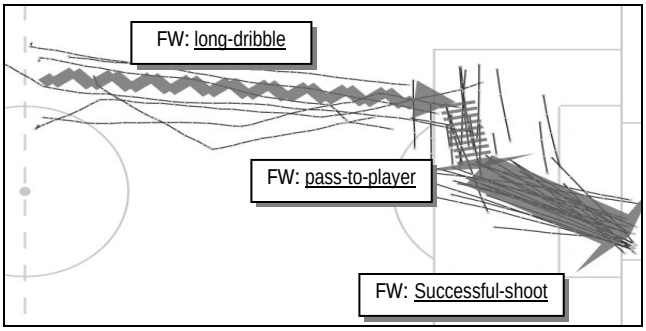


Figure 8: Discovered strategic action description (thick arrows) and original action sequence instances (thin lines).

LTeam.FW:Long-dribble (First frame in Figure 1): RTeam.C-MF:Moving-away-slow $\wedge$ RTeam.L-FB:Still $\wedge$ RTeam.R-FB:Short-distance
LTeam.FW:To-player (Second frame in Figure 1): RTeam.R-FB:Immediate
LTeam.FW:Successful-shoot (Third frame in Figure 1): RTeam.C-FW:Moving-away $\wedge$ LTeam.R-FW:Short-distance
LTeam.FW:Successful-shoot-END (Last frame in Figure 1): RTeam.RC-FB:Left $\wedge$ RTeam.RC-FB:Moving-away-fast $\wedge$ RTeam.R-FB:Long-distance

Table 6: Rules describing discovered strategy

At a first glance the rules seem a bit uninformative. However, expert analysis revealed that the rules do describe important strategic information:

- LTeam.FW:Long-dribble

RTeam.C-MF:Moving-away-slow: The right team's center-midfielder was occupied with previous ball owner and was for a while moving away from the ball. Consequently, he stayed behind the action.

RTeam.L-FB:Still: This player is at the bottom of the first frame in Figure 1 moving adjacent to the ball. In terms of relative movement to the ball he appears still. His movement is defensive, but he fails to aim to directly intercept or cover any of the attackers.

RTeam.R-FB:Short-distance: The right team's right fullback player unsuccessfully tried to intercept the dribbler. He consequently stayed at short distance behind the ball. Humans would probably favor the equivalent *RTeam.R-FB:behind* feature.

The three clauses successfully point out three main reasons for the success of the attack.

- LTeam.FW:To-player

RTeam.R-FB:Immediate: This rule essentially states that a pass occurred because right team's right fullback came too close to the ball. A ball owner was threatened by the opponent so he passed the ball to the teammate, who was left uncovered.

- LTeam.FW:Successful-shoot

RTeam.C-FW:Moving-away: Since shots represent a very fast ball movement, players are moving away from the ball. The rule inducer therefore just picked one feature among many equivalents. This is essentially the same as *ball:fast* which would have been far more intuitive.

LTeam.R-FW:Short-distance: there are many agents at short ball distance (*LTeam.L-FW, LTeam.R-FW, RTeam.L-FB, RTeam.LC-FB, RTeam.RC-FB, and RTeam.R-FB*). As rule inducer considers shorter rules to be better than longer, it only included one such feature instead of more, as would be expected by humans.

It is evident that rule inducer is not able to choose the most appropriate human-favored features. One reason for this is in a small number of learning examples and consequently substantial number of equivalent *agent:feature* pairs for each situation. As rule inducer has no meanings to favor one feature over another, it simply picks one. However, expert analysis showed that this nevertheless results in meaningful rules describing actual causal strategic relations. In terms of human comprehension, it appears that rule inducer would greatly benefit from some sort of domain-dependent knowledge about features. We consider the construction of such feature rating mechanism as our future work.

5. EXPERIMENTAL EVALUATION

We evaluated MASDA on 10 RoboCup games played during SSIL [14]. A leave-one-out strategy was used to generate 10 learning tasks. A pre-determined 3-action strategy, shown in Table 1 and in Figure 7, was used as a reference and was generated from all 10 games, for abstractions AAG_1 to AAG_{20}. For each learning task, a strategy was generated on 9 games and tested on the remaining game, again for AAG_1 to AAG_{20}. Tests measured the difference between action descriptions, the quality of generated rules for all three actions and joint use of rules together with action descriptions. One learning or test example is one 3-action sequence which is positive if it fits the reference strategy description. If the positive test example fits the learned strategy then this example is considered a true positive, otherwise it is considered a false negative. The results presented in Table 8 are averaged over 10 tests where the x-axis is the level of abstraction (parameter *dist* for AAG_{dist}).

The results in Table 8a indicate that the accuracy of action descriptions is approximately constant with the abstraction. However, the accuracy of rules quickly increases until it peaks at *dist*=4 and then slowly decreases.

This is expected because for lower abstractions, nodes represent only a few action instances, which is insufficient for rule inducer to generate good rules. This is different with high *dist* values where nodes represent quite different action concepts, thus producing more abstract and less accurate rules. Accuracy when using both rules and description follows rule accuracy for lower abstraction values, since rules are too specific. With higher abstraction values (>14) overall accuracy follows description accuracy since rules become too general.

When measuring recall, all the three test scenarios give similar results as shown in Table 8b. That is, the quality of classifying true cases increases to *dist*=3 then remains steady until *dist*=14 when it quickly drops. The similar phenomenon is observed when measuring precision, shown in Table 8c. This can be explained by the generated agent behavior being too abstract.

The last test was performed to verify if the abstraction process really generates more abstract descriptions. For this test we measured the abstraction of generated rules, defined as an average feature depth in feature taxonomy for features used in rules. The lower the depth, the more abstract the rule is. The results, presented in Table 7, clearly show that average feature depth is negatively correlated with the abstraction parameter *dist*. This confirms that as abstraction value increases the rules contain more abstract features. Since rules were generated from node data, we can safely conclude that generated strategies and AAGs exhibit the same principle: that the AAG with higher *dist* value presents more abstract action activity than the one with lower *dist* value.

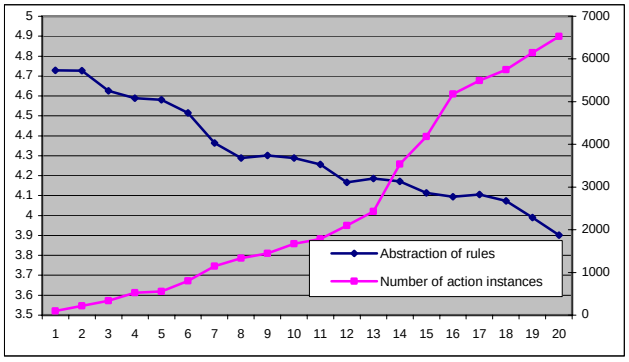
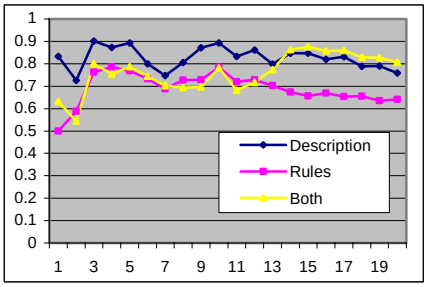
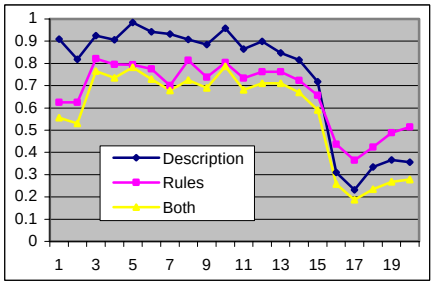


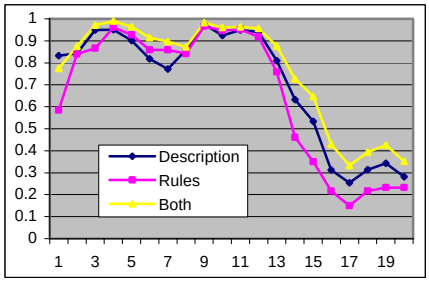
Table 7: The abstraction of generated rules and number of action instances in one strategy



a) Accuracy



b) Recall



c) Precision

Table 8: Results. The x-axis is the degree of abstraction (parameter *dist*).

6. DISCUSSION

We have designed and implemented MASDA as a general domain-independent framework for discovering strategic behavior of multi-agent systems. Domain-specific knowledge was introduced in the form of role, action and domain feature taxonomies. We assume that changing a domain is a straightforward task that would require changing specific domain-knowledge of a similar form. We believe that there is a wide range of possible domains that can be exploited by MASDA since its essence is stepwise abstraction in the domain-space.

The experiments indicate that the approach is practically useful in terms of accuracy, recall and precision. Our tests also confirm that the abstraction process in fact generates more abstract descriptions of agent activity.

There are some open questions that need to be addressed. First, MASDA was evaluated only on the RoboCup domain. Although the authors believe that no major problem should emerge when introducing another domain, this should be verified in practice. Second, while the discovered rules seemed sensible to the soccer expert, they were rather unintuitive for non-experts to comprehend. The rule discovery process would greatly benefit from some sort of feature relevance function, based on domain-knowledge that would favor human-desired features. This kind of knowledge should be provided by a domain expert.

7. ACKNOWLEDGEMENT

The authors would like to thank Gal Kaminka for helpful discussion and thoughtful comments.

REFERENCES

[1] A. Bezek: *Modeling Multiagent Games Using Action Graphs*. Proceedings of Modeling Other Agents from Observations (MOO 2004), 2004.

[2] A. Bezek: *Discovering Strategic Multi-Agent Behavior in a Robotic Soccer Domain*, Proceedings of AAMAS 05, 2005.

[3] J. Calmet, A. Daemi: *From Entropy to Ontology*. Proceedings of Fourth International Symposium From Agent Theory to Agent Implementation, 2004.

[4] W. W. Cohen and Y. Singer: *A simple, fast, and effective rule learner*. Proceedings of the sixteenth national conference on Artificial intelligence, pp.: 335 - 342, Orlando, United States, 1999.

[5] S. Hirano in S. Tsumoto: *Finding Interesting Pass Patterns from Soccer Game Records*. The Eighth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2004), 2004.

[6] G. Kaminka in D. Avrahami: *Symbolic Behavior-Recognition*. Proceedings of Modeling Other Agents from Observations (MOO 2004), 2004.

[7] G. Kaminka, M. Fidanboylu, A. Chang, in M. Veloso: *Learning the sequential coordinated behavior of teams from observations*. Proceedings of the RoboCup-2002 Symposium, 2002.

[8] D. Koller and B. Milch: *Multi-Agent Influence Diagrams for Representing and Solving Games.* Games and Economic Behavior, 45(1), pp. 181-221, 2003.

[9] G. Kuhlmann, P. Stone, and J. Lallinger: *The UT Austin Villa 2003 Champion Simulator Coach: A Machine Learning Approach.* In Daniele Nardi, et al., editors, RoboCup-2004: Robot Soccer World Cup VIII, Springer Verlag, Berlin, 2005.

[10] R. Nair, M. Tambe in S. Marsella: *Role allocation and reallocation in multiagent teams: Towards a practical analysis*. Proceedings of the second International Joint conference on agents and multiagent systems, 2003.

[11] S. S. Intille: *Visual Recognition of Multi-Agent Action*. PhD Thesis, MIT, 1999.

[12] I. Noda et. al: *Overview of RoboCup-97*. In Hiroaki Kitano, editor, RoboCup-97: Robot Soccer World Cup I, pp. 20-41. Springer Verlag, 1997.

[13] T. Raines, M. Tambe, and S. Marsella: *Towards automated team analysis: a machine learning approach*. Third international RoboCup competitions and workshop, 1999.

[14] RoboCup 2004: *RoboCup Simulation League.* RoboCup04 world cup game repository, http://carol.science.uva.nl/~jellekok/robocup/rc04/, 2004.

[15] T. Steffens et. al: *RoboCup Special Interest Group on Multi-Agent Modelling.* http://www.cogsci.uni-osnabrueck.de/~tsteffen/mamsig/, 2001.

[16] G. Sukthankar and K. Sycara: *A cost minimization approach to human behavior recognition.* Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, The Netherlands, ACM Press, 2005.

[17] D. Suryadi and P. J. Gmytrasiewicz: *Learning models of other agents using influence diagrams*. Proceedings of the Int. Conference on User Modeling, pp. 223-232, 1999.

On the Response of EMT-based Control to Interacting Targets and Models

Zinovi Rabinovich Jeffrey S. Rosenschein
School of Engineering and Computer Science
The Hebrew University of Jerusalem
Jerusalem, Israel
{nomad, jeff}@cs.huji.ac.il

ABSTRACT

A novel control mechanism was recently introduced based on Extended Markov Tracking (EMT) [9, 10]. In this paper, we present a study of its response to multiple interacting control goals. We show a simple extension that can be integrated into EMT-based control, and which provides it with the ability to handle several behavioral targets. Experimental support for the validity of this extension is provided. We also describe an experiment with a simulated robot, where EMT-based controllers interact and interfere indirectly via the environment. Experiments support the resilience of multiagent EMT-based team control to potential conflicts that may appear within a team.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Control Theory*;
I.2.9 [**Artificial Intelligence**]: Robotics;
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents, Multiagent Systems*

General Terms

Algorithms, Performance

Keywords

Control, Extended Markov Tracking, Markovian Environment, Autonomous robots and robot teams

1. INTRODUCTION

Imagine a car moving quickly along a highway; a bend in the road approaches, and the car's autopilot considers possible actions. It faces two interacting and potentially interfering considerations: first, the car has to turn fast enough to fit the road's curvature, or it will run off the road; second, a car has to turn slowly enough to maintain its grip on the road, or it will spin out of control. Furthermore, not only its radial speed, but also its linear speed affects the car's successful negotiation of the bend in the road, and the autopilot has to consider (or at least respond to) the environment-enforced interaction between these two motion parameters. That is, the autopilot faces both explicit and implicit multi-objective control problems.

These can be solved using a variety of methods. For example, proper control strategy can be achieved through learning (as, for instance, in [5]). Depending on the controller representation, and if some basic control capabilities are available, behavior-based (e.g., [1, 2]) or expert-based (e.g., [4, 12, 8]) approaches can also be used to achieve the required balance. Both, however, have their limitations. For example, the expert-based approach provides a fluent merger of basic advice from multiple sources, but concerns itself only with this merger and does not dictate how the basic advice is produced. The behavior-based approach, on the other hand, explicitly states different basic behaviors and their interrelationships with respect to action selection, and is capable of complex behavior composition. Unfortunately, each basic behavior is usually obtained by specific controller design and specialization. This results in explicit and simple controllers, but sacrifices the controller's universality and reuse, in case another behavior has to be included.

In this paper we consider the multi-criteria problem when basic control is performed using a control scheme based on Extended Markov Tracking (EMT) [9, 10]. EMT-based control provides a means to achieve a complex behavioral goal. We use a scaled additive mechanism directly integrated into EMT-based control to combine rationales for action with respect to multiple behavioral targets. This creates a fluent merger out of a well-defined underlying action choice mechanism, and we carry out experiments to test this new multi-target EMT control in an explicit multi-objective control scenario.

We then applied EMT-based control to implicit multi-objective control. We created a simulated robot setting using the Player/Stage simulator [6], in which an EMT-controlled robot had to follow a target. In this experiment, linear and radial speeds of the robot were controlled by two independent EMT-based controllers, which forced them to solve an implicit multi-objective problem, since the robot's motion correlates the two speeds, and thus the two controllers. In addition, we complicated the setting by providing controllers with inexact environment models. EMT-based control demonstrated strong resistance to multiple sources of system noise, and successfully handled the scenario.

The rest of the paper is organized as follows. Section 2 briefly describes the formal setting of Extended Markov tracking and a control scheme based on it, with Subsection 2.1 describing our extension of the additive combination mechanism to EMT-based control. We proceed in Section 3 to the experiments and experimental results of a simple multi-target problem. Section 4 describes our

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

experiment with the simulated robot scenario of target following. We discuss in Section 5 our results and future work.

2. EXTENDED MARKOV TRACKING AND CONTROL

EMT-based control [9, 10] is a recently introduced framework for control of a stochastic environment. It relies on the fact that the system environment has a Markovian model of the form $< S, s_0, A, T, O, \Omega >$, where

- S is the set of all possible environment states;
- s_0 is the initial state of the environment (which can also be viewed as a distribution over S);
- A is the set of all possible actions applicable in the environment;
- T is the environment's probabilistic transition function: a mapping $T : S \times A \rightarrow \Pi(S)$. That is, $T(s'|a, s)$ is the probability that the environment will move from state s to state s' under action a;
- O is the set of all possible observations. This is what the sensor input would look like for an outside observer;
- Ω is the observation probability function: a mapping $\Omega : S \times A \times S \rightarrow \Pi(O)$. That is, $\Omega(o|s', a, s)$ is the probability that one will observe o given that the environment has moved from state s to state s' under action a.

Extended Markov Tracking (or EMT for short) then uses this environment model to follow the system development, and maintains two estimators. First is the regular estimator for this type of environment — the state estimator, or state beliefs, expressed by a distribution vector over the system states $p_t \in \Pi(S)$. The second is the system dynamics estimator $PD : S \rightarrow \Pi(S)$, which can be seen as a stochastic function that "explains away" the changes in the system state beliefs seen so far.

To keep track of the actual system development, EMT dictates that its estimators be updated as follows:

- For state beliefs a Bayesian update is used. If at time t an action $a \in A$ was taken and then an observation $o \in O$ received, the new state beliefs p_{t+1} would be expressed by the formula:

$$p_{t+1}(s) \propto \Omega(o|s, a) \sum_{s'} T(s|a, s')p_t(s')$$

- For the system dynamics estimate EMT uses the Kullback-Leibler distance [3]

$$D_{KL}(p\|q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

to perform a conservative update based on the following convex optimization problem:

$$PD_{t+1}(\cdot|s) = \arg \min_{Q(\cdot|s)} \langle D_{KL}(Q(\cdot|s)\|PD_t(\cdot|s))\rangle_{p_t(s)}$$

$$s.t.$$

$$\forall s' \;\; p_{t+1}(s') = \sum_s Q(s'|s)p_t(s)$$

$$\forall s \;\; \sum_{s'} Q(s'|s) = 1$$

That is, an explanation is sought for the change in system state beliefs from p_t to p_{t+1}, similar (as much as possible) to the old dynamics estimate PD_t. This update is denoted for short as $PD_{t+1} = H(p_{t+1}, p_t, PD_t)$.

Despite the fact that the optimization problem contained within the dynamics estimate update looks difficult, it has rather attractive properties. First, it is a convex optimization problem over a convex domain, which means that it can be solved efficiently. Second, a numerical iterative solution exists, tailored specifically for this type of optimization problem — iterative proportional fitting [7].

EMT-based control's approach to estimation has been embedded within a simple greedy control algorithm that is summarized below. The algorithm employs a reference system dynamics $r : S \rightarrow \Pi(S)$, termed the *ideal system dynamics* or *tactical target*; the control algorithm greedily selects actions so as to modify the estimated system dynamics to concur with the reference. This approach is similar to closed-loop differential control [11].

0. Initialize estimators:
   - the system state estimator $p_0(s) = s_0 \in \Pi(S)$,
   - system dynamics estimator
   $$PD_0(\bar{s}|s) = prior(\bar{s}|s)$$
   Set time to $t = 0$.
1. Select action a^* to apply using the following computation:
   - For each action $a \in A$ predict the future state distribution $\bar{p}^a_{t+1} = T_a * p_t$;
   - For each action, compute
   $$D_a = H(\bar{p}^a_{t+1}, p_t, PD_t)$$
   - Select $a^* = \arg\min_a \langle D_{KL}\left(D_a\|r\right)\rangle_{p_t}$
2. Apply the selected action a^* and receive an observation $o \in O$.
3. Compute p_{t+1} due to the Bayesian update.
4. Compute $PD_{t+1} = H(p_{t+1}, p_t, PD_t)$.
5. Set $t := t + 1$, goto 1.

2.1 Multi-Target EMT-based Control

At times, there may be several behavioral preferences. For example, in the case of multi-robot movement in formation, two preferences on motion direction exist — one dictated by keeping in formation, the other by obstacle collision avoidance. Successful formation navigation requires a robot to adhere to, and balance, both of these behaviors. For EMT-based control, this would mean facing several tactical targets $\{r_k\}_{k=1}^K$, and the question becomes how to merge and balance them. A balancing mechanism can be applied to resolve this issue.

Note that EMT-based control, while selecting an action, creates a preference vector over the set of actions based on their predicted performance with respect to a given target. If these preference vectors are normalized, they can be combined into a single unified preference. We thus replace stage 1 of EMT-based control (the action selection stage) by the following:

1. Given a set of tactical targets $\{r_k\}_{k=1}^K$, and their corresponding weights $w(k)$, select action a^* based on the following computations:
   - For each action $a \in A$ predict the future state distribution $\bar{p}^a_{t+1} = T_a * p_t$;

- For each action, compute
$$D_a = H(\bar{p}^a_{t+1}, p_t, PD_t)$$
- For each $a \in A$ and r_k tactical target, denote
$$V(a,k) = \langle D_{KL}\left(D_a \| r_k\right)\rangle_{p_t}.$$
Let $V_k(a) = \frac{1}{Z_k} V(a,k)$, where $Z_k = \sum_{a \in A} V(a,k)$ is a normalization factor.
- Select $a^* = \arg\min_a \sum_{k=1}^{k} w(k) V_k(a)$

The weight vector $\vec{w} = (w_1, ..., w_K)$ allows the additional "tuning of importance" among tactical targets without the need to redesign the targets themselves. This balancing method is also seamlessly integrated into the EMT-based control flow of operation, and is compatible with its multi-agent extension, as explored in [10]. This compatibility makes possible the following experiment.

3. MULTIPLE TARGETS IN A SCALE BALANCING EXPERIMENT

Originally, EMT-based multi-agent control was tested for its team coordination capabilities over a small balancing problem involving two agents. In this problem, two agents of equal mass are standing on a long bar resting with its ends on two equal springs (Figure 1).

At each time step of the system, each agent has the choice of three actions: moving left one step, moving right one step, or staying put. Every movement of an agent has a non-zero probability of failing, and the probability is biased by the inclination of the bar. That is, an uphill motion will have less probability of succeeding than if the bar were level, and downhill motion will have greater probability of succeeding than if the bar were level. Notice that bar inclination depends on the current agent positions on the bar, thus creating a correlation between the effects of the agent actions, and providing for implicit information transfer between the agents.

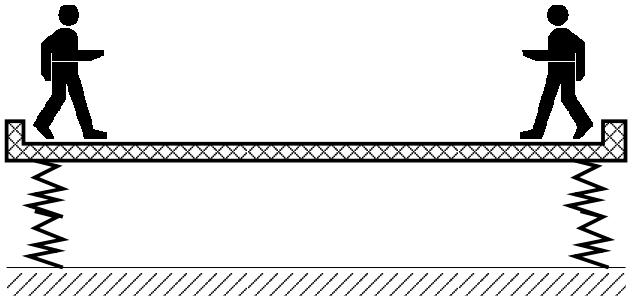


Figure 1: Springed bar setting

Formally, the system state is described by the positions of the two agents on the bar, $S = [1 : d_{max}]^2$, where d_{max} is the length of the bar in "steps", and the initial state is an unbalanced one, so that $s_0 = (1, \frac{d_{max}}{2} + 1)$. The action sets are $A_i = \{left, stay, right\}$, and the transition probability is built according to the physics of motion as previously described.

In this paper, we consider only one observation scenario among the original two: both agents receive independent noisy observations about their joint position. That is, $O_i = S = \{$all positions of the two agents$\}$, $\Omega_1 = \Omega_2$ and creates uniform noise over the immediate neighborhood of the real joint position of agents.

However, unlike the original paper [10], we have modified action selection to accommodate multiple targets, as described in Section 2.1. We then set the agents two conflicting targets. One is to balance the springed bar, while the other is to maintain a preset distance between themselves. Note that these targets are not requirements of the system state, but of the laws governing its behavior. For instance, distance maintenance is expressed by a dynamics matrix, that shifts any given state into one that possesses the correct distance property, and EMT-based control sets out to achieve this kind of law within the system.

Although the two behavioral targets we have chosen are compatible, that is, there exists a position of agents on the bar that satisfies both, the targets are indeed conflicting and interfering. For example, assume that we want agents to be at a distance of 4 from each other. Denote by 0 the coordinate line at the center of the bar, and assume that the system's noisy response forced the agents into positions -2 and $+3$. In this case, the balancing target can encourage the motion of the left agent from -2 to -3 thus balancing the bar, but violating the distance constraint. On the other hand, the distancing target could be satisfied by the same agent moving right to -1, violating bar balancing even further.

In our experiment sets, we tested both the reaction of EMT-control to multiple targets, and to their balancing by weight vector $\vec{w}$. The bar length was set to 14, so that the leftmost position on the bar would have coordinate 1 and the rightmost position would have coordinate 15, and the agents were also required by the second target to keep the distance between them at 4 steps.

Despite the constant conflict between the two targets, multiagent EMT-based control equipped with our multi-target action selection approach managed to maintain both targets quite closely. Although relentless system noise caused fluctuations, as seen from an example run at Figure 2, these fluctuations occurred around the only common position that satisfies the demands of both targets. In fact, the mean values of distance between agents and the position of the center of mass almost perfectly matches the ideal, as can be seen from the value distributions in Figure 3, which in our experiments would be 4 and 8 respectively.

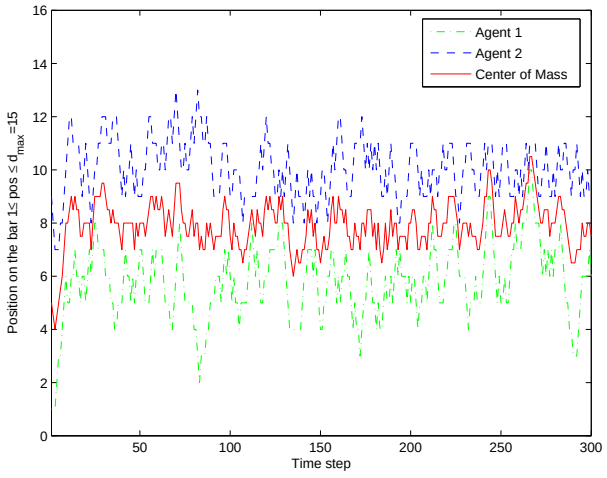


Figure 2: Example Run of Dual-Target Springed-Bar Problem

As was mentioned above, our experiments included different balancing vectors $\vec{w}$, e.g., the distribution in Figure 3 was obtained from the weight vector $\vec{w} = (0.2, 0.8)$, that is, setting the balancing target at 0.2 and the distancing target at 0.8. Changing the weights vector to be $\vec{w} = (0.4, 0.6)$ exposes an interesting property of multi-target EMT-based control. Since the targets were weakly

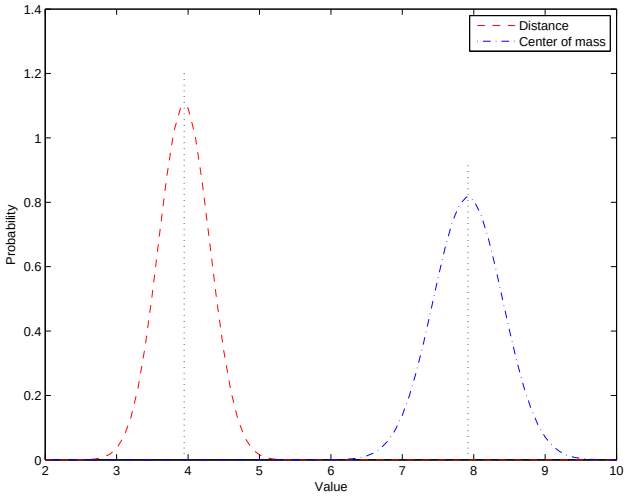


Figure 3: (Fitted normal) Distribution of distance and center of mass in dual-target springed-bar problem with (0.2, 0.8) balancing

compatible, the algorithm maintained both targets with the new balancing vector, as it did with the old one. The difference occurred when the control algorithm had to correct system behavior in response to noise — the algorithm was more ready to briefly deviate from the distancing target than the balancing one, as shown by the distributions in Figure 4.

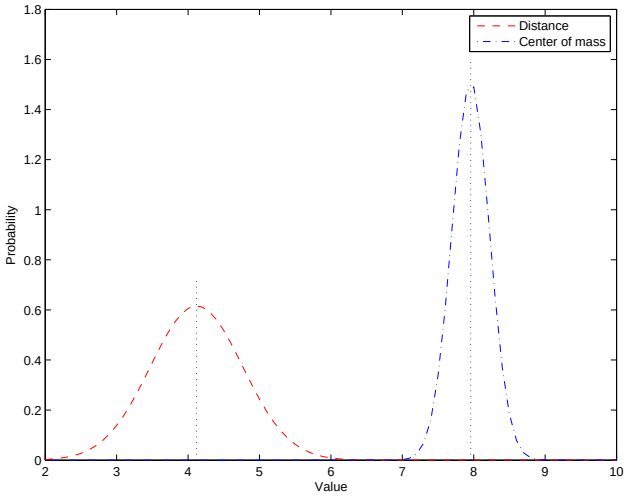


Figure 4: (Fitted normal) Distribution of distance and center of mass in dual-target springed-bar problem with (0.4, 0.6) balancing

However, as the weight of the the balancing target increased, not only the variance of the distance between the agents changed, but also the mean. EMT-based control began to lean strongly towards the balancing target, almost abandoning the distancing target. This can be seen clearly from Figure 5, which depicts changes of the distance and center of mass distributions mean, as a function of weight of the target.

Note that the balancing target had a very strong presence at the weight 0.4, as expressed by the mean and variance of the center

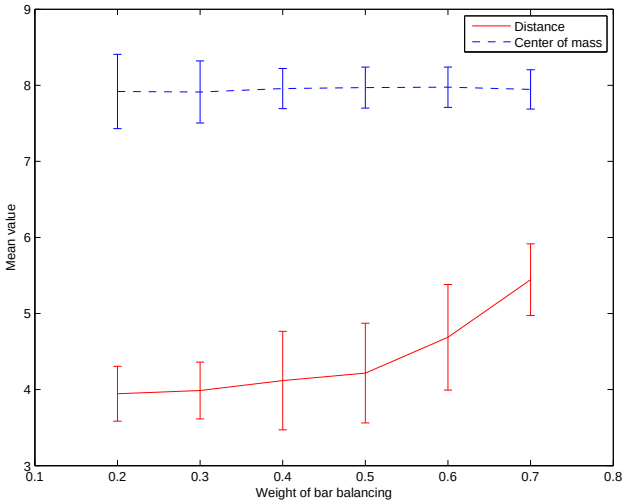


Figure 5: Means of distributions with respect to the weight of the balancing target. Error bars depict variance.

of mass distribution. On the other hand, the distancing target at the weight of 0.4 had much less attention from EMT-based control. This inequality in attention to targets with respect to symmetric weight vectors was at first surprising, but can be explained. It has to do with the strength of preference as expressed by the two targets used.

While constructing the targets for the EMT-based controller, a simple inverse exponential scheme was selected to produce the preferences. The balancing target was constructed to have inverse exponential preference over the center of mass deviation from an ideal, while the distancing target was composed based on deviation from the distance ideal. However, simple parameter selection for this scheme produced two targets with different *strength* of preference. While the balancing target included preferential differences of dozens of orders of magnitude, the distancing target featured preference differences of only 2–4 orders of magnitude.

As a result, any change in system dynamics had much higher impact with respect to the balancing behavioral preference, which thus had a stronger effect on proper system behavior. The distance tactical target was very mild in comparison, and thus had less influence on action choices.

4. SIMULATED OBJECT TRACKING EXPERIMENTS

The experiments above with multiple targets were the first step in testing EMT-based controllers for inter-controller interference. It is possible and even necessary to consider situations where separate controllers interact via the environment, interfering with one another. This situation often occurs naturally in robotics, and we chose to test EMT-based control in that domain.

4.1 Robot Motion with EMT Control

Computerized algorithmic control of a robot, unlike low-level reactive control, requires a discretization of sensory information received by the robot, and later the carrying out, by the robot's actuators, of a discrete action description. In effect, the actual robot serves as an interface between the real world and a discrete controller, transforming sensory information to fit the internal environment model used to make decisions.

Two adjustments are required, in order to apply EMT control to robotic motion.

If the internal model is chosen to be a Markovian model, $< S, A, T, O, \Omega, s_0 >$, then sensory discretization requires mapping real world sensations onto the observation set O. This makes the set of states S abstract — it is only indirectly connected to the real world, and all $s \in S$ gain meaning through the observation probability function Ω.

Abstraction of S has several consequences. First, a control algorithm based on a Markovian model would usually require the transition function $T : S \times A \rightarrow \Pi(S)$, also called the "action model," to be strongly coherent with the environmental dynamics of the system state's *meanings*, determined by O and Ω. However, the EMT Controller manages to achieve reasonable, though not optimal, results even with an inaccurate action model (Section 4.2).

Second, if EMT Control is to be applied, its strategic layer has to create a tactical target by "reverse engineering" — a high-level target is described by limitations on *observations*, and the strategic layer uses Ω to convert it into system state dynamics, $r : S \rightarrow \Pi(S)$.

Other than the two adjustments above, the basic EMT Controller remains unchanged for robot control.

4.2 Target-Following Experiment

We designed a straightforward simulation experiment, within the Player/Stage simulation environment [6], to see whether EMT Control would be effective for robot control, and whether it would be able to cope with implicit controller interference and an inaccurate environment model.

In the experiment, two independent EMT Controllers, $EMTC_1$ and $EMTC_2$, were applied to linear and rotation speed modulation of a single (simulated) Pioneer-2X robot, with the task of following another robot's motion. The sensory information was received through a *blob finder* — an on-robot camera with basic image analysis that makes possible the detection of color blobs within the picture. Camera information was approximately mapped onto the observation sets: color blob relative area and centering within the picture. Thus, the observation distributions provided state meanings of linear distance for $EMTC_1$, and angular distance for $EMTC_2$.

To follow (but not capture) an object, a robot must solve two balancing problems: stay within a certain distance from the object, and stay directed straight at it.

A well-studied formal balancing problem exists that can easily be adopted for robotic controllers: the Drunk Man Walk (DMW). This problem concerns a weakly-controlled random walk over a linear graph, as depicted in Figure 6. The actions in this model tilt the probability of left and right steps, while the observation distribution blurs the current position.

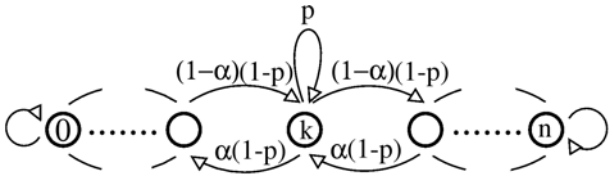


Figure 6: Drunk Man state transition diagram. α is subject to action effect.

Since EMT-based control was shown to be successful at solving this problem [9], we chose to use the DMW as the internal model for the $EMTC_1$ and $EMTC_2$ controllers. Their balancing acts translated into a tactical target in a natural way, essentially projecting all states in S_i onto $s = \frac{|S_i|}{2}$.

Notice, however, that the transition function of a DMW (our action model) is not entirely coherent with the real world's reactions, and with the meaning of system states. This incoherence actually serves our purposes, since we want to verify the performance of EMT-control with respect to an imperfect model. For example, $EMTC_1$ uses system states S_1 to represent distance from a followed object, and since their meaning is based on the camera picture, they represent an irregular set of distances. This lack of regularity would dictate uneven transition probabilities from different states under the same action, but this is not the case in the DMW model, where all system states are symmetric. A similar problem exists within $EMTC_2$, because change in the visible angle of an object may depend on its linear speed as well as on its angular speed. However, this is not accounted for by the DMW model within $EMTC_2$.

This incoherence indeed influences the EMT Controller's performance; however, it was still able to successfully perform the tracking task. A sample run of the EMT-controlled robot can be seen in Figure 7, depicting three positions of the robots at different times. In this run, the followed robot (which we shall call the *prey*) performed a constant loop, and the EMT-controlled robot that followed it (the *predator*) managed to capture this motion.[1] The predator traces a smaller loop, concentric with the one traced by the prey. However, due to action model incoherence, the predator did not perform optimally and reacted to the change in the prey's position with insufficient correcting actions. Effectively, it resulted in the predator being significantly further away from the prey than was required by the tactical target.

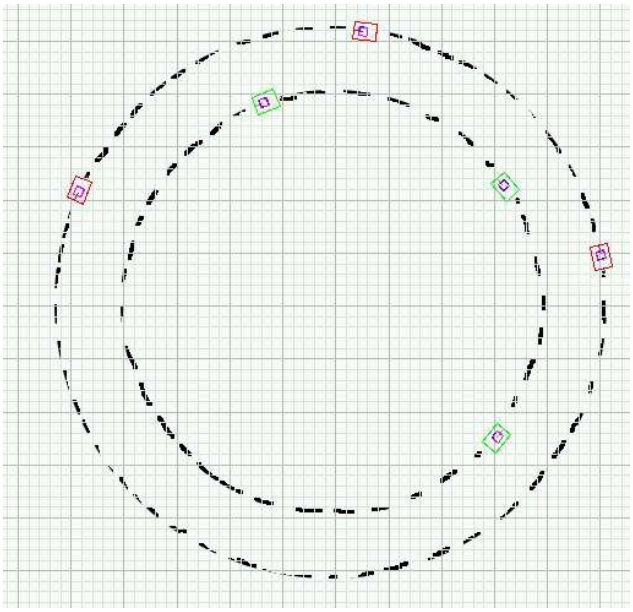


Figure 7: Target Following with a Weakly Coherent Model

5. DISCUSSION AND FUTURE WORK

In this research we explored the performance of EMT-based control in response to multiple interacting targets, as well as interacting environment models. In the latter, EMT-based control was applied

[1]Predator/Prey terminology is used here for naming convenience only, since there is no actual 'capture' intended.

in an experimental setting with the additional complication of an incoherent environment model.

For EMT to handle explicit multiple objectives, we introduced an additive weighing scheme that integrated directly into EMT operations, and allowed it to select actions in a well-balanced way. In fact, in our experiments EMT control managed in many situations to achieve both targets, despite their interference and, at times, conflict with one another.

Our results also showed that EMT-based control is sensitive to the strength of its tactical targets, and actions are selected in correspondence to that strength. As the proposed weighing scheme affects EMT's attention to targets as well, we conclude that it can and should become a part of an overall target design process, allowing for finer tuning of the combined behavior.

The simplicity with which tactical behavioral targets can be balanced within EMT-based control suggests further development of dynamic balancing schemes, guided, perhaps, by some external global system behavior evaluation. For example, in our car driving scenario above, it might be the successful passing of a bend in the road.

As our weighted balancing scheme is closely related to other multi-objective algorithms, it is important to continue research into the possibility of their integration. Since EMT-based control includes a prediction mechanism, expert-based balancing approaches (such as [4, 12, 8]) would be of particular interest. Insights from multi-objective reinforcement learning (e.g., [5, 2]) could also be used.

EMT sensitivity to preference strength can be further exploited by introducing non-linear balancing and weight schemes, in effect amplifying preferences as needed, without complicating tactical target design.

The proposed multi-target approach, combined with the natural abilities of EMT control with respect to implicit interference of multiple controllers, can be readily used in a multi-robot setting, quickly providing a sustainable behavior of a team. Formation restricted navigation, multi-featured manipulator control, multi-robot sweeping, and many other tasks become susceptible to the use of EMT with the aid of our multi-target approach, and we plan to explore these applications in the future.

6. ACKNOWLEDGMENT

This work was partially supported by grant #039-7582 from the Israel Science Foundation.

7. REFERENCES

[1] R. C. Arkin. *Behavior-Based Robotics*. MIT Press, 1998.

[2] O. Buffet, A. Dutech, and F. Charpillet. Learning to weigh basic behaviors in scalable agents. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 3, pages 1264–1265, 2002.

[3] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley, 1991.

[4] Y. Freund, R. E. Schapire, Y. Singer, and M. K. Warmuth. Using and combining predictors that specialize. In *Proceedings of the 12th Annual ACM Symposium on the Theory of Computing (STOC-97)*, pages 334–343, 1997.

[5] Z. Gabor, Z. Kalmar, and C. Szepesvari. Multi-criteria reinforcement learning. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, pages 197–205, 1998.

[6] B. Gerkey, R. T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR-03)*, pages 317–323, 2003.

[7] S. Kullback. Probability densities with given marginals. *The Annals of Mathematical Statistics*, 39(4):1236–1243, 1968.

[8] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

[9] Z. Rabinovich and J. S. Rosenschein. Extended Markov Tracking with an application to control. In *The Workshop on Agent Tracking: Modeling Other Agents from Observations, at the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 95–100, New York, July 2004.

[10] Z. Rabinovich and J. S. Rosenschein. Multiagent coordination by Extended Markov Tracking. In *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 431–438, Utrecht, The Netherlands, July 2005.

[11] R. F. Stengel. *Optimal Control and Estimation*. Dover Publications, 1994.

[12] V. Vovk and C. Watkins. Universal portfolio selection. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT-98)*, pages 12–23, 1998.

Multiple UAV Task Allocation using Negotiation

P.B. Sujit
Department of Aerospace Engineering
IISc, Bangalore, India
sujit@aero.iisc.ernet.in

A. Sinha
Department of Aerospace Engineering
IISc, Bangalore, India
asinha@aero.iisc.ernet.in

D. Ghose
Department of Aerospace Engineering
IISc, Bangalore, India
dghose@aero.iisc.ernet.in

ABSTRACT

A multiple UAV search and attack mission involves searching for targets in an unknown region, followed by attack on the detected targets. An effective mission involves assigning the tasks to UAVs efficiently. Task allocation becomes difficult when the UAVs have limited range to detect the targets and neighbouring UAVs. The UAVs are also subject to limited communication range. With these constraints, allocating tasks efficiently to UAVs become difficult. In this paper, we propose a negotiation scheme that efficiently allocates tasks for multiple UAVs. The performance of the negotiation based task allocation is studied on a battle field scenario. We study the effect of sensor ranges on the task allocation and compare the results with that of greedy strategy and a variation of the negotiation mechanism. The results show that the negotiation based task allocation mechanism performs far better than the greedy strategy. Negotiation with target information exchange between neighbours performs better than without target information exchange, when the sensor radius is low.

Keywords

Task allocation, Negotiation, Multiple agents, UAV search and attack tasks

1. INTRODUCTION

Unmanned aerial vehicles (UAVs) are being extensively used for military purposes, like search, surveillance and as munitions in the battlefield [1] - [10]. They play a crucial role in information gathering from hostile and unknown regions. These UAVs can also be used as munitions to search, attack and destroy targets in the unknown region. The UAVs used for these applications may have limited capabilities and may not have the required stealth capability and ammunition payload to complete the task single-handedly. Hence, there is a necessity for such UAVs to be deployed in swarms. A desirable feature for these UAV swarms is to have autonomous decision making and coordinating capability. As the UAVs perform search, they may find several targets in the search region. The number of UAVs may be more than the number of targets available or the number of targets available may be more than the number of UAVs. In either case, we need an efficient task allocation method for assigning the UAVs to the targets. The classical solution for task allocation problem is to have a centralized task allocation system that generates the necessary commands for the UAVs. But, centralized task allocation system have well known limitations and do not address scalability issues well. Hence, there is a necessity to develop a decentralized task allocation algorithm. This algorithm must be suitable for implementation in a multiple agent UAV swarm. It should also be scalable, and consume low computational overhead. An efficient task allocation strategy should have the ultimate objective to complete the mission (that is, destroy all targets) in minimum time by cooperating and coordinating with other UAVs [1]. Cooperation can be achieved by communication with neighbouring UAVs, explicitly or implicitly. Here, we use negotiation mechanism to develop such a decentralized task allocation algorithm for multiple UAVs that communicate with each other for decision-making while performing search and attack tasks in an unknown region.

The UAVs that we consider are small in size, have limited fuel capacity (limited flight time), and limited sensing capabilities. The UAVs can detect the presence of neighbouring UAVs and targets within its sensor range only. The UAVs communicate with their neighbours and make their decisions based only on the information they receive from their sensors. All the UAVs are assumed to be homogeneous, have constant speed, and have no turn radius constraints. The UAVs have to carry out the mission within the given flight time. Collision avoidance between UAVs is also not an issue.

The UAVs can perform search, attack, and speculative tasks. Search task refers to searching for targets in the unknown region. Once a target is found by the UAV, it executes a speculative task to ensure that the target is a real target and not a false target. The target that has been verified as a real target is attacked by the UAV. We assume that once a target is attacked, it is destroyed and hence battle damage assessment task on the target is not necessary to be performed. However, it can be incorporated in the formulation fairly easily.

The speculative task is performed in the following way: Once a UAV finds a target, it estimates the value of the target. This value is used for negotiating with the neighbouring UAVs for assignment of the target.

Each UAV performs decision-making independently, based

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

on the estimate of the target status. The decision taken by the UAV also depends on the number of neighbouring UAVs. Here, we assume that the UAVs do not have sufficient memory to remember the path it has travelled.

1.1 Literature

Task allocation of UAVs is an active research area for the past few years. Nygard et al. [1], propose a network flow optimization model for allocating UAVs to targets. The network optimization problem is formulated as a linear programming problem to obtain decisions for allocating the UAVs. The authors assume that global communication between UAVs exists. The network flow model has been further extensively studied by Schumacher [2, 3, 4] for wide area search munitions with variable path lengths, assignment with timing constraints and path planning. Chandler et al. [5, 6], explore various other techniques like iterative network flow, auctions, linear programming, and mixed integer linear programming, for multi-UAV task allocation. The effect of communication delays on the task allocation using the iterative network flow model is dealt with in Mitchell [7]. Curtis [8] presents a task allocation methodology for simultaneous search and target assignment, where the search and task assignments are posed as a single optimization problem. Turra et al. [9] present a task allocation algorithm for multiple UAVs performing search, identification, attack, and verification tasks in an unknown region for targets that move in real time. These authors also address the problem of obstacle avoidance. Jin et al. [10] propose a probabilistic task allocation scheme for the scenario presented in [5, 6].

In most of the algorithms presented in the papers cited above, global communication between agents is assumed. So, information obtained by an agent is assumed to be communicated to all the other agents. However, the task allocation decision algorithm is autonomously executed by the agents. Sujit et al. [11] present a team theoretic approach that allows the UAVs to perform decision-making independently when there is no communication between UAVs. The UAVs can only sense the location of their neighbours and targets. The task allocation is performed based on a linear programming formulation, where the cost associated with each target is estimated with some probability. Rajnarayan and Ghose [12] pose a multiple agent search problem as a problem in the team theory framework and propose optimal solution to the problem of determination of search regions.

Recently, market based approaches have shown some considerable increase in performance of task allocation strategies for multiple agent applications. Dias and Stenz [13, 14, 15] introduce a novel approach for coordinating robots based on the free market architecture in economics. The approach defines revenues and cost functions across the possible plans for executing a specified task. The task is accomplished by decomposing it into sub-tasks and allowing the robots to bid and negotiate to carry out these sub-tasks. Gerkey and Mataric [16] use an auction mechanism for multi-robot coordination. Although there are many cooperative strategy algorithms available for multiple agents, very little attention has been paid to the role of communication which plays a vital role in the development and performance of a search strategy. Usually, the communication between agents is quantified by the amount of information that flows between them. The communication and computational complexity involved in multi-robot task allocation are analyzed in [17]. Mataric et al. [18] develop various task allocation strategies and study their performance on a multi robot application with sensor noise. The simulation study is compared with the results obtained using experiments. Gurfil [19] also uses auctions mechanism for task allocation among multiple UAVs performing search and destroy mission. Lagoudakis et al. [20] uses auction algorithm for assigning unexplored tasks to group of mobile robots. The paper also provides some theoretical bounds on the computational complexity of the proposed algorithm.

In this paper, we present a task allocation algorithm for multiple UAVs performing search and attack tasks in an unknown region using negotiation scheme for the scenario given in the next section. This is one of the very few applications available that exploits the use of negotiation for a network of UAVs involved in a practical problem of decision-making.

1.2 Problem Scenario

Consider a planar search space consisting of an unknown number of targets. The location of the targets are not known *a priori* to the UAVs. A search and destroy mission would be undertaken by sending a fleet of UAVs to search a region and destroy as many targets as possible within the flight endurance time of the UAVs. The task involved in such missions are search and attack. A search task is to search the environment for targets and, if found, engage a UAV to attack the target. The UAVs have to coordinate among each other so as to accomplish the mission efficiently and in minimum time. The UAVs have limited sensor and communication range. The UAVs can sense the exact position of all the targets and its value within its sensor range. The location of all the neighbouring UAVs within a UAV's sensor region are also assumed to be detected. The UAVs should perform task allocation so that as much of the mission as possible is completed. The task allocation mechanism we use is based on negotiation.

2. PROBLEM FORMULATION

Consider N UAVs/agents performing a search and destroy operation on a bounded region consisting of M targets whose exact positions are not known *a priori*. The basic problem of task allocation is to assign agent $A_i \in N$, to target $m_i \in M$, efficiently such that the mission is completed as fast as possible. The task allocation problem can be solved either using a centralized controller or a decentralized controller. In the former case each agent communicates the information it has to the centralized controller that solves a task allocation algorithm and assigns each agent to a particular task. However, implementing this task allocation strategy in real-time requires large communication overhead and may not be scalable with increase in number of agents and targets. Also, these strategies are not robust to failures. Hence, a decentralized task allocation strategy is necessary for implementing on a multi-agent system. The decentralized task allocation can be performed by any one agent in the multiple agent system. This agent receives the necessary information from all the agents and solves a task allocation algorithm and broadcasts the assignments to the agents. Another way of implementing a decentralized task allocation would be that, each agent broadcasts its information to all the other agents and hence each agent has the required information to solve the task allocation problem independently and assign a task for itself. The implementation of this task allocation

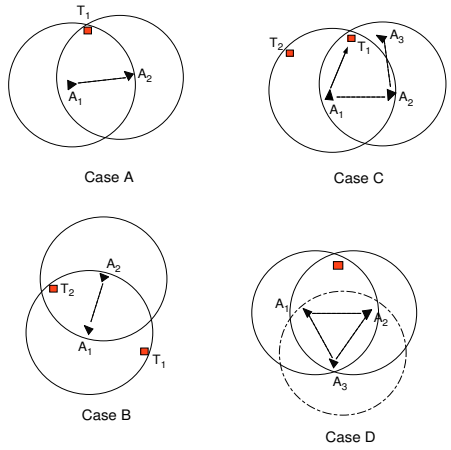


Figure 1: Some scenarios for decision-making

strategy requires large amount of communication among the agents.

The implementation of decentralized task allocation with finite communication range poses some challenging problems. For instance, consider Case A from Figure 1 where agent A_1 and A_2 have target T_1 in their sensor range and an allocation has to take place as to which agent should be assigned to the target. The task allocation can be done using a greedy strategy, in which case both the agents would move towards the same target which is not desirable. Another task allocation mechanism used in multi-robot literature is based on auctions. But in this case since the system is decentralized then each agent would become am auctioneer and hence both the agents auction the same target. A modification to the standard auction algorithm may eliminate some of the difficult issues, however this would complicate the decision-making rules for multiple agents using auction mechanism locally.

Consider Case B, wherein A_1 has T_1 and T_2 in its sensor range while A_2 has T_2 only. Auction requires broadcast of all the target and their associate costs. Resolving conflicts using auctions is a difficult task. In Case C, we can see that A_1 sees T_1 while A_3 is already on its way to attack T_1. So, A_1 wastes some resource in moving towards a target that is already assigned, Since the communication is limited it does not have access to the assignment of other agents. Instead of T_1 it could have attacked T_2. Here also greedy and auction algorithm would not yield good performance. In Case D, agent A_3 gets the auction information from A_1 and A_2 about T_1, now A_3 does not know to which agent it has to send the bid. These complications in using auctions for limited communication cases, give rise to the use of negotiation as a tool that can handle these situations efficiently. In cases A, A_1 and A_2 can negotiate which agent would be assigned to target T_1. While in case B, A_1 and A_2 can negotiate such that one agent attacks T_1 and the other moves towards T_2. In case C, A_2 can detect a conflict between A_1 and A_3 and send decisions such that A_1 or A_3 moves towards T_1. However in Case D, A_3 actually negotiates between A_1 and A_2 which are not neighbours and detects an early conflict and hence provides an efficient task allocation decision and has better fuel usage as only one agent can continue search to select other targets instead of moving to T_1.

However, the implementation of negotiation scheme involves designing of negotiation rules over which the process of decision-making takes place. In the next section we describe the negotiation scheme employed in decision-making.

At every time step each agent has to perform a task. The task can be search or attack a target. Each agent senses its environment consisting of other agents and targets. An agents' assignment for a task depends on four different situations. These situations are dependent on the availability of neighbouring agents and targets. The four situations, in which agent A_i has to perform a task and make a decision are:

1. No targets and no neighbours

   Task: Search

   Decision: Continue to move in the same direction

2. No targets but has neighbours

   Task: Perform search or attack. The target information may be provided by the neighbouring agents.

   Decision: Acts as a negotiator for neighbouring agents

3. Targets are present but no neighbours

   Task: Attack

   Decision: Select a target that yields maximum value

4. Target as well as neighbours are present

   Task: Search or attack

   Decision: Negotiate with neighbours

Once an agent A_i is present within a distance d from the target, we assume that the target is destroyed. The agents have to negotiate with its neighbouring agents for an efficient task allocation. The agents are not subjected to any turn radius constraints and hence can move in any direction. The agents have to maximize the number of targets destroyed in the search space by coordinating with its neighbouring agents through negotiation.

3. DECISION-MAKING

3.1 Negotiation as a tool to handle uncertainty in agent actions

In general, negotiation refers to the communication process that facilitates coordination and cooperation among a group of agents [22]. In multi-agent systems, its aim is to resolve problems related to resource allocation and task assignments between various agents in a decentralized setting.

Our approach is somewhat similar to Rubinstein's model of strategic negotiation [21] where agents make proposals that are either accepted or rejected by other agents; and whether an agent implements its proposal or not depends on what other agents do. However, our approach is different from Rubinstein's model on many counts due to the nature of the task allocation problem. Unlike most negotiation models we do not have a situation where each proposal is vetted by all the other agents. In fact, due to the connectivity restrictions we have a network of agents where an agent is not necessarily directly connected to all other agents. So, each agents decision is based on the response of

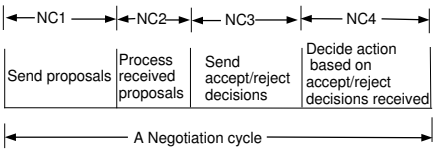


Figure 2: Negotiation cycle

only those agents that are connected to it. Moreover, unlike in Rubinstein's model, agents make simultaneous offers at pre-defined decision epochs and the actions are accordingly distributed between agents. Another way in which our model differs from Rubinstein's model is due to the fact that, in a task allocation problem the need for negotiation arises mainly because of lack of information about the action of other agents. So, the whole process of negotiation is geared towards determining the action of an agent in a coordinated autonomous fashion without assuming any kind of hierarchy or priority among agents.

A coordinated decision by an agent would be one that is not in conflict with the decision of its neighbors. There is no conflict except that which arises due to uncertainty of agent actions. For example, it occurs when more than one agent is planning to attack the same target, thus decreasing the effectiveness of the mission. Resolution of such conflicts can be effected either by

(i) Direct communication/negotiation as in the case when an Agent A_i and another agent A_j are within communication range.

(ii) Indirect negotiation when an Agent A_i and another agent A_j, $A_j \notin \mathcal{N}(A_i)$ want to attack the same target T, and A_i and A_j are connected through a sequence of communication links through other agents. For instance, they may be connected through a third agent A_k with $A_j \in \mathcal{N}(A_k))$ and $A_k \in \mathcal{N}(A_i)$.

In the first case, since A_j is within the communication range of A_i, it can exchange information with A_j and resolve the conflict. While, in the second case, A_i does not know about the existence of A_j, and so direct communication is not feasible. So, the intermediate agents are important in the negotiation process. In the negotiation scheme developed next, we will show that it is the neighboring agents who contribute to the decision-making of agent A_i.

3.2 Negotiation scheme

Each agent A_i performs the following actions during decision-making: (i) Sends/receives proposals (ii) Processes received proposals and sends Accept/Reject decisions to proposing agents (iii) Computes own route decision (iv) Implements decision. All these actions happen within each negotiation cycle. This is shown in Figure 2. Note that an agent A_i that has no targets will have the second segment only, while the agents that have targets as well as neighbouring agents will have all the four segments of decision-making.

The different segments of the negotiation cycle are described below:

3.2.1 Send/receive proposals (NC1)

Each agent evaluates the benefit associated with each target. Let $b_i(T_j)$ be the expected benefit A_i gets by attacking target T_j, which is given by

$$b_i(T_j) = V_{T_j} w_r \ - \ S_t \tag{1}$$

where, V_{T_j} = value of target T_j, w_r = the weight given to search task over the task of attacking a target, S_t = (time to reach the target T_j)/(total flight time). The benefit set $\mathcal{B}_i$ of A_i consists of benefits for all the tasks an agent has. Let $\mathcal{T}_i$ be the set of all targets. The benefit set for agent A_i is represented as:

$$\mathcal{B}_i = \{b_i(T_j) \mid T_j \in \mathcal{T}_j\} \tag{2}$$

Agent A_i chooses a target T_{S_i} for which A_i gets the maximum value, as

$$S_i \ = \ \arg\max_j \{b_i(T_j) \in \mathcal{B}_i\} \tag{3}$$

The proposal of agent A_i, sent to its neighboring agents, is of the form $Q_i = (A_i, T_{S_i}, b_i(T_{S_i}))$, containing the proposer agent's identification, proposed target, and the value associated with T_{S_i}.

3.2.2 Processing received proposals (NC2) and send decisions (NC3)

Let $\mathcal{Q}_i$ be the set of proposals received by agent A_i from its neighbors A_j, including its own proposal.

$$\mathcal{Q}_i = \{(A_j, C_{S_j}, \beta_j); L(A_j) \in \mathcal{N}(L(A_i), q_c)\}$$

Let T_k^i be a target that appears in at least one of the proposals received by A_i. That is, $T_k^i = T_{S_j}$ for some $Q_j \in \mathcal{Q}_i$. For each such T_k^i, define $\mathcal{A}(T_k^i)$ as the set of agents that have proposed T_k^i, and $\mathcal{B}(T_k^i)$ as the set of values associated with agents in $\mathcal{A}(T_k^i)$. So,

$$\begin{aligned} \mathcal{A}(T_k^i) &= \{A_j \mid Q_j \in \mathcal{Q}_i, T_{S_j} = T_k^i\} \\ \mathcal{B}(T_k^i) &= \{b_i(T_j) \mid A_j \in \mathcal{A}(T_k^i))\} \end{aligned} \tag{4}$$

Using the above two sets ($\mathcal{A}(T_k^i)$) and $\mathcal{B}(T_k^i)$, agent A_i sends *accept* or *reject* decision to its neighbors using the following rules:

Rule 1: An agent A_i sends *accept* to agent A_j, if

$$\mathcal{A}(T_k^i) = \{A_j\} \tag{5}$$

That is, $\mathcal{A}(T_k^i)$ is a singleton containing only agent A_j (note that A_j could be A_i itself).

Rule 2: If $\mathcal{A}(T_k^i)$ is not a singleton then agent A_i sends *accept* to that agent in $\mathcal{A}(T_k^i)$ which has the maximum value by attacking target T_k^i and *reject* to all other agents in $\mathcal{A}(T_k^i)$. That is, *accept* is sent to $A_{j'} \in \mathcal{A}(T_k^i)$ if,

$$j' = \arg\max_j \{b_i(T_j) \in \mathcal{B}(T_k^i)\} \tag{6}$$

Note that Rule 2 subsumes Rule 1. But they are stated separately for clarity. Again $A_{j'}$ can be A_i itself.

Rule 3: An agent can send only one *accept* for one target. If there are more than one j' then the agent selects one of them.

Rule 4: For A_i to decide on its action at the current search step it has to get *accept* from all its neighboring agents to which it had sent its proposals.

Rule 1 implies that when an agents' proposal is not in conflict with other agents' proposals an *accept* can be sent

without considering other agents' decision. When more than one agent proposes to attack T_k then there is a conflict between the proposing agents which A_i has to resolve. The conflict can be resolved by comparing the benfits' proposed by the agents. Agent A_i compares the $b_i(T_j)$ received for target T_k and sends *accept* decision to an agent A_k which has the highest $b_i(T_j)$ and *reject* decisions to the remaining agents. An agent A_i can receive a mix of *accept* and *reject* decisions from its neighbors. If we allow the agent to attack a target T_k, since it has got acceptance from some of the agents, this assignment would cause ineffective performance as multiple agents will get assigned to the same target. Hence, *Rule 4* guards against agents getting multiple assignment. The Rules 1-4 are the key to the negotiation scheme. While implementing *Rule 3*, we may encounter situations where more than one agent has the same $b_i(T_j)$, in which case we use a deadlock resolution scheme that resolves such deadlocks.

3.2.3 Computing route decision (NC4)

Agent A_i decides whether to implement or discard its proposed task based on the *accept* or *reject* decisions received from its neighbors. The agent implements its proposal if it receives *accept* decisions from all its neighbors and discards it if the agent receives a *reject* from even one of its neighbors. An agent that received a *reject* for its proposal from at least one neighbor will go on to the next negotiation cycle and this process will continue till it receives all *accept* decisions. An agent that has arrived at a decision (after receiving accept from all its neighbors) will not send any more proposals during subsequent negotiation cycles. The sequence of negotiation cycles will terminate automatically when all the cycles have converged to a decision. Later we will prove that only a finite number of negotiation cycles are necessary.

When an agent A_i receives reject for all its proposals, it adopts the search task.

3.3 Additional target information exchange

The agents that have received acceptance to their proposal may have other targets within its sensor range. An agent A_i can send this information to its neighbouring agents who can use it. The information that an agents sends is the target location and its value as perceived by A_i. This information will be more useful for those agents that may not have decided any targets but are neighbours of A_i. The target information broadcast by A_i can also be useful if all the proposals of agent $A_j \in \mathcal{N}(A_i)$ are rejected.

Once an agent receives the available targets from agent A_i, it can make assignment to any of the targets based on random number generation, greedy strategy, or start a negotiation with its neighbouring agents for obtaining an assignment. In this paper we use greedy strategy for simplicity.

3.4 Deadlock resolution mechanism

We define a deadlock, when an agent A_i is unable to decide to whom it has to send an acceptance. This situation can happen when more than one agent, with the same $b_i(T_j)$ value, seeks target T_j to attack. Since the $b_i(T_j)$ values are same, use of *Rule 2* is not possible and agent A_i cannot send acceptance to all the agents as that will violate *Rule 3*. There are two possible ways of resolving deadlock: loss information and token algorithm.

Loss information: In this scheme, agent A_i requests for more information from agents in $\mathcal{A}(T_k^i)$. This additional information will aid in effective decision-making. The additional information that an agent requests is the value of possible loss that each proposing agent suffers if it chooses the next best action instead of the proposed action. Let the new benefit vector for agent A_k be $\hat{\mathcal{B}}_k$ and the loss λ_k be evaluated using (7) as,

$$\hat{\mathcal{B}}_k = \{\mathcal{B}_k \setminus b_i(C_{S_K})\}$$
$$\lambda_k = \max \mathcal{B}_k - \max \hat{\mathcal{B}}_k \quad (7)$$

where, $'\setminus'$ denotes set difference. When agent A_i requests for loss information, the loss λ_k is sent to agent A_i. Let Λ_i represent the set of loss information received from all the agents in $\mathcal{A}(T_k^i)$. An *accept* is sent to an agent A_j that satisfies the condition in (8) and *reject* is sent to the remaining agents.

$$A_j = \arg\max_i(\Lambda_i) \quad (8)$$

Suppose there are multiple $b_i(T_j)$'s that are at the next highest level, then the same procedure needs to be repeated. Using the loss information does not guarantee that the deadlock will be resolved. This situation can arise when multiple agents have the same loss value. In that case, we use a token algorithm as given below.

Token Algorithm: Every agent A_i carries a unique token number K_i. Whenever the above situation (of the loss being equal) occurs wherein the agent is unable to decide to whom it has to send acceptance, the agent requests for token number of the agents $A_k, A_k \in \mathcal{A}(T_k^i)$. Let $\mathcal{K}_i$ be the set of token numbers received by agent A_i. Agent A_i compares these token numbers and chooses an agent A_j with the least token number.

The token number K_j is increased by a number $\hat{N}$, where $\hat{N}$ is an arbitrary large number greater than N. Agent A_j updates its token number to $K_j + \hat{N}$, after receiving the acceptance from agent A_i. This scheme ensures that an agent that has been selected earlier in this situation, will not be selected again in a similar situation if there is at least one other agent which has not been selected before.

Whenever a situation occurs when many agents are proposing for the same target, using the rules specified and the deadlock resolution mechanisms, it is guaranteed that there would be at least one agent that gets acceptance from all its neighboring agents.

3.5 Some theoretical results

THEOREM 1. *If more than one agent is proposing a target T_j, then at least one of the agents will receive all acceptances from its neighbors.*

Proof: Let $\mathcal{A}(T_j^i)$ be the set of agents proposing target T_j as their proposal. Then, by *Rule* 2, agent A_i sends an *accept* decision to agent A_j which has the maximum $b_i(T_j)$. If there are multiple agents with same $b_i(T_j)$ then A_i invokes the deadlock resolution mechanism by which one agent would receive an *accept*. □

THEOREM 2. *The negotiation terminates in a finite number of negotiation cycles.*

Proof: From Theorem 1 we observe that, at each negotiation cycle, at least one of the agents gets all *accept* and so decides

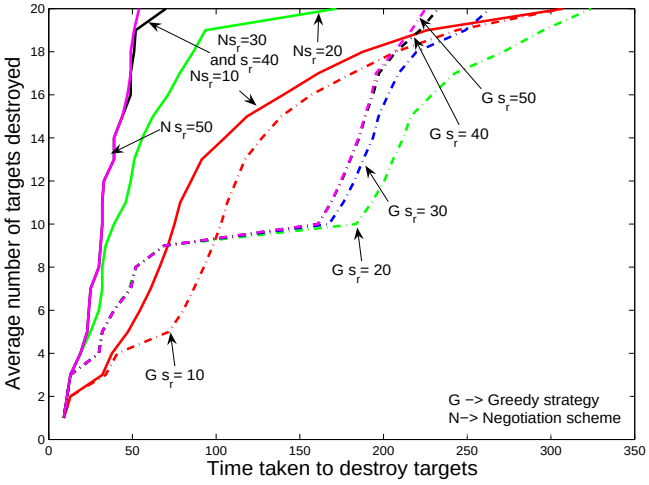


Figure 3: Average number of target hits for 100 different target positions

upon a target for its next step. Since there are a finite number of agents, in a finite number of negotiation cycles each agent would decide upon a target to attack. And if the target are not available then they choose search task. Hence, all the agents would decide upon a task in a finite number of negotiation cycles. The maximum number of negotiation cycles an agent can go through is N. □

4. SIMULATION RESULTS

A simulation study is conducted on a battle field scenario of size 100×100. Through these simulations we show that the negotiation scheme performs better than greedy strategy in terms of average number of targets destroyed. The simulation is carried out using 7 UAVs for 100 different sets of target positions with each set having 20 targets. The *a priori* knowledge about number of targets present in the space and their initial positions are not available to the UAVs. We also study the performance of negotiation and greedy schemes for various sensor radius.

From Figure 3 we can see that the negotiation scheme outperforms the greedy strategy. The number of targets using negotiation scheme is higher and also the time taken to accomplish the mission is comparatively low. An expected result of increase in performance with increase in sensor range can be seen for the performance curves of negotiation scheme in the figure. However, this intuitive result is not true for greedy strategy for all the sensor radius.

As we can see from the figure the performance of greedy strategy with sensor radius $s_r = 10$, fares better than higher sensor radius $s_r = 20$ to $s_r = 50$. This is due to the fact with low sensor radius, the UAVs are unable to the sense the targets initially and hence move in the initial heading direction (spreading out). But, with higher sensor radius the agents are able to sense the target from their initial positions and hence all the UAVs move in the direction of sensed target as a swarm. Hence, the performance is worse when compared to lower sensor radius.

We carried out another set of simulations to study the performance of task allocation algorithm for different target

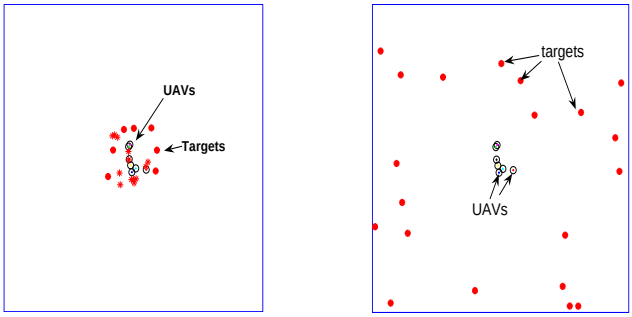


Figure 4: Battle field with 20 targets for proximity factors $\rho = 0.625$ and $\rho = 0.11$, while the sensor radius $s_r = 10$.

distributions on the search space. In order to conduct these experiments we define a proximity factor that determines the nature of the distribution or spread of targets in the search space. The proximity factor is defined as:

$$\rho = \frac{S_r}{\frac{1}{N}\sum_{i=1}^{N}\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2}} \quad (9)$$

where N is number of targets, (x_i, y_i) represents the position of the ith target location, (x_c, y_c) the mean of all the target positions and S_r the sensor radius. Low proximity factor implies well separated targets compared to the sensor radius. While higher proximity factor ensures that the targets are placed very closely. Figures 4 show different target distributions in the search space.

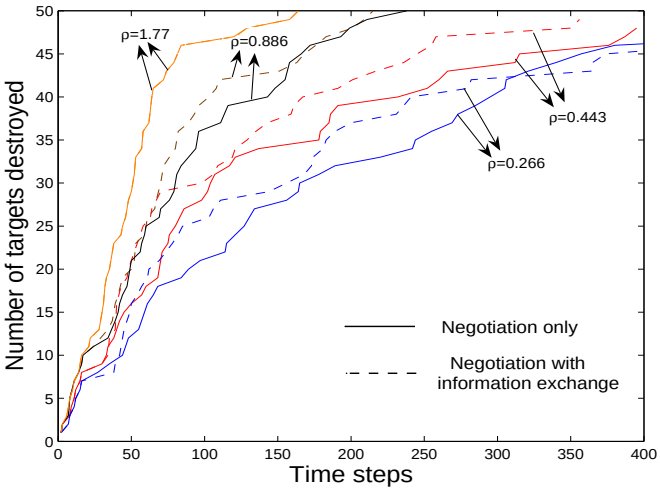


Figure 5: Number of targets destroyed for different proximity factors

The simulation is carried out using 7 UAVs for a search space consisting of 50 targets, with different proximity factors. Figure 5 shows the performance of negotiation and negotiation with target information based task allocation schemes. From the figure we can see that for lower proximity factors the number of targets destroyed are low as

compared to the number of targets destroyed in the higher proximity factor case. When the proximity factor is small, the effect of target information sharing during decision making by the agents that have targets in their sensor range is significant. For $\rho = 0.266$, we can see from the figure that the performance of negotiation with target information based task allocation is better than that using negotiation only. Here, the target information broadcast plays a crucial role in enhancing the performance. Similar kind of effect can be seen for $\rho = 0.443$. However, for $\sigma = 0.886$, the negotiation based task allocation is better than that of with target information exchange. This is due to the fact that the additional information about distant targets makes the agent choose targets to attack rather than perform search in its own neighborhood. This causes it to miss nearer targets outside its sensor range. For $\rho = 1.77$, the performance is same for both the negotiation schemes. Since the proximity factor is high, all the agents can sense all the targets hence there is no improvement in performance with information exchange. It should be noted that the amount of information broadcasted also plays a crucial role in the performance of the task allocation. Hence, there is always a tarde off between how much of information should be broadcast and the performance.

5. CONCLUSIONS

In this paper, we demonstrate an efficient task allocation scheme using negotiation for multiple UAVs performing search and destroy mission on an unknown region. The performance achieved using negotiation based task allocation is compared with greedy strategy for various sensor ranges and it is found that negotiation based task allocation is significantly better. We also studied the effect of task allocation mechanism on the spread of targets in the search space. The results show that with lower proximity factor the number of targets destroyed is low while with higher proximity factor the number of targets destroyed is high. We also studied the performance of negotiation scheme when agent communicate additional information to its neighbouring agents about the availability of targets. It is found that with lower sensor radius this information yield better results.

6. REFERENCES

[1] K.E. Nygard, P.R. Chandler, and M. Pachter: Dynamic network flow optimization models for air vehicle resource allocation, *Proc. of the the American Control Conference*, Arlington, Texas, June 2001, pp. 1853-1858.

[2] C. Schumacher, P. Chandler, M. Pachter, and L.S. Pachter: UAV task assignment with timing constraints, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, August, Austin, Texas, 2003, AIAA 2003-5664.

[3] C. Schumacher, P.R. Chandler, S. J. Rasmussen, and D. Walker: Task allocation for wide area search munitions with variable path length, *Proc. of the American Control Conference*, June, Denver, Colorado, 2003, pp. 3472-3477.

[4] C. Schumacher, P. Chandler, and S. J. Rasmussen: Task allocation for wide area search munitions via iterative netowrk flow, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Monterey, California, August 2002, AIAA 2002-4586.

[5] P. Chandler, M. Pachter, S. J. Rasmussen, and C. Schumacher: Distributed control for multiple UAVs with strongle coupled tasks, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Austin, Texas, August 2003, AIAA 2003-5799.

[6] P. Chandler, M. Pachter, S. J. Rasmussen, and C. Schumacher: Multiple task assignment for a UAV team, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Monterey, California, August 2002, AIAA 2002-4587.

[7] J.W. Mitchell, P. Chandler, M. Pachter, and S. J. Rasmussen: Communication delays in the cooperative control of wide area search munitions via iterative network flow, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Austin, Texas, August 2003, AIAA 2003-5665.

[8] J.W. Curtis and R. Murphey: Simultaneaous area search and task assignment for a team of cooperative agents, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Austin, Texas, August 2003, AIAA 2003-5584.

[9] D. Turra, L. Pollini, and M. Innocenti: Real-time unmanned vehicles task allocation with moving targets, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Providence, Rhode Island, August 2004, AIAA 2004-5253.

[10] Y. Jin, A. A. Minai, M. M. Polycarpou: Cooperative real-time search and task allocation in UAV teams, *IEEE Conference on Decision and Control*, Maui, Hawaii, December 2003, Vol. 1 , pp. 7 - 12.

[11] P.B. Sujit, A. Sinha, and D. Ghose: Multi-UAV task allocation using team theory, *Proc. of the IEEE Conference on Decision and Control*, Seville, Spain, December 2005.

[12] D.G. Rajnarayan and D. Ghose: Multiple agent team theoretic decision-making for searching unknown environments, *Proc. IEEE Conference on Decision and Control*, Maui, Hawaii, December 2003, pp. 2543-2548.

[13] M.B. Dias and A. Stentz: A free market architecture for distributed control of a multirobot system, *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, Venice, Italy, July 2000, pp. 115-122.

[14] M.B. Dias, R.M. Zlot, N. Kalra, and A. Stentz: Market-based multirobot coordination: A survey and analysis, *Technical report CMU-RI-TR-05-13*, Robotics Institute, Carnegie Mellon University, April 2005.

[15] M.B. Dias and A. Stentz: Enhanced negotiation and opportunistic optimization for market-based multirobot coordination, *Techical Report CMU-RI -TR-02-18*, Robotics Institute, Carnegie Mellon University, August, 2002.

[16] B. Gerkey, and M.J. Mataric: Sold!: Auction methods for multi-robot control, *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, October 2002, pp. 758-768.

[17] B. Gerkey, and M.J. Mataric: A Formal Framework for the Study of Task Allocation in Multi-Robot Systems, *International Journal of Robotics Research*, Vol. 23, No.9, Sep 2004, pp. 939-954.

[18] M.J. Mataric, G.S. Sukhatme, and E.H. Stergaard: Multi-robot task allocation in uncertain environments, *Autonomous Robots*, Vol. 14, 2003, pp. 255263.

[19] P. Gurfil: Evaluating UAV flock mission performance using Dudeks taxonomy, *Proc. of the American Control Conference*, Portland, Oregon, June 2005, pp. 4679-4684.

[20] M. Lagoudakis, P. Keskinocak, A. Kleywegt, and S. Koenig: Auctions with performance guarantees for multi-robot task allocation, *Proc. of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, Sendai, Japan, September 2004, pp. 1957-1962.

[21] A. Rubinstein: Perfect equilibrium in a bargaining model, *Econometrica*, Vol. 50, No. 1, 1982, pp. 97-109.

[22] S. Kraus: Automated negotiation and decision making in multiagent environments, *Multi-Agent Systems and Applications*, Springer LNAI 2086, (Eds.) M.Luck, V. Marik, O. Stepankova, and R. Trappl, 2001, pp. 150-172.

Distributed Automatic Target Recognition Using Multi-agent UAV Swarms

Prithviraj Dasgupta
Computer Science Department,
University of Nebraska, Omaha, NE 68182, USA
pdasgupta@mail.unomaha.edu

ABSTRACT

In modern day warfare, reconnaissance operations such as automatic target recognition(ATR) using unmanned aerial vehicles(UAVs) constitute a strategic war tactic. Traditionally, ATR is performed by UAVs that fly within the reconnaissance area to collect image data through sensors and upload the data to a central base station for analyzing and identifying potential targets. The centralized approach to ATR introduces several problems including scalability with the number of UAVs, network delays in communicating with the central location, and, susceptibility of the system to malicious attacks on the central location. These challenges can be addressed using a distributed system for performing ATR. In this paper, we describe a multi-agent system called COMSTAR-UAV (COoperative Multi-agent System for automatic TArget Recognition using Unmanned Aerial Vehicles) that uses swarming techniques inspired from insect colonies to perform ATR in a distributed manner.

Categories and Subject Descriptors: I.2.11 Distributed Artificial Intelligence: Multi-agent Systems
General Terms: Experimentation.
Keywords: Collective and Emergent Behavior, Simulation and Modeling, Distributed automatic target recognition.

1. INTRODUCTION

Automatic target recognition (ATR) involves determining the visual and other distinguishing features of an object, usually from a distance, and using this signature to automatically identify the object as a potential target. Traditionally this is achieved using unmanned aerial vehicles(UAVs) that obtain image data through their sensors from objects within a reconnaissance area and uploaded the data to a central location such as a base station, where potential targets are identified using image analysis algorithms [2, 4, 6]. However, this centralized model for ATR introduces several problems such as scalability and network delays in communication with the central location.

With the advancement of technology, the computation power of processors on UAVs has improved significantly and mini-UAVs can perform moderate computation beyond the trivial image data acquisition. However, the processors on UAVs are still not sophisticated enough to perform image identification solitarily on-board the UAV itself. We envisage that swarming[1, 3] provides a suitable mechanism to assimilate the capabilities of individual UAVs into a group of coordinating UAVs that perform ATR in a distributed manner. In a swarmed system, multiple mobile entities are directed to converge on a single point of interest, disperse and regroup again. To achieve distributed ATR using swarming, each UAV individually searches for potential targets within an area of interest using its image sensor. As soon as the image of an object is sensed to be a possible target by a UAV, other UAVs cooperate with it by swarming towards the potential target to collectively perform ATR and confirm the object as a target. In contrast to a centralized model for ATR, a distributed ATR model can: (1) scale up efficiently in the number of UAVs deployed and in the amount of data collected and processed for ATR, (2) improve the efficiency of the system because most computation is done locally within each UAV, which in turn reduces the time required to upload every image data from different UAVs onto the central location, and, (3) reduce the communication between the central location and individual UAVs so that the system is less susceptible to loss of data and failures due to wireless communication problems between an individual UAV and the central location. In this paper, we describe a multi-agent system called COMSTAR(Co-Operative Multi-agent Swarm for automatic TArget Recognition) that enables a group of UAVs deployed within a reconnaissance area to behave collectively as a swarm and perform ATR in a distributed manner.

Our swarming mechanism for ATR is based on the *stigmergetic* activity of social insects such as ants [1] to locate food. Stigmergy is a reinforcement learning mechanism that enables ants to indirectly communicate with each other about their environment using a chemical substance called pheromone. For example, while searching for food ants start from their nest and leave behind a pheromone trail along the path they traverse. The path leading from the nest to the food receives the highest amount of pheromone. Pheromone provides positive reinforcement to future ants, and, ants searching for the food later on use the trail as a positive reinforcement to reach the location of the food. To enable the ant algorithm in our system, we use pheromone to indicate potential targets while ants are implemented through software agents located within the computers on the UAVs. Software agents are characterized by a small footprint (few kilobytes) and are suitable for execution on computers within UAVs with limited computation power.

2. UAV OPERATION

We consider a battlefield scenario where UAVs are deployed from the base station into the area of interest(AOI). The AOI is logically divided into smaller sub-areas and a group of UAVs is deployed by the base station into each of these *home-areas*. We

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

assume that each UAV is provided with the following equipment to enable it to perform ATR: (1) Wireless communication capability for sending and receiving messages from other UAVs and the base station. (2) One or more image sensors capable of capturing snapshots of the area over which the UAV flies. (3) Processor with limited computing capability. (4) Rudimentary image processing software that enables the UAV to identify objects obtained by its image sensors as potential targets. (5) A global positioning system(GPS) that returns the 3-d coordinates (latitude, longitude, altitude) of the UAV's current location. Because of limited computational resources available on a single UAV, we assume that for definite target identification, $(n-1)$ other UAVs need to visit the location of a potential target reported by a single UAV to observe and confirm it as a definitive target.

After being deployed within its home-area, each UAV uses a deterministic foraging algorithm to navigate within its home area while searching for potential targets. When a UAV locates a potential target it associates a particular amount of pheromone with the target. Each UAV continuously exchanges information about targets it is aware of with other UAVs within its communication range using a gossip mechanism. The local map for potential targets that a UAV is aware of is maintained within it on a *pheromone landscape*. A UAV's task is to plan a navigation route by prioritizing the points on its pheromone landscape. The individual behavior of each UAV results in the movement of multiple UAVs towards the most prominent unconfirmed potential target in their vicinity, thereby manifesting the swarming behavior of the system. Because the computations on a UAV are performed by the agent located on it, we have used the terms agent and UAV interchangeably in the rest of the paper.

3. MODEL

We consider a set of $\mathcal{A}$ agents. Each agent $i \in \mathcal{A}$ maintains two sets: Ψ_i: Pheromone landscape of agent i comprising the set of potential targets agent i received through gossip or discovered through its sensors, and, Γ_i: set of tasks corresponding to the targets agent i is aware of. Each point $\psi \in \Psi_i$ corresponds to a potential target and is defined as $\psi = \{pher, loc, hops, time, vel, bearing\}$, where $pher \in [0,1]$ is the amount of pheromone associated with the potential target, $loc \in L \subset \Re^3$ is the location of the potential target in 3-d coordinates (latitude, longitude, altitude) obtained through the UAV's GPS, $hops$ is the number of agents that have forwarded the pheromone point ψ using the gossip mechanism before it reached agent i, and $time$ is the time at which this pheromone was last updated. The parameters vel and $bearing$ are used for mobile target tracking. In our system, time is measured in logical units called *ticks*. Every entity receives the current time measured in ticks from a globally available clock in the COMSTAR simulator. The availability of globally available clock avoids synchronization problems between different UAVs. Each task $\gamma \in \Gamma_i$ corresponds to a pheromone-point on the pheromone landscape Ψ_i of agent i and is defined as $\gamma = \{\psi, prio, status\}$ where $\psi \in \Psi_i$ is the pheromone corresponding to the task, $prio \in [0,1]$ is the priority of task γ, and, $status = \{Potential, Definite\}$ gives the identification status of the target. The priority of each task $\gamma \in \Gamma_i$ is recalculated every time the pheromone landscape of agent i gets updated. An agent's set of actions is given by: (i) *Navigate* to the location of the highest priority task within its task set, (ii) *Gossip* pheromone values that it is aware of, and, (iii) *Forage* deterministically to explore the AOI when the task set is empty.

Deterministic Foraging: When an agent is not swarming towards a pheromone point, it uses the deterministic foraging algorithm to navigate itself while avoiding flight-path conflicts with other agents. The deterministic foraging algorithm divides the immediate vicinity of an agent into sectors and then selects the least dense sector to move to. This ensures that each agent moves in a direction away from its neighboring areas that have higher concentration of agents.

Pheromone Landscape Update: The pheromone update equation used by agent i for pheromone point ψ is given by:

$$\begin{aligned} \psi.pher \leftarrow \lambda\psi.pher + (1-\lambda)(w_1 s + w_2 c) \\ \psi.time = currentTime; \\ \psi.loc = currentLocation, \end{aligned} \quad (1)$$

Parameters $s \in \Re$ and $c \in \Re$ in Equation 2 represent the severity of the potential target and the confidence in identifying the image from the sensor as a potential target. Weights w_1 and w_2 represent agent i's belief in the parameters s and c, while the parameter λ controls the preference between agent i's own perception and other agents' perception (pheromone value) of the potential target. The pheromone decay equation used by agent i is given by:

$$\begin{aligned} \epsilon = \text{currentTime} + \text{time to reach } \psi.loc - \psi.time, \\ \psi.pher \leftarrow \psi.pher.e^{-\beta\epsilon} \end{aligned} \quad (2)$$

where β is a control parameter, and, ϵ gives the time required by a UAV to reach the location $\psi.loc$ corresponding to pheromone point ψ that was last updated at $\psi.time$.

Gossip mechanism: Each agent employs a gossip mechanism[5] to exchange pheromone information with other agents. To set up a communication channel with other agents for sending the pheromone information, an agent broadcasts a *ping* message at regular intervals. Other agents that are within communication range of the agent originating the ping message respond with a *ping-ack* message that sets up the wireless communication channel. To disseminate gossip information across the system, each agent forwards gossip information it receives using a controlled flooding mechanism.

Task Prioritization: We consider two task prioritization strategies: ***(1) Myopic Strategy:*** For a myopic agent, the priority of its tasks is determined from the pheromone value of a point on its pheromone landscape perceived by itself only. The priority function of a myopic agent gives a higher preference to pheromone points that are significantly prominent than the agent's current destination, or, are en-route to the agent's current destination. ***(2) Look-Ahead Strategy:*** In the lookahead strategy, before prioritizing a new task, an agent takes into account the intention of other agents to perform that task. If the agent discovers that $(n-1)$ other agents (n agents are required to confirm a target) are already performing or planning to perform the task, it continues with its current mission and does not consider the new task for prioritization.

Mobile Target-Tracking With Anti-Pheromone: In our mobile target identification algorithm, the most recently observed location of a mobile target is marked with pheromone as before while the previous locations of the target are marked with anti-pheromone. Unlike pheromone, anti-pheromone provides negative reinforcement and prevents an agent from visting the point marked with anti-pheromone on its pheromone landscape. Our anti-pheromone based target tracking mechanism directs the UAV swarm towards the estimated location of a mobile target based on the target's most recently observed location and dynamics, and, is capable of tracking targets that are both continously and intermittently mobile.

4. SIMULATION RESULTS

For all our experiments we have considered the time (measured in no. of ticks) required to confirm all targets within an AOI as

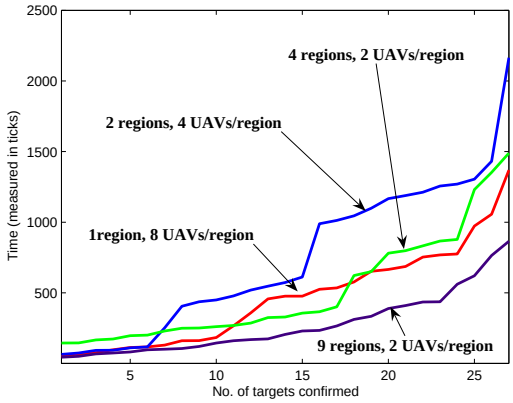


Figure 1: Time required to confirm targets in an AOI for different number of home-areas(regions) into which the AOI is sub-divided.

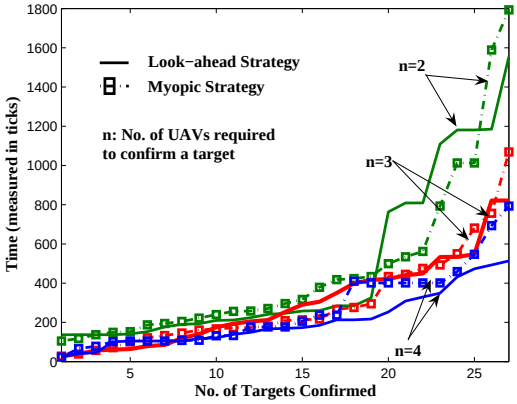


Figure 2: Time required to confirm targets in an AOI with myopic and lookahead strategies for different initial UAV densities per region.

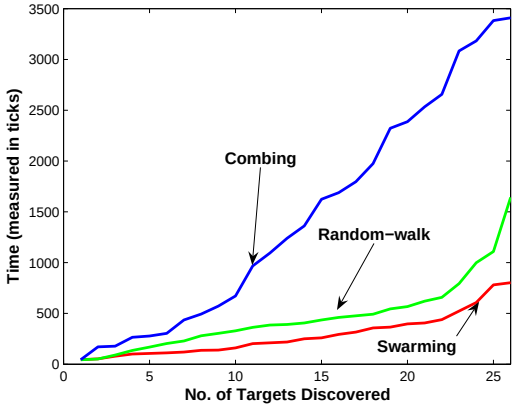


Figure 3: Time required to cover a square AOI using three different coverage strategies.

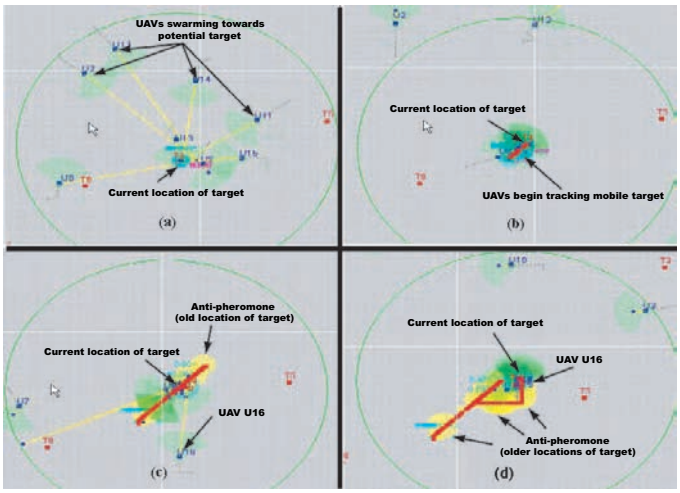


Figure 4: Snapshots illustrating the anti-pheromone update by different UAVs while tracking a mobile target.

the metric for quantifying system performance. The AOI is considered to be an invariant square region. However, the number of regions or home-areas into which the AOI is subdivided is varied to measure the effect of the size and number of home-areas on system performance. Targets are assumed to distributed randomly within the AOI. The communication range of each UAV is assumed to be $1/3$ of one side of the square AOI. The pheromone decay rate is set at $1/200$ per tick. Our experimental results are reported in Figures 1 through 4.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have reported the results from software based simulations for a swarmed model of ATR and shown that is compares favorably with existing ATR strategies. In the future, we are investigating mechanisms to improve the lookahead strategy using a temporal inference technique, improve the communication protocol using a a decision theoretic mechanism, and, address issues related to security and fault tolerance in the system.

6. REFERENCES

[1] E. Bonabeau, M. Dorigo and G. Theraulaz, ”Swarm Intelligence: From Natural to Artificial Systems,” Oxford University Press, 1999.

[2] M. Cohen, ”An Introduction to Automatic Target Recognition,” EW Design Engineers’ Handbook, 1989-1990, pp. 2-1 - 2-2.

[3] S. Edwards, ”Swarming on the Battlefield: Past, present and future,” RAND National Security Research Div. Report, 2000.

[4] F. Gaudiano, Shargel, and E. Bonabeau, ”Control of UAV swarms: What the bugs can teach us,” Proc. of 2nd AIAA Unmanned Unlimited, 2003.

[5] M. Portmann and A. Seneviratne, “Cost-effective broadcast for fully decentralized peer-to-peer networks,” Computer Communications, vol. 26, no. 11, 2003, pp. 1159-1167.

[6] J. Sauter, R. Matthews, H. Parunak, and S. Brueckner, “Performance of Digital Pheromones for Swarming Vehicle Control,” Proc. of AAMAS 2005, pp. 903-910.

Action Awareness – Enabling Agents to Optimize, Transform, and Coordinate Plans

Freek Stulp
stulp@in.tum.de

Michael Beetz
beetz@in.tum.de

Intelligent Autonomous Systems Group, Department of Computer Science IX
Technische Universität München, Boltzmannstr. 3, D-85748, Garching bei Munich, Germany

ABSTRACT

As agent systems are solving more and more complex tasks in increasingly challenging domains, the systems themselves are becoming more complex too, often compromising their adaptivity and robustness. A promising approach to solve this problem is to provide agents with reflective capabilities. Agents that can reflect on the effects and expected performance of their actions, are more aware and knowledgeable of their capabilities and shortcomings.

In this paper, we introduce a computational model for what we call *action awareness*. To achieve this awareness, agents learn predictive action models from observed experience. This knowledge is then used to optimize, transform and coordinate plans. We apply this computational model to a number of typical scenarios from robotic soccer. Various experiments on real robots demonstrate that action awareness enables the robots to improve the performance of their plans substantially.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics—*autonomous vehicles*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*plan execution, formation, and generation*

General Terms

Computational Model

Keywords

learning action models, plan optimization

1. INTRODUCTION

Improvements in agent technology allows agent systems to solve more and more complex tasks. As the tasks and environments become more complex, so do the agent systems themselves. This often compromises the robustness and adaptivity of the system, leading to a shorter operational life-time. A promising approach to solve this problem is to provide the agents with reflective capabilities [2]. When the agents are aware of what they are doing, and why, they become more robust, and can adapt to new situations. We call this *action awareness*.

We investigate the issues of action awareness and its applications in the context of multi-robot control. Robotic soccer is a good example of a domain in which the agent's interactions with the environment are often very complex. Therefore, hand-coding the controller is intractable and failure-prone. However, by reasoning about their actions, robots are able to choose the appropriate action, and optimize its parameterization on-line themselves.

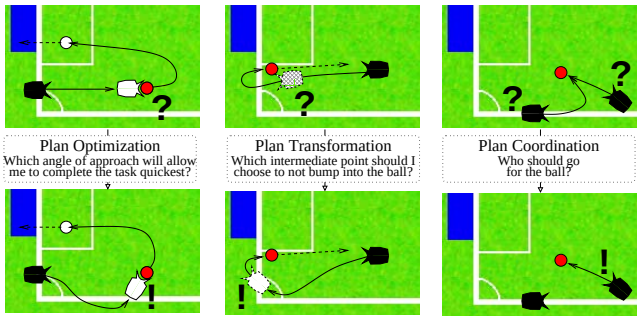


Figure 1: Three typical robotic soccer scenarios including control tasks requiring action awareness.

In this paper, we will propose a novel **computational model** for the acquisition and application of action awareness. It shows how action awareness can be acquired by **learning predictive action models** from observed experience (Section 2). We then explain how action awareness can be used to **optimize** (Section 3.1), **transform** (Section 3.2), and **coordinate** (Section 3.3) underspecified plans with highly parameterizable actions in the context of robotic soccer. While the individual contributions are not new, we do not know of a system implementing this as a comprehensive computational model, and applying it to autonomous robot control.

The computational model is depicted in Figure 2. To the left are the more or less 'traditional' action selection modules, and to the right are the novel modules that generate and exploit action awareness. When the robot has idle time, it learns prediction models for the actions in the ac-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

tion library. During operation time, action chains are generated. Three modules then take these chains, and optimize, transform, and coordinate them using the pre- and post-conditions, as well as the prediction models. The modified chains then simply replace the original ones, and are executed.

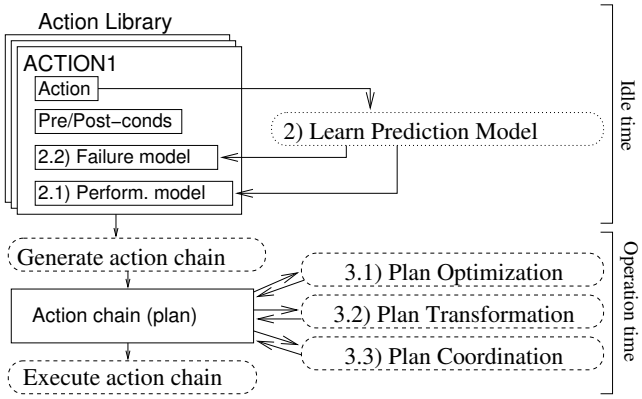


Figure 2: Computational Model

2. BECOMING ACTION AWARE: LEARNING PREDICTION MODELS

The robots achieve action awareness by learning action prediction models from observed experience. Given a specific scenario, these models should predict whether an action is likely to succeed, and what its expected performance will be. In this section, we will explain how temporal and failure prediction models were learned. The experiments in this paper were conducted with three customized Pioneer I robots of our RoboCup mid-size team the Agilo RoboCuppers. The robots use differential drive for locomotion, and a single forward facing CCD camera for state estimation.

2.1 Action Performance Prediction

Being aware of an action's expected performance is valuable knowledge. In many robotic domains, execution time is an important performance measure. Therefore we learn temporal prediction models. For our actions `goToPose` and `dribbleBall`, the model should predict the expected time of navigating from a start to a goal pose with this action. This model is acquired by training a model tree with data acquired on real robots. The training examples for the model tree were acquired by randomly choosing goal destination on the field, executing the `goToPose` or `dribbleBall` action, and measuring the time it took for the approach. In [6] we determined that 40 minutes operation time on a real robot yields enough examples to acquire an accurate temporal prediction model. The mean absolute error between the actual an predicted approach time for the `goToPose` and `dribbleBall` actions are 0.18s and 0.22s respectively.

2.2 Action Failure Prediction

The `goToPose` action can often be used well to approach the ball. However, in some situations it will bump into the ball before achieving the desired orientation, as can be seen in Figure 1. The second performance model we have learned predicts whether executing `goToPose` will lead to a collision with the ball or not. This model was learned by training a decision tree on data acquired in simulation. The training examples for this tree were acquired by having the robot executed `goToPose` a thousand times, with random initial and destination poses, the ball always positioned at the destination pose. For each run, we record if the robot had a `Collision` with the ball before reaching its desired position and orientation. If not, the run was a `Success`. On a test set, the learned decision tree predicts collisions correctly in almost 90%.

3. APPLYING PREDICTION MODELS TO AGENT CONTROL

In this section we will show how prediction models can be used to optimize, transform and coordinate plans.

3.1 Plan Optimization

Actions and action chains are often underspecified in plans. Take the example in Figure 3. The robot's plan is to approach the ball, and dribble it to the specified point. If it approached the ball as fast as possible, it would end up in the position on the top field in the image. This is an unfortunate position from which to start dribbling towards the goal. The problem is that, according to the planner, being at the ball is considered sufficient for dribbling the ball, and the angle of approach is considered to be irrelevant for the consecutive dribbling action. So, the intermediate state is under-specified, and we are allowed to choose any state from the set of intermediate states that the conditions deem valid.

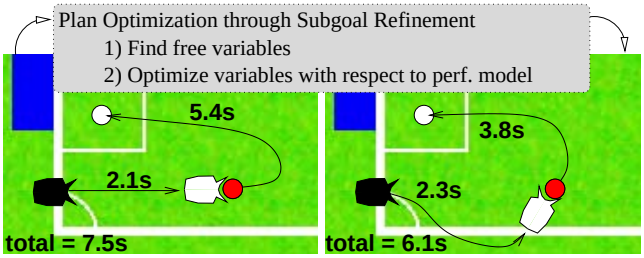


Figure 3: Overview of plan optimization.

What we would like the robot to do is to go to the ball *in order* to dribble it towards the goal afterwards. The robot should, as depicted in the right field in Figure 3, perform the first action sub-optimally in order to achieve a much better position for executing the second plan step. Our action-aware plan execution system can determine optimal intermediate states on the fly, using predictive models. We call this *subgoal refinement*.

First of all, the agent has to determine which variables can be used for optimization. That is, which variables are not bound by the plan. In this example, the only free variable is the angle of approach. For an explanation of how free variables are determined automatically, we refer to [4]. To optimize the action chain, we will have to find those values for the free variables for which the performance of the action chain is the highest. This overall performance is estimated by summing over the temporal prediction models of all actions that constitute the action chain. In Figure 3,

the predicted execution time of the first action in the left situation is 2.1s. However, the total time is 7.5s, because the second action takes 5.4s for this angle. Although the first action is executed very fast, this is not the optimum *overall* performance. By performing the first action suboptimally, an minimum overall predicted performance of 6.1s can be achieved, for an angle of 59°. This particular example yields a performance increase of 38%. The mean increase over a 1000 examples with random robot, ball and final goal positions was 12%.

Without subgoal refinement, the transitions between actions were very abrupt. In general, these motion patterns are so characteristic for robots that people trying to imitate robotic behavior will do so by making abrupt movements between actions. In contrast, one of the impressive capabilities of animals and humans is their capability to perform chains of actions in optimal ways and with seamless transitions between subsequent actions. It is interesting to see that requiring optimal performance can implicitly yield smooth transitions in robotic and natural domains, even though smoothness in itself is not an explicit goal in either domain.

3.2 Plan Transformation

In planning systems, the pre-conditions determine a set of states in which the action can be executed. Frequently, actions can be reused to perform other tasks than those they were originally designed for. For instance, the `goToPose` action can also be used to approach the ball. However, since `goToPose` does not consider the ball, it will sometimes bump into it before achieving the goal pose. In Section 2.2 we saw that a decision tree could be learned to accurately predict when this happens. In a sense, this decision tree can be seen as a learned pre-condition for a new `approachBall` action. This pre-condition can be used to determine when a plan to approach the ball will fail, and transform it so that a collision is avoided. Exactly how this is done is discussed in more detail in [5].

3.3 Plan Coordination

Many complex application tasks require two or more agents to cooperate in order to solve the task. A key aspect of these systems is that multiple agents share the same workspace, and can therefore not abstract away from the actions of other robots. Therefore, agents must not only be aware of their own actions, but also of those of others. Humans are very good at performing joint actions in shared workspaces, because they are very aware of the actions of others. Humans achieve this by inferring the intentions of others. Once the beliefs and desires of the cooperating party are known, we imagine what we would do in that situation. In contrast, coordination in multi-agent systems is usually achieved by extensive communication of intentions or utilities [3].

We believe that there are many domains in which implicit coordination is essential. Rescue robotics and autonomous vehicles operating in traffic are examples of domains in which robust communication is not guaranteed, but where correct coordination and action anticipation is a matter of life and death. We also consider implicit coordination to be essential for natural interaction between robots and humans. It cannot be expected of humans to continuously communicate their intentions. Instead, the robot must be able to anticipate a human's intentions, based on predictive models of human behavior.

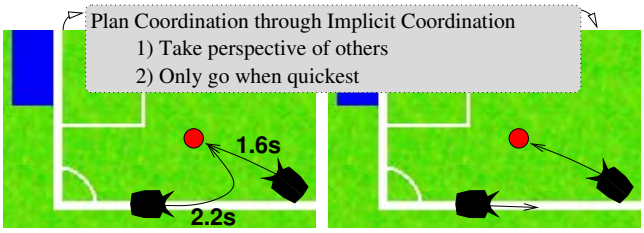


Figure 4: Overview of plan coordination.

A typical coordination task from the robotic soccer domain is to regain ball possession. Acquiring ball possession is a goal for the team as a whole, but it can be achieved by only one of the field players. Of course, the robots must agree upon which robot will approach the ball. The intuitive underlying rule is that only the robot who is quickest to the ball should approach it. The key to achieving implicit coordination is *perspective taking*. If the robots have estimates of each other's states, they can predict ball approach time for the others just as well as for themselves. In a sense, the awareness of their own action also provides them with awareness of actions of others. Once each robot has computed each robot's predicted ball approach time, each robot can determine if it is the fastest or not.

Our experiments have shown that the robots reach almost perfect agreement on this (99%), without communication [6]. The robot on which agreement is reached is indeed the quickest in 96% of the experiments.

Acknowledgments

The work described in this paper is partially funded by the German Research Foundation (DFG) under the contract RA 359/7-2, as part of the Priority Research Program SPP1125 *Cooperative Teams of Mobile Robots in Dynamic Environments*.

4. REFERENCES

[1] T. Belker, M. Hammel, and J. Hertzberg. Learning to optimize mobile robot navigation based on HTN plans. In *Proceedings of the International Conference on Robotics and Automation (ICRA03)*, 2003.

[2] R. Brachman. Systems that know what they're doing. *IEEE Intelligent Systems*, 2002.

[3] B. P. Gerkey and M. J. Matarić. Multi-robot task allocation: Analyzing the complexity and optimality of key architectures. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2003.

[4] F. Stulp and M. Beetz. Optimized execution of action chains using learned performance models of abstract actions. In *Proc. of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

[5] F. Stulp and M. Beetz. Tailoring action parameterizations to their task contexts, 2005. IJCAI Workshop "Agents in Real-Time and Dynamic Environments".

[6] F. Stulp, M. Isik, and M. Beetz. Implicit coordination in robotic teams using learned prediction models. Accepted for the IEEE International Conference on Robotics and Automation (ICRA), 2006.

Programming Modular Robots with the TOTA Middleware

Marco Mamei
Dipartimento di Scienze e Metodi dell'Ingegneria
University of Modena and Reggio Emilia
Reggio Emilia, ITALY
marco.mamei@unimore.it

Franco Zambonelli
Dipartimento di Scienze e Metodi dell'Ingegneria
University of Modena and Reggio Emilia
Reggio Emilia, ITALY
franco.zambonelli@unimore.it

ABSTRACT

Modular robots represent a perfect application scenario for multiagent coordination. The autonomous modules composing the robot must coordinate their respective activities to enforce a specific global shape or a coherent motion gait. Here we show how the TOTA ("Tuples On The Air") middleware can be effectively exploited to support agents' coordination in this context. The key idea in TOTA is to rely on spatially distributed tuples, spread across the robot, to guide the agents' activities in moving and reshaping the robot. Three simulated examples are presented to support our claims.

Categories and Subject Descriptors

D.3.3 [**Programming Languages**]: Language Constructs and Features; C.2.4 [**Distributed Systems**]: Distributed Applications

General Terms

Modular Robots, TOTA Middleware

Keywords

Coordination, Modular Robots, TOTA Middleware

1. INTRODUCTION

A modular (or self-reconfigurable) robot is a flexible robot made up of a collection of (typically simple) autonomous elements connected with each other with some degree of freedom in their relative movements [2, 3].

The key idea underlying a modular robot is to have its components execute distributed control algorithms so as to coordinate their actions and let the robot assume a specific shape or move according to some specific motion pattern (i.e., gait).

Modular robots represent a perfect application scenario for multiagent systems. On the one hand, the modules constituting the modular robot should be autonomous and running agent applications. This, in fact, avoids single point of failure and bottlenecks. If one module breaks down the other can reorganize their activities leaving the broken element behind. Autonomy enables also disconnected operations: modules can disassemble, move along different paths and reassemble subsequently. On the other hand, the main task the modules have to undertake is *coordination*. The agents governing the modules must coordinate their respective activities to enforce a specific global shape or a coherent motion gait in the whole robot (i.e., multiagent system).

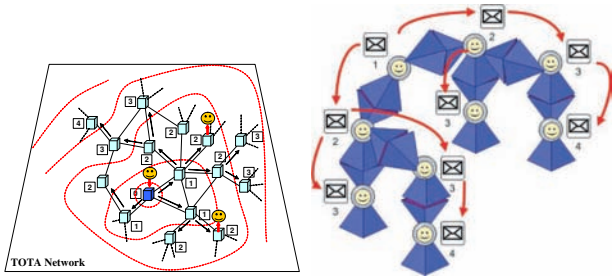


Figure 1: (left) The general scenario of TOTA. (right) TOTA in modular robots.

For the purpose of supporting agent coordination, in our research, we developed a general middleware called "Tuples On The Air" (TOTA). TOTA supports the creation of distributed overlay data structures spread across a distributed network. In this paper, we want to test the TOTA's generality by implementing some advanced mechanisms required for the control of a modular robot. In particular, our goal is to show that TOTA enables to easily and flexibly express and program biologically-inspired control algorithms (i.e. hormone-based [3]) for modular robots' coordination.

2. THE TUPLES ON THE AIR APPROACH

TOTA is a general-purpose middleware for multiagent coordination, in distributed computing scenarios. In TOTA we assume the presence of a network of possibly mobile nodes, each running an agent application. Each agent is supported by a local version of the TOTA middleware. Nodes are connected by only short-range network links. Each agent has only a local (one-hop) perception of its environment. Long range interactions must be mediated by other agents. Upon the distributed space identified by the dynamic network of TOTA nodes, each agent is capable of locally storing tuples

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for pro t or commercial advantage and that copies bear this notice and the full citation on the rst page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior speci c permission and/or a fee.
AAMAS'06 May 8 12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1›59593›303›4/06/0005 ...$5.00.

and letting them diffuse through the network. Tuples are injected in the system from a particular node, and spread hop-by-hop accordingly to a specified propagation rule (see Fig. 1(left)). In the modular robot scenario, this reflects in having each agent installed in a module of the robot. The modules (and thus the agents) are connected according to the topology of the robot. Tuples are injected and propagate across the robot's body (see Fig. 1(right)).

In TOTA, distributed tuples **T=(C,P,M)** are characterized by a content **C**, a propagation rule **P** and a maintenance rule **M**. The content **C** is an ordered set of typed fields representing the information carried on by the tuple. The propagation rule **P** determines how the tuple should be distributed and propagated in the network, and how the tuple content should change during propagation. The maintenance rule **M** determines how a tuple's distributed structure should react to events occurring in the environment.

We developed a first prototype of TOTA running on Linux IPAQs equipped with 802.11b WLAN and Java (J2ME, CDC, Personal profile). Moreover, we have implemented an emulator to analyze TOTA behavior in presence of hundreds of nodes [1]. To perform experiments with modular robots, we connected our TOTA simulator with the Polybot modular robot simulator [2]. This program can simulate the behavior of various types of modular robots, taking into account both the characteristics of the joints connecting different parts of the robot and the physical forces acting on the robot (e.g., gravitation), and offers a 3D view of the robot's actual configuration and movements.

3. MULTIAGENT COORDINATION IN MODULAR ROBOTS

From a multiagent perspective, the main challenge in modular robots is the design of decentralized coordination mechanisms enabling autonomous agents (i.e., modules) to coordinate their actions to achieve a specific global shape, or a specific motion gait. Some of the most innovative approaches to control a modular robot adopt the biologically inspired idea of hormones [3]. Hormone signals are actually sort of messages, spread across the robot and triggering the individual actuator's bending. A motion gait would be encoded by means of a specific sequence of hormones to be injected in the robot and by means of specific reactions triggered by these hormones, changing the bending angles.

The idea of hormones is a perfect match for TOTA distributed tuples and our approach has been to re-implement hormones with the support of our middleware. With this regard, it is fair to remark that we do not propose a novel approach for modular robot coordination. We just take advantage of the TOTA middleware to implement (with some changes) the hormone-based approach [3]. In our opinion, the importance of how to actually program such systems should not be overlooked and the proposed approach is valuable with regard to this topic.

In particular, the main subject of our research has been chain-type modular robots. This kind of robots is characterized by the fact modules are connected in a line configuration (e.g., snake-like), or - eventually - in a tree-like configuration (e.g., robot with legs), see the figures in next pages. In our experiments, we assumed that the robot is composed of very simple equal modules (i.e., joint actuators). Each module has a "front" side and a "back" side. On each side there is an infrared (IR) network link. Each module runs the TOTA middleware and an agent in charge of driving the module joint. The agent, looking at the active IR links, is able to infer whether it is the "head," the "tail," or a part of the "body" of the robot. Specifically, the "head" agent is the one with only the back IR link active, the "tail" agent is the one with only the front IR link active, a "body" agent is one having both the IR links active. The process of assessing whether an IR link is active or not can be based on "ping" messages and can be executed iteratively to take into account topological reconfigurations and module breakdown.

From a methodology point of view and in very general terms, the proposed approach consists in codifying a motion gait by means of a *Gait* tuple (examples are in the next section). Once this tuple is injected in the modular robot, it propagates hop by hop across all the modules changing its content to the desired gait. When an agent installed on a module senses the income of a tuple of this kind, it reacts by bending the module joint by the angle specified in the tuple. So, for example, if in a robot composed of N modules a tuple having in its content a fixed angle of about $(360/N)^\circ$ is spread, the robot closes into a loop.

4. EXPERIMENTS

In the rest of this section we will use TOTA to support three motion gaits in modular robot: the "caterpillar gait" (that lets the robot proceed by mimicking the motion of a snake) and the "rolling gait" (that lets the robot close in a loop, and then roll). Finally, we will present a gait for legged robots where agents coordinate legs' movements to proceed forward. It is worth noting that while the mechanisms underlying the former two gaits are well known in modular robot research [3], the latter one has been designed from scratch.

4.1 Caterpillar Gait

To implement the caterpillar gait, the head agent injects a caterpillar tuple (i.e., a TOTA tuple of the class *Caterpillar Gait Tuple*). The general structure of such a tuple is depicted below it propagates across the robot letting it bend accordingly. Once the tail agent receives the tuple, according to the general description given above, it injects another tuple to notify the head that a new step is ready to be executed. At this point, the head agent repropagates the tuple. This process is iterated letting the whole robot move performing the caterpillar gait (see Fig. 2).

Caterpillar Gait Tuple
C = (state, angle)
P = (propagate hop-by-hop, storing on intermediate nodes changing the content accordingly to the following table)

Current State	New State	New Angle
INIT	A	$+45^\circ$
A	B	$+45^\circ$
B	C	-45°
C	D	-45°
D	A	$+45^\circ$

M = (If on the head node and upon the receipt of a gait-tuple, re-apply propagation)

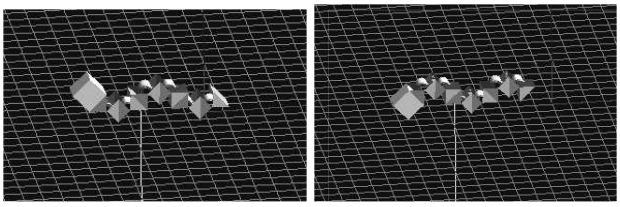


Figure 2: Some stages of a caterpillar gait, in a chain-typed modular robot, composed of six actuators

4.2 Rolling Gait

The idea of this gait is to let the robot close in a loop and then roll. Unfortunately, the modular robot simulator we employed does not allow structures with loops. Structures with loops are overconstrained. The simulation does not solve the constraint satisfaction problem. To overcome this problem, we let the robot bend in an open loop (something like a 'C' shape) and then roll. Although this complicates the rolling procedure, it allows us to maintain the general approach described before. In fact, we still have a "head" and a "tail" agent that would be otherwise removed if the loop were actually closed (i.e., with only "body" agents). The *Rolling Gait Tuple* is the tuple employed to let the robot roll. In general terms, it can have two states, T (turn) and F (flat). Consider, for example, a robot composed of 12 modules and assuming a turning angle of 45°. A tuple spread in the robot with a distributed shape like "FTTFTTFT-TFTT" closes in a loop. Then, if the tuple changes its content by "rolling" the above string (like the ROL assembler command), the robot performs the rolling gait. From the single tuple point of view, this consists in changing its content to assume values F - T - T iteratively. It is worth noting that such a kind of content change critically depends on the number of modules composing the robot and the number of turns we want to implement to let the robot close in a loop. For example, three turns of 60° each create a triangular track, four turns of 90° each create a rectangular track, etc. Moreover, it depends on the number of modules involved in each turn. Despite all these parameters, it is rather easy to build a general algorithm enabling a tuple to create dynamically, at runtime, the sequence of F and T states it has to cycle and the corresponding turning angle to enable the rolling gait. Some snapshots showing the rolling gait in action are in Fig. 3.

Rolling Gait Tuple
C = (state, angle)
P = (propagate hop-by-hop, cycling between the states F - T - T. Set the angle to 45deg if the state is T. Set the angle to 0 if the state is F)
M = (if the network topology changes, restart propagation according to new head, body and tail position)

4.3 Walking Gait

We performed this experiment on a "legged" robot that can move by coordinating legs movements. The idea of this experiment is to have the modules of the robot connected in a 6-legs configuration, and then coordinate the actions of these modules so as to let the legged robot walk. The robot

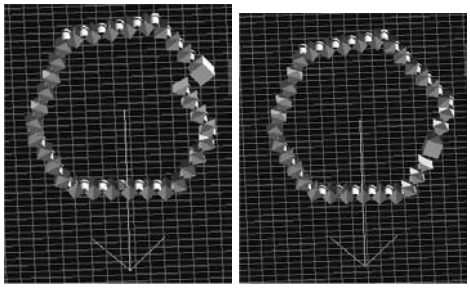


Figure 3: Some stages of a rolling gait in a chain-typed modular robot composed of 32 actuators.

can start moving the legs to proceed upward. This again is realized by letting the head of the robot inject another tuple that propagates across the modules. This *Walker Gait Tuple* tuple is very simple: it alternatively lets the left and right robot legs swing 45° forward. Some screen-shots of the actual robot movement are presented in Fig. 4.

Walker Gait Tuple
C = (state = LEFT—RIGHT, angle)
P = (propagate hop-by-hop. if current node is a spine node switch state. if state == LEFT turn left legs to 45deg and right legs to -45deg. if state == RIGHT turn right legs to 45deg and left legs to -45deg.
M = (if the network topology changes, restart propagation according to new head, body and tail position)

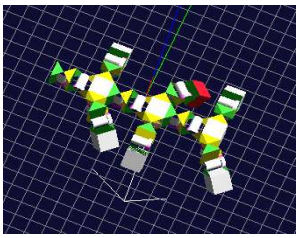


Figure 4: The legged robot walks by coordinating legs movements.

5. ACKNOWLEDGMENTS

Work supported by the project CASCADAS (IST-027807) funded by the FET Program of the European Commission.

6. REFERENCES

[1] M. Mamei and F. Zambonelli. Programming pervasive and mobile computing applications with the tota middleware. In *Proceedings of the International Conference On Pervasive Computing (Percom)*. IEEE CS Press, Orlando, Florida, USA, 2004.

[2] Modular reconfigurable robotics at parc. http://www2.parc.com/spl/projects/modrobots.

[3] W. Shen, B. Salemi, and P. Will. Hormone-inspired adaptive communication and distributed control for conro self-reconfigurable robots. *IEEE Transactions on Robotics and Automation*, 18(5):1 – 12, 2002.

RADIC - A Generic Component for the Integration of Existing Reactive and Deliberative Layers

Matthias Scheutz
Artificial Intelligence & Robotics Lab
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
mscheutz@cse.nd.edu

James Kramer
Artificial Intelligence & Robotics Lab
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
jkramer3@cse.nd.edu

ABSTRACT

Hybrid architectures have been developed to preserve the responsiveness of reactive layers while also providing the benefits of higher level deliberative capabilities. The challenge of hybrid architecture design is to integrate layers of very different functional roles. We propose a component, called RADIC, that uses a generic technique for the construction of hybrid architectures from pre-existing reactive and deliberative layers that requires only minimal modifications to each layer. In particular, we give a high-level description of the operation and benefits of the component and its algorithms.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Design

Keywords

hybrid architectures, architecture integration, robotics

1. INTRODUCTION

Hybrid architectures for autonomous robots combine reactive and deliberative layers to preserve both responsiveness and the efficiency of symbolic representations. Yet, as pointed out in [11], "hybridism usually translates to *ad hoc* or unprincipled designs with all its attendant problems" and "hybrid architectures tend to be very application-specific."

To address these issues, we present a Reactive And Deliberative Layer Integration Component (RADIC) that provides a generalized "bridge" between layers. By maintaining layer separation, RADIC allows their principled design in isolation; by effecting a mapping of data between layers of disparate functionality, RADIC provides a general connection method to reduce application related specificity.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. HYBRID ARCHITECTURES

For autonomous robotic agents, which require fast, highly responsive reactive layers (e.g., [7, 10]), the design of hybrid architectures is difficult, largely because the nature of the layers is so different. The challenge of hybrid architectures is to integrate the layers' disparate aspects of operation, at least achieving the following functional mappings:

F1: between "global" and "egocentric" relations
F2: from discrete actions to continuous motion
F3: between logical update time and real-time operation
F4: from stateful to stateless operation

We give a brief overview of a varied selection of hybrid architectures, describing how each satisfies items **F1-F4**:

***SSS* [4]:** Short for "*S*ervo, *S*ubsumption, *S*ymbolic" layers, the symbolic layer uses a coarse-grained world model that configures the reactive layer via parameterization of selected behaviors in the subsumption layer. SSS explicitly addresses the functional mappings by discretizing space between the servo and subsumption layers (**F1, F2**) and time between the subsumption and symbolic layers (**F3, F4**).

***AuRA* [2]:** AuRA's deliberative layer is composed of a planner, spatial reasoner, and sequencer, while the reactive layer consists of libraries of behavior schemas, activated by the sequencer and adaptable via a homeostatic component, producing actuation commands via vector summation of schema output. Functional mappings **F1**, **F3** and **F4** are made through the interaction of the schema controller with the plan sequencer, spatial reasoner, and planner, while **F2** is accomplished by the selection of behavior schemas.

***3T* [3]:** The 3T (short for "three tier") architecture, which extends work on both the RAPs [5] and ATLANTIS [6] architectures, uses a deliberative planner to synthesize goals in a partially ordered task list, executed by reactive skills (RAPs), bridged by a skill manager/sequencer (RAP interpreter). Tiers tend to operate at different time scales to fulfill mapping **F3**. Mappings **F1** and **F4** use all tiers; the planner operates at a high level of abstraction, the sequencer selects a set

of skills from the RAP library, while skills operate in a context dependent manner. Mapping **F2** is performed by the integration of skills and the sequencer; RAPs are symbolic, discrete steps for accomplishing a task composed of skills that implement continuous motion.

***Saphira* [8]:** Saphira uses a strong internal world model called the *local perceptual space* (LPS) that coordinates sensory data with representations. Planning is performed by the *PRS-Lite* component, which (de)activates sets of behaviors that are fused using fuzzy logic. Functional mappings **F1** and **F3** are accomplished via the LPS and PRS-Lite, while **F2** is accounted for by behavior selection and fuzzy logic fusion. Abstract representations are integrated in the LPS to fulfill mapping **F4**.

***4-D/RCS* [1]:** 4-D/RCS features a hierarchy of six levels (actuator, servo, primitive, subsystem, vehicle, and section) that operate on different scales. Each contains a supervisor and a set of subordinate agents that supervise units at the next lower level. Each node contains behavior generation, world modeling, sensory processing, value judgment processes and a knowledge database. Mappings **F1-F4** are explicitly made between each level.

While all of the above hybrid architectures successfully combines reactive and deliberative layers, each subscribes to a specific design philosophy, if not in integrating layers, then by specifying the design of the layers themselves. This restriction not only limits potential reuses of the developed layers and software, but more importantly the extent to which the designs themselves generalize. It would be desirable to achieve a more general connection or integration method that could be employed for many different reactive and deliberative layers without the need for much adaptation of either layer. Ideally, we would like a single, generic, "mediating component" to connect existing reactive and deliberative layers that successfully achieves the requisite functional mappings **F1-F4**. Several design desiderata suggest themselves for such a component; in particular, the method should:

I1: require minimal changes to existing layers

I2: provide integration of individual layers, while executing independently, autonomously, and in parallel

I3: use minimal processing time

I4: add functionality that improves the performance of systems that contain the layers in isolation

All the hybrid architectures above fail to satisfy some of **I1-I4**, in addition to the various criticisms (including those presented in [6, 3, 9]). Neither Saphira nor 4-D/RCS satisfies **I1**, albeit for different reasons. While the large amount of integration performed by the LPS in Saphira has the benefit of *coherency*, it also means that functional changes necessitate large modifications. On the other hand, the strict specification of interfaces between levels and substantial data flow within a single level in 4-D/RCS has the effect that slight modifications to functionality may entail large changes across components or levels. None of SSS, AuRA, and 3T meet **I2** because all switch completely back and forth between reactive and deliberative layers; the deliberative layer produces a fully formulated plan that is passed to the reactive layer, which then remains in control until a step either completes or fails. Both Saphira and 4-D/RCS suffer from difficulties with **I3**; the use of the LPS in Saphira requires a substantial amount of processing, while the full complement of components in 4-D/RCS introduces possibly redundant or unnecessary processing. Finally, **I4** presents difficulties for 3T, AuRA, and 4-D/RCS. In both 3T and AuRA, operation of the reactive layer is tightly knit and depends completely on the sequencer, making it difficult to add external functionality. A similar situation occurs with 4-D/RCS, where the functionality of a given level is specified completely by its internal operation. The proposed RADIC component, which we describe next, has been designed to satisfy all the functional mappings **F1-F4** and design desiderata **I1-I4**.

3. THE RADIC APPROACH

Functional mappings **F1-F4** are made in all hybrid architectures. The approach taken with RADIC can be described as effecting a *generic technique* for integrating (1) a *sequence of goals* originating from a deliberative layer, (2) a stream of *sensor data*, and (3) a *sequence of actions* (e.g., motor commands) from the reactive layer (see Figure 1), as performed by the `updateRADIC` algorithm given in Figure 2. The universality of a particular RADIC component is dependent on the representations used in mapping data between layers; at some application-dependent level, representations are either too abstract to be useful or too narrow to be applicable. For instance, a deliberative layer that produces a goal list composed of an ordered sequence of spatial locations to "visit" (such as in navigation or arm movement) requires translation of world-coordinates into egocentric-coordinates. While a RADIC component that integrates such layers may be applied to other tasks requiring spatial reasoning, it may not be adequate for, say, a higher level goal list that includes what to do at each location. RADIC provides a generic technique for integrating layers, but specificity is dependent on mapping implementation.

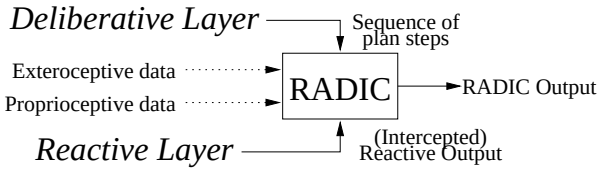


Figure 1: Functional diagram of the RADIC component showing (possible) inputs and output.

Functional mapping **F1** is satisfied by the `convertGoal` function, detailed below. The `updateRADIC` function is called at least as frequently as the reactive layer produces outputs (e.g., sends motor commands), and thus satisfies item **F3**. The functional mappings **F2** and **F4** are an inherent part of the RADIC component's operation: a goal is a discrete representation, while RADIC's output effects continuous motion (**F2**, either via discrete actions or via discrete motor commands), and the state of the deliberative layer is maintained in internal memory, while not requiring any state from the reactive layer other than its outputs (**F4**).

In more detail, the `updateRADIC` function takes as arguments the outputs from the reactive layer *R-out* (typically, intercepted motor commands), the changes as determined by sensors *S-change* (proprioceptive, exteroceptive, or missing), a sequence of goals from the delibera-

tive layer *D-goals* (which can be empty), and a strategy for goal attainment *G-strat*. The sequence of goals *D-goals* $= \langle G_1, G_2, ..., G_n \rangle$ is transformed, in accordance with item **F1**, from a "global" representation into an ego-centric representation $\langle \overline{G}_1, \overline{G}_2, ..., \overline{G}_n \rangle$ by the `convertGoal` function and stored in RADIC's internal memory M. `convertGoal` also associates a set of "blocking goals" B_G and a set of goal conditions C_G with each goal G that will depend on *G-strat* (in general, *G-strat* is a function that defines or influences the relation among G, B_G, and C_G). Blocking goals are used to impose a priority ordering on goals (goals that are blocked are not considered for action selection). Similarly, goal conditions are used to determine when goals have been achieved. A goal can then either be *active* or *inactive*, depending on whether or not its set of "blocking goals" is empty. A goal G is removed from M when at least one of its goal conditions $C \in C_G$ is met. The state of the robot is computed based on the old state using `updateState`, while the default action (i.e., output) is chosen in `defaultAction` based on *R-out*. All goals are then updated in `updateGoal` based on *R-out* and any possible changes detected via sensory inputs *S-change* and the active goals are used to determine the action of the robot via the `chooseAction` function based on the goal and the chosen action so far.

```
FUNCTION updateRADIC(R-out,S-change,D-goals,G-strat)
static M ← ∅
static currentState ← ∅
static oldState ← ∅
M ← M + convertGoal(D-goals, G-strat)
oldState ← currentState
currentState ← updateState(S-change)
action ← defaultAction(R-out)
for all G ∈ M do
  for all C ∈ C_G do
    if C_G then
      M ← M − {G}
    end if
  end for
  G ← updateGoal(R-out, S-change)
  if B_G = ∅ then
    action ← chooseAction(G, action)
  end if
end for
return action
```

Figure 2: The generic update algorithm for RADIC.

Note that the above algorithm is kept as generic as possible, allowing application to the broadest range of deliberative and reactive layers. To apply it to a particular architecture (e.g., to integrate two pre-existing layers), the five functions `convertGoal`, `updateState`, `defaultAction`, `updateGoal`, and `chooseAction` need to be implemented. RADIC can also be used in a standalone fashion with either type of layer, so long as the functionality of the other layer is implemented internally. Reactive layer functionality requires mapping goals to actions, while deliberative layer functionality requires appropriate internal representations (akin to the symbolic layer in SSS).

Unlike other hybrids, RADIC satisfies all of items **I1-4**. Minimal changes to each layer are required for integration (**I1**), as layer output is simply redirected as RADIC input; RADIC relies on internal data structures and functions for its operation. The RADIC component operates independently of either layer, performing its tasks in parallel with the layers' operation, responding to whatever input is received from either layer (**I2**). Computational cost is small (**I3**), as only state updates, goal list updates, and action choice are required. Updates of the goal list require both a measure of state change (via `updateState`) and a subsequent application of the calculated change for each point (via `updateGoal`). Only "active" goals are considered in modifications of the reactive layer's output at any given time. Finally, combining the layers improves overall performance (**I4**) by augmenting the reactive layer with planning capacities and providing the deliberative layer with the ability to act. Furthermore, custom improvements can easily be incorporated into the `updateState` (such as additional processing of *S-change*) and `updateGoal` (such as goal reordering or progress monitoring) functions.

4. CONCLUSION

RADIC has been experimentally validated and compared with other hybrid architectures in a trajectory following task. Beyond demonstrated performance improvements (and unlike other hybrid architectures), RADIC has the benefit of satisfying design desiderata **I1-4**, in the most extreme case actually substituting for a layer. The RADIC algorithm is not limited to navigation tasks, but can be used as a generic action sequencer in hybrid architectures for *any task*. RADIC can be attached to *any reactive layer* by virtue of its method of intercepting reactive output. Moreover, it can subsequently be used with *any deliberative layer* for which the `convertGoal` function is implemented.

5. REFERENCES

[1] J. Albus. 4-D/RCS reference model architecture for unmanned ground vehicles. In *Proc. of ICRA*, 2000.

[2] R. Arkin and T. Balch. AuRA: Principles and practice in review. *JETAI*, 9(2-3):175–189, 1997.

[3] R. Bonasso, J. Firby, E. Gat, D. Kortenkamp, D. Miller, and M. Slack. Experiences with an architecture for intelligent, reactive agents. *JETAI*, 9(2/3):237–256, Apr 1997.

[4] J. Connell. SSS: A hybrid architecture applied to robot navigation. In *Proc. of ICRA 1992*, pages 2719–2724, 1992.

[5] R. J. Firby. Task networks for controlling continuous processes. In *Artificial Intelligence Planning Systems*, pages 49–54, Chicago, IL, June 1994.

[6] E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *AAAI*, pages 809–815, 1992.

[7] R. Jensen and M. Veloso. Interleaving deliberative and reactive planning in dynamic multi-agent domains. In *Proc. of the AAAI Fall Symposium*. AAAI Press, 1998.

[8] K. Konolige, K. Myers, E. Ruspini, and A. Saffiotti. The Saphira architecture: A design for autonomy. *JETAI*, 9(1):215–235, 1997.

[9] D. Lyons and A. Hendriks. Planning as incremental adaptation of a reactive system. *Robotics and Autonomous Systems*, 14(4):255–288, June 1995.

[10] P. Maes. Situated agents can have goals. In P. Maes, editor, *Designing Autonomous Agents*, pages 49–70. MIT Press, 1990.

[11] H. Nwana. Software agents: An overview. *Knowledge Engineering Review*, 11(2):205–244, 1995.

A Constraint Optimization Framework for Fractured Robot Teams *

Mary Koes†
The Robotics Institute
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
mberna@cs.cmu.edu

Katia Sycara
The Robotics Institute
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
katia@cs.cmu.edu

Illah Nourbakhsh
The Robotics Institute
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
illah@cs.cmu.edu

ABSTRACT

In dangerous and uncertain environments initial plans must be revised. Communication failures hamper this replanning. We introduce *fractured subteams* as a novel formalism for modeling breakdowns in communication. We present a hybrid approach that employs distributed coordination mechanisms to provide robustness to these communication breakdowns and exploits opportunistic centralization. By modeling the problem as a mixed integer linear programming problem, we are able to apply constraint optimization techniques to efficiently find optimal or near optimal solutions to the difficult class of time critical tight coordination team planning problems. We then demonstrate that explicitly reasoning about communication failures through the incorporation of selective disruption minimization can improve team performance.

Keywords

Autonomous robots and robot teams, cooperation and coordination among agents, multi-agent planning

1. BACKGROUND

Teams of robots are necessary for a number of applications in which a single robot is unable or inefficient at accomplishing the goals. For example, time critical domains such as search and rescue, where lives are at stake, require careful consideration of plan efficiency. While some domains consist entirely of tasks that may be independently completed by individual robots permitting *loose coordination*, we are interested in the more challenging problem of *tight coordination* [2] where robots must work together to accomplish a *joint goal*. This requires robots to simultaneously solve the dual $\mathcal{NP}$-hard scheduling and task allocation problems with additional consideration for path planning [3].

The resulting domain of *time critical tight coordination team planning problems* are characterized by a set of robots, $\mathcal{R}$, with heterogeneous capabilities seeking to accomplish a set of joint goals, $\mathcal{G}$, with heterogeneous requirements. These goals are distributed throughout some physical environment, E. Additional system constraints, $\mathcal{C}$, may apply to robots, goals, or resources in the problem. The objective is to maximize team utility by accomplishing goals while goal rewards decrease over time to 0 at some time limit, T_{max}.

For teams of robots operating in physical environments, coordination inefficiencies may by magnified by the spatial constraints inherent to the problem. The time needed for robots to traverse the environment often dominates team performance. Consequently, it is beneficial to generate an optimal or near optimal plan at the start. In previous work [3], we describe COCoA, a Constraint Optimization Coordination Architecture, that maps this team planning problem to a mixed integer linear programming (MILP) problem and then combines domain specific heuristics and standard integer programming algorithms to efficiently generate optimal or near optimal plans. The resulting *team plan* generated by COCoA includes a schedule for each robot on the team broken down into the three components necessary to accomplish each of the robot's goals: time traveling to the goal, time waiting for teammates, and time working on the goal.

Unfortunately, few domains are so benign that this original plan can be successfully executed. Robots must interleave planning and execution as robots fail or are incapacitated, new robots join the team or receive new capabilities, goals are added and removed, goal requirements, rewards, or durations are changed, paths through the environment are found to be longer or shorter than believed, system constraints are added or removed, or the time limit is changed. Events that necessitate replanning are known as *replanning catalysts*. The discovery of replanning catalysts is beyond the scope of this paper but is discussed in other work [2].

Due to the physical nature of the environment, robots may not be in full communication at any given time. Although communication failure is a common phenomenon in multirobot coordination, models for these communication breakdowns are relatively primitive. Multirobot coordination architectures that address the problem assume any given message is lost with some uniform probability. However, since robots generally depend on peer to peer wireless

*This work is supported by NSF Award IIS-0205526. The authors would also like to thank the ILOG corporation for their generous support.

†Student Author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

RF networks for communication, signal strength between two robots varies according to $\frac{1}{d^n}$ where d is the distance between the robots and n is the path loss exponent that is dependent on the environment [6]. Since robot communication is frequently based on TCP which is a reliable protocol, dropped packets are relatively unimportant. A more accurate model for robots which frequently have some limited range of communication is to model subsets of the team with good local communication but little or no communication between subsets. We introduce the terminology *fractured subteam* to refer to subsets of robots in the original team that are able to communicate with each other, directly or indirectly, but not with the rest of the team.

2. FRACTURED SUBTEAMS

A fractured subteam consists of a set of robots that jointly believe in a time critical tight coordination team planning problem, share a team plan, share a belief evolution model, are able to communicate with each other through transitive closure, and are unable to communicate with any robot not in the fractured subteam. A *belief evolution model* is a data structure that represents this fractured subteam's knowledge of beliefs of robots in other fractured subteams. Since all robots start with a common problem and team plan, all robots' belief evolution models contain a common element.

As robots seek to execute their team plan in dangerous and uncertain environments, they may discover that the problem and environment which they encounter does not match the problem for which they planned requiring them to refine their model of the problem and replan. This replanning must be performed quickly due to the time sensitive nature of the domain; COCoA has algorithms that can find good solutions in real time [3]. The second problem arises when failures in communication prevent the refined problem and solution from being disseminated to all team members resulting in inconsistencies. Fractured subteams force distributed replanning.

We extend COCoA from a purely centralized solver to a hybrid architecture. This hybrid architecture employs opportunistic centralization but is fundamentally distributed in that each fractured subteam is unable to communicate with other fractured subteams requiring distributed replanning. Each fractured subteam maintains its own knowledge from all previous subteams encountered which it uses for this replanning. Fractured subteams are themselves another variable in the system which must be considered while replanning.

Since COCoA was designed to be a centralized planner, only a single robot was required to possess the planning capabilities and complete problem and team plan. Other than this centralized planning robot, robots required only their individual tightly coupled schedule. In the distributed version of COCoA, each robot possesses full planning capabilities and a copy of the complete problem and team plan. The increase in communication cost is marginal. When a replanning catalyst is discovered, the architecture leverages the nature of a fractured subteam for opportunistic centralization. A robot from that fractured subteam is elected to replan and broadcast the results to the subteam.

Although the replanning is locally centralized, COCoA is distributed with respect to the individual fractured subteams. Each fractured subteam maintains its own problem and team plan. Each robot carries with it the complete knowledge of the fractured subteam to which it belongs. When two fractured subteams, F_1 and F_2, are merged into a single fractured subteam, their problems, team plans, and belief evolution models must likewise be merged. We have developed a simple algorithm for this merge, though we intend to investigate more sophisticated methods in the future since merging different beliefs is a deep research problem.

1. If the currently held problem and team plan of F_1 match the currently held problem and team plan of F_2, no changes need to be made.
2. If the currently held problem and team plan of F_1 match a state in the belief evolution model of F_2 and the team plan of F_2 is valid given the current status of the members of F_1 and F_2, all members of the merged fractured subteam adopt the problem, team plan, and belief evolution model of F_2.
3. By symmetry, the previous rule applies if the currently held problem and team plan of F_2 match a state in the belief evolution model of F_1.
4. If none of the above hold, create a new problem by combining the team planning problems of F_1 and F_2 and generate a new team plan to the resulting problem.

While full replanning works well in perfect communication, it tends to perform poorly with fractured subteams due to the inconsistencies in team plans. We can improve performance by explicitly reasoning about communication failure and fractured subteam composition during replanning and preferring solutions that only change the commitments of robots in the fractured subteam. The changes in coordination commitments are known as *disruption*.

One of the advantages of the MILP problem formulation in COCoA is the relative ease in combining multiple objective functions. We selectively minimize disruption by adding a cost function. There are several possible cost functions. We penalize near term changes more than long term changes since there is a higher probability that changes in the plan will be discovered in time for the robots to respond if the change is sufficiently far in the future. We then define our disruption objective function as follows. Minimize:

$$\sum_{m \text{ s.t. } G^m \in \mathcal{G}} [Cost(G^m) + \sum_{n \text{ s.t. } R^n \in \mathcal{R}} Cost(R^n, G^m)]$$

In general, every goal assigned to a robot represents a commitment to the team. However, not all changes to commitments are undesirable. If a robot is in the fractured subteam and aware of the replanning catalyst, some changes in the robot's schedule are advantageous as they increase system performance. Since the problem is tightly coupled, distinguishing between commitments that should be preserved and those that may be modified is challenging. We have developed *commitment graphs* to represent dependencies between robots in different fractured subteams. Commitment graphs can be easily generated from the team plan as described in Algorithm 1.

The cost function for each goal, $Cost(G^m)$, can be computed from the commitment graph and the current membership of the fractured subteam. The original time at which the goal was scheduled is denoted by $oldTime$. The new time at which the goal is to be scheduled (a variable in the MILP) is $newTime$, and the system time when replanning, is $replanTime$. If G^m has edges linking it to nodes of robots not in the fractured subteam and G^m was originally sched-

Algorithm 1 Build commitment graph

```
for all Goals G^m ∈ 𝒢 such that isScheduled(G^m) do
    Add goal node to graph
end for
for all Robots R^n ∈ ℛ do
    Add robot node to graph
    S_{R^n} = last known schedule for R^n in belief evolution model
    for all Goals in this robot's schedule do
        Add edge with robot coordination commitments from robot node to goal node
    end for
end for
for all Temporal goal constraints C_j ∈ 𝒞 do
    Add edge C_j between constrained goal nodes
end for
```

uled, canceling the goal results in a penalty of

$$\alpha_{cancel}(1 - \frac{oldTime - replanTime}{T_{max} - replanTime})$$

and changing the start time results in a penalty of

$$\alpha_{insert}(1 - \frac{newTime - replanTime}{T_{max} - replanTime})$$

The cost function for each robot not a member of the fractured subteam, $Cost(R^n, G^m)$, can also be calculated. If R^n is not currently allocated to G^m, the cost adding the goal into robot n's schedule is determined by equation (1).

$$\beta_{insert}(1 - \frac{newTime - replanTime}{T_{max} - replanTime}) \quad (1)$$

Similarly, the cost of removing a goal from robot n's schedule is determined by equation (2).

$$\beta_{cancel}(1 - \frac{oldTime - replanTime}{T_{max} - replanTime}) \quad (2)$$

We model the cost of changing the time of a goal in robot n's schedule as canceling and then adding a goal or the sum of equations (1) and (2).

The constants α and β may be tuned depending on the desired emphasis but should match the level of communication disruption in the environment. The new MILP combines the reward maximizing objective function [3] and the disruption minimization objective function.

3. EVALUATION

We conducted preliminary experiments to analyze the effect of fractured subteams on system performance. We used an abstract simulator with a randomly generated environment, 5 robots, and 10 goals. The environments are randomly partitioned into various blackout zones which impose the communication limitations, resulting in a set of fractured subteams. The number of fractured subteams is dependent on the topology of the environment and the composition of these subteams changes as robots traverse the environment. As an starting point for understanding the effect of communication breakdown on performance, we conducted a simple set of experiments in which one robot in the team was randomly disabled at some random time early in the plan.

We compared three different replanning strategies: *no replanning* where robots attempt to execute their original schedules, *full replanning* in which robots replan without consideration of communication failure, and *replanning with selective disruption minimization*. The results vary depending on the topology of the environment but we found that, in general, if the communication failures are minimal, full replanning outperforms the no replanning strategy, but as violations of the implicit assumption of full communication in the full replanning strategy increase, full replanning actually decreases team performance. By applying selective disruption minimization, as the number of partitions in the environment (communication failures) increases, the performance eventually converges to the *no replanning* strategy. Choosing α and β to match the amount of communication failure in the environment is an important challenge that we will address in future work.

4. RELATED WORK AND CONCLUSIONS

Market-based algorithms are a popular approach for multirobot coordination and have been successfully demonstrated in a variety of domains. A survey of the field [2] cites the lack of performance guarantees as one of the biggest drawbacks to current market-based approaches and suggests opportunistic centralization in future challenges. Our coordination approach addresses both of these issues.

In the MAS literature, token-based algorithms have been shown empirically to perform well in large scale teams [7]. However, token-based algorithms fail to provide the performance guarantees necessary for time critical domains. Existing work has assumed perfect communication. Pynadath and Tambe provide the COM-MTDP model for reasoning about the cost of communication [5] but do not explicitly model communication failure. The COM-MTDP model does not apply to time critical team planning problems since it uses a Markov assumption which does not immediately accommodate the time varying rewards or temporal constraints in our formulation of the problem. Distributed constraint optimization (DCOP) has been successfully applied to meeting scheduling [4] and to the multiagent plan coordination problem (MPCP) [1]. However, MPCPs require loose rather than tight coordination.

This preliminary work in developing an architecture interleaving planning and execution with communication failures for tightly coupled robot teams is promising and warrants additional investigation.

5. REFERENCES

[1] J. S. Cox, E. H. Durfee, and T. Bartold. A distributed framework for solving the multiagent plan coordination problem. In *AAMAS '05*, pages 821–827, New York, NY, USA, 2005. ACM Press.

[2] M. B. Dias, R. M. Zlot, N. Kalra, and A. T. Stentz. Market-based multirobot coordination: A survey and analysis. Technical Report CMU-RI-TR-05-13, Pittsburgh, PA, April 2005.

[3] M. Koes, I. Nourbakhsh, and K. Sycara. Constraint optimization coordination architecture for search and rescue robotics. In *Submitted to the International Conference on Robotics and Automation*, 2006.

[4] P. J. Modi and M. Veloso. Multiagent meeting scheduling with rescheduling. In *Distributed Constraint Reasoning, (DCR) 2004*, 2004.

[5] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[6] V. Rodoplu and T. Meng. Minimum energy mobile wireless networks. *IEEE J. Select. Areas Commun.*, 17(8):1333 – 1344, Aug. 1999.

[7] Y. Xu, P. Scerri, B. Yu, S. Okamoto, M. Lewis, and K. Sycara. An integrated token-based algorithm for scalable coordination. In *AAMAS '05*, pages 407–414, New York, NY, USA, 2005. ACM Press.

Computational complexity in agent systems

Junta Distributions and the Average-Case Complexity of Manipulating Elections

Ariel D. Procaccia Jeffrey S. Rosenschein
School of Engineering and Computer Science
The Hebrew University of Jerusalem
Jerusalem, Israel
{arielpro, jeff}@cs.huji.ac.il

ABSTRACT

Encouraging voters to truthfully reveal their preferences in an election has long been an important issue. Previous studies have shown that some voting protocols are hard to manipulate, but predictably used $\mathcal{NP}$-hardness as the complexity measure. Such a *worst-case* analysis may be an insufficient guarantee of resistance to manipulation.

Indeed, we demonstrate that $\mathcal{NP}$-hard manipulations may be tractable in the *average-case*. For this purpose, we augment the existing theory of average-case complexity with new concepts; we consider elections distributed with respect to *junta distributions*, which concentrate on hard instances, and introduce a notion of *heuristic* polynomial time. We use our techniques to prove that a family of important voting protocols is susceptible to manipulation by coalitions, when the number of candidates is constant.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity;
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*;
J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

General Terms

Algorithms, Theory, Economics

Keywords

Computational complexity, Voting

1. INTRODUCTION

In multiagent environments, it may be the case that different agents have diverse preferences, and it is therefore important to find a way to aggregate agent preferences. A general scheme for preference aggregation is *voting*: the agents reveal their preferences by ranking a set of candidates, and a winner is determined according to a voting protocol. The candidates can be various entities such as beliefs or plans, and indeed may be potential real-life parliament members.

Things are made complicated by the fact that in many settings (as in reality) the agents are self-interested. Such an agent may reveal its preferences untruthfully, if it believes this would make the final outcome of the elections more favorable for it. Consequently, the outcome may be one that does not maximize social welfare. This problem is provably acute: it is known [8, 10] that, for elections with three or more candidates, in any voting protocol that is non-dictatorial,[1] there are elections where an agent is better off by voting untruthfully.

Fortunately, it is reasonable to make the assumption that the agents are computationally bounded. Therefore, although in principle an agent may be able to manipulate an election, the computation required may be infeasible. This has motivated researchers to study the computational complexity of manipulating voting protocols. It has long been known [3] that there are voting protocols that are $\mathcal{NP}$-hard to manipulate by a single voter. Recent results by Conitzer and Sandholm [5, 4] show that some manipulations of common voting protocols are $\mathcal{NP}$-hard, even for a small number of candidates. Moreover, in [6], it is shown that adding a pre-round to some voting protocols can make manipulations hard (even $\mathcal{PSPACE}$-hard in some cases). Elkind and Lipmaa [7] show that the notion of pre-round, together with one-way functions, can be used to construct protocols that are hard to manipulate even by a large minority fraction of the voters.

In Computer Science, the notion of hardness is usually considered in the sense of worst-case complexity. Not surprisingly, most results on the complexity of manipulation use $\mathcal{NP}$-hardness as the complexity measure. However, it may still be the case that most instances of the problem are easy to manipulate.

A relatively little-known theory of average case complexity exists [11]; that theory introduces the concept of distributional problems, and defines what a reduction between distributional problems is. It is also known that average-case complete problems exist (albeit artificial ones, such as a distributional version of the halting problem).

Sadly, it is very difficult to show that a certain problem is average-case complete, and such results are known only for a handful of problems. Additionally, the goal of the

[1] In a dictatorial protocol, there is an agent that dictates the outcome regardless of the others' choices.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

existing theory is to define when a problem is *hard* in the average-case; it does not provide criteria for deciding when a problem is *easy*. A step towards showing that a manipulation is easy on average was made in [7]. It involves an analysis of the plurality protocol with a pre-round, but focuses on a very specific distribution, which does not satisfy some basic desiderata as to what properties an "interesting" distribution should have.

In this paper, we engage in a novel average-case analysis, based on criteria we propose. Coming up with an "interesting" distribution of problem instances with respect to which the average-case complexity is computed is a difficult task, and the solution may be controversial. We analyze problems whose instances are distributed with respect to a *junta distribution*. Such a distribution must satisfy several conditions, which (arguably) guarantee that it focuses on instances that are harder to manipulate. We consider a protocol to be *susceptible* to manipulation when there is a polynomial time algorithm that can usually manipulate it: the probability of failure (when the instances are distributed according to a junta distribution) must be inverse-polynomial. Such an algorithm is known as a *heuristic* polynomial time algorithm.

We use these new methods to prove our main result: an important family of protocols, called *scoring* protocols, is susceptible to coalitional manipulation when the number of candidates is constant. Specifically, we contemplate *sensitive* scoring protocols, which include such well-known protocols as Borda and Veto. To accomplish this task, we define a natural distribution μ^* over the instances of a well-defined coalitional manipulation problem, and show that this is a junta distribution. Furthermore, we present the manipulation algorithm Greedy, and prove that it usually succeeds with respect to μ^*.

We also show that all protocols are susceptible to a certain setting of manipulation, where the manipulator is unsure about the others' votes. This result depends upon a basic conjecture regarding junta distributions, but also has implications that transcend our specific definition of these distributions.

In Section 2, we outline some important voting protocols, and properly define the manipulation problems we shall discuss. In Section 3, we formally introduce the tools for our average case analysis: junta distributions, heuristic polynomial time, and susceptibility to manipulations. In Section 4 we prove our main result: sensitive scoring protocols are susceptible to coalitional manipulation with few candidates. In Section 5, we discuss the case when a single manipulator is unsure about the other voters' votes. Finally, in Section 6, we present conclusions and directions for future research.

2. PRELIMINARIES

We first describe some common voting protocols and formally define the manipulation problems with which we shall deal. Next, we introduce a useful lemma from probability theory.

2.1 Elections and Manipulations

An election consists of a set C of m candidates, and a set V of n voters, who provide a total order on the candidates. An election also includes a winner determination function from the set of all possible combinations of votes to C. We note that throughout this paper, $m = O(1)$, so the complexity results are in terms of n.

Different voting protocols are distinguished by their winner determination functions. The protocols we shall discuss are:

- *Scoring protocols:* A scoring protocol is defined by vector $\vec{\alpha} = \langle \alpha_1, \alpha_2, \ldots, \alpha_m \rangle$, such that $\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_m$ and $\alpha_i \in \mathbb{N} \cup \{0\}$. A candidate receives α_i points for each voter which ranks it in the i'th place. Examples of scoring protocols are:
  - *Plurality:* $\vec{\alpha} = \langle 1, 0, \ldots, 0, 0 \rangle$.
  - *Veto:* $\vec{\alpha} = \langle 1, 1, \ldots, 1, 0 \rangle$.
  - *Borda:* $\vec{\alpha} = \langle m-1, m-2, \ldots, 1, 0 \rangle$.
- *Copeland:* For each possible pair of candidates, simulate an election; a candidate wins such a pairwise election if more voters prefer it over the opponent. A candidate gets 1 point for each pairwise election it wins, and -1 for each pairwise election it loses.
- *Maximin:* A candidate's score in a pairwise election is the number of voters that prefer it over the opponent. The winner is the candidate whose minimum score over all pairwise elections is highest.
- *Single Transferable Vote (STV):* The election proceeds in rounds. In each round, the candidate's score is the number of voters that rank it highest among the remaining candidates; the candidate with the lowest score is eliminated.

Remark 1. We assume that tie-breaking is always adversarial to the manipulator.[2]

In the case of weighted votes, a voter with weight $k \in \mathbb{N}$ is naturally regarded as k voters who vote unanimously. In this paper, we consider weights in $[0, 1]$. This is equivalent, since any set of integer weights in the range $1, \ldots,$ poly$n$ can be scaled down to weights in the segment $[0, 1]$ with $O(\log n)$ bits of precision.

The main results of the paper focus on scoring protocols. We shall require the following definition:

Definition 1. Let P be a scoring protocol with parameters $\vec{\alpha} = \langle \alpha_1, \alpha_2, \ldots, \alpha_m \rangle$. We say that P is *sensitive* iff $\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_{m-1} > \alpha_m = 0$ (notice the strict inequality on the right).

In particular, Borda and Veto are sensitive scoring protocols.

Remark 2. Generally, from any scoring protocol with $\alpha_{m-1} > \alpha_m$, an equivalent sensitive scoring protocol can be obtained by subtracting α_m on a coordinate-by-coordinate basis from the vector $\vec{\alpha}$. Moreover, observe that if a protocol is a scoring protocol but is not sensitive, and $\alpha_m = 0$, then $\alpha_{m-1} = 0$. In this case, for three candidates it is equivalent to the plurality protocol, for which most manipulations are tractable even in the worst-case. Therefore, it is sufficient to restrict our results to sensitive scoring protocols.

[2]This is a standard assumption, also made, for example, in [5, 4].

We next consider some types of manipulations, state the appropriate complexity results, and introduce some notations.

REMARK 3. We discuss the *constructive* cases, where the goal is trying to make a candidate win, as opposed to *destructive* manipulation, where the goal is to make a candidate lose. Constructive manipulations are always at least as hard (in the worst-case sense) as their destructive counterparts, and in some cases strictly harder (if one is able to determine whether p can be made to win, one can also ask whether any of the other $m-1$ candidates can be made to win, thus making p lose).

DEFINITION 2. In the INDIVIDUAL-MANIPULATION problem, we are given all the other votes, and a preferred candidate p. We are asked whether there is a way for the manipulator to cast its vote so that p wins.

Bartholdi and Orlin [3] show that IM is $\mathcal{NP}$-complete in Single Transferable Vote, provided the number of candidates is unbounded. However, the problem is in $\mathcal{P}$ for most voting schemes, and hence will not be studied here.

DEFINITION 3. In the COALITIONAL-WEIGHTED-MANIPULATION (CWM) problem, we are given a set of weighted votes S, the weights of a set of votes T which have not been cast, and a preferred candidate p. We are asked whether there is a way to cast the votes in T so that p wins the election.

We know [5, 4] that CWM is $\mathcal{NP}$-complete in Borda, Veto and Single Transferable Vote, even with 3 candidates, and in Maximin and Copeland with at least 4 candidates.

The CWM version that we shall analyze, which is specifically tailored for scoring protocols, is a slightly modified version whose analysis is more straightforward:

DEFINITION 4. In the SCORING-COALITIONAL-WEIGHTED-MANIPULATION (SCWM) problem, we are given an initial score $S[c]$ for each candidate c, the weights of a set of votes T which have not been cast, and a preferred candidate p. We are asked whether there is a way to cast the votes in T so that p wins the election.

$S[c]$ can be interpreted as c's total score from the votes in S. However, we do not require that there exist a combination of votes that actually induces $S[c]$ for all c.

DEFINITION 5. In the UNCERTAIN-VOTES-WEIGHTED-EVALUATION (UVWE) problem, we are given a weight for each voter, a distribution over all the votes, a candidate p, and a number $r \in [0,1]$. We are asked whether the probability of p winning is greater than r.

DEFINITION 6. In the UNCERTAIN-VOTES-WEIGHTED-MANIPULATION (UVWM) problem, we are given a single manipulative voter with a weight, weights for all other voters, a distribution over all the others' votes, a candidate p, and a number r, where $r \in [0,1]$. We are asked whether the manipulator can cast its vote so that p wins with probability greater than r.

If CWM is $\mathcal{NP}$-hard in a protocol, then UVWE and UVWM are also $\mathcal{NP}$-hard in it [5]. These problems will be studied in Section 5. We make the assumption that the given distributions over the others' votes can be sampled in polynomial time.

2.2 Chernoff's Bounds

The following lemma will be of much use later on. Informally, it states that the average of independent identically distributed (i.i.d.) random variables is almost always close to the expectation.

LEMMA 1 (CHERNOFF'S BOUNDS). *Let $X_1, \ldots, X_t$ be i.i.d. random variables such that $a \leq X_i \leq b$ and $\mathrm{E}[X_i] = \mu$. Then for any $\epsilon > 0$, it holds that:*

- $\Pr[\frac{1}{t}\sum_{i=1}^{t} X_i \geq \mu + \epsilon] \leq e^{-2t\frac{\epsilon^2}{(b-a)^2}}$
- $\Pr[\frac{1}{t}\sum_{i=1}^{t} X_i \leq \mu - \epsilon] \leq e^{-2t\frac{\epsilon^2}{(b-a)^2}}$

3. JUNTA DISTRIBUTIONS AND SUSCEPTIBLE MECHANISMS

In this section we lay the mathematical foundations required for an average-case analysis of the complexity of manipulations. All of the definitions are as general as possible; they can be applied to the manipulation of any mechanism, not merely to the manipulation of voting protocols.

We describe a distribution over the instances of a problem as a collection of distributions $\mu_1, \ldots, \mu_n, \ldots$, where μ_n is a distribution over the instances x such that $|x| = n$. We wish to analyze problems whose instances are distributed with respect to a distribution which focuses on hard-to-manipulate instances. Ideally, we would like to insure that if one manages to produce an algorithm which can usually manipulate instances according to this distinguished "difficult" distribution, the algorithm would also usually succeed when the instances are distributed with respect to most other reasonable distributions.

DEFINITION 7. Let $\mu = \{\mu_n\}_{n\in\mathbb{N}}$ be a distribution over the possible instances of an $\mathcal{NP}$-hard manipulation problem M. μ is a *junta* distribution if and only if μ has the following properties:

1. Hardness: The restriction of M to μ is the manipulation problem whose possible instances are only:
$$\bigcup_{n\in\mathbb{N}} \{x : |x| = n \wedge \mu_n(x) > 0\}.$$
Deciding this restricted problem is still $\mathcal{NP}$-hard.
2. Balance: There exist a constant $c > 1$ and $N \in \mathbb{N}$ such that for all $n \geq N$:
$$\frac{1}{c} \leq \Pr_{x\sim\mu_n}[M(x) = 1] \leq 1 - \frac{1}{c}.$$
3. Dichotomy: for all n and instances x such that $|x| = n$:
$$\mu_n(x) \geq 2^{-\mathrm{poly}n} \vee \mu_n(x) = 0.$$

If M is a voting manipulation problem, we also require the following property:

4. Symmetry: Let v be a voter whose vote is given, let $c_1, c_2 \neq p$ be two candidates, and let $i \in \{1, \ldots, m\}$. The probability that v ranks c_1 in the i'th place is the same as the probability that v ranks c_2 in the i'th place.

If M is a coalitional manipulation problem, we also require the following property:

5. Refinement: Let x be an instance such that $|x| = n$ and $\mu_n(x) > 0$; if all colluders voted identically, then p would not be elected.

The name "junta distribution" comes from the idea that in such a distribution, relatively few "powerful" and difficult instances represent all the other problem instances. Alternatively, our intent is to have a few problematic distributions (the family of junta distributions) convincingly represent all other distributions with respect to the average-case analysis.

The first three properties are basic, and are relevant to problems of manipulating any mechanism. The definition is modular, and additional properties may be added on top of the basic three, in case one wishes to analyze a mechanism which is not a voting protocol.

The exact choice of properties is of extreme importance (and, as we mentioned above, may be arguable). We shall briefly explain our choices. Hardness is meant to insure that the junta distribution contains hard instances. Balance guarantees that a trivial algorithm which always accepts (or always rejects) has a significant chance of failure. The dichotomy property helps in preventing situations where the distribution gives a (positive but) negligible probability to all the hard instances, and a high probability to several easy instances.

We now examine the properties that are specific to manipulation problems. The necessity of symmetry is best explained by an example. Consider CWM in STV with $m \geq 3$. One could design a distribution where p wins if and only if a distinguished candidate loses the first round. Such a distribution could be tailored to satisfy the other conditions, but misses many of the hard instances. In the context of SCWM, we interpret symmetry in the following way: for every two candidates $c_1, c_2 \neq p$ and $y \in \mathbb{R}$,

$$\Pr_{x \sim \mu_n}[S[c_1] = y] = \Pr_{x \sim \mu_n}[S[c_2] = y].$$

Refinement is less important than the other four properties, but seems to help in concentrating the probability on hard instances. Observe that refinement is only relevant to coalitional manipulation; we believe that in the analysis of individual voting manipulation problems, the first four properties are sufficient.

Definition 8. [11] A *distributional problem* is a pair $\langle L, \mu \rangle$ where L is a decision problem and μ is a distribution over the set $\{0,1\}^*$ of possible inputs.

Informally, an algorithm is a heuristic polynomial time algorithm for a distributional problem if it runs in polynomial time, and fails only on a small fraction of the inputs. We now give a formal definition; this definition is inspired by [11] (there the same name is used for a somewhat different definition).

Definition 9. Let M be a manipulation problem and let $\langle M, \mu \rangle$ be a distributional problem.

1. An algorithm A is a *deterministic heuristic polynomial time* algorithm for the distributional manipulation problem $\langle M, \mu \rangle$ if A always runs in polynomial time, and there exists a polynomial p and $N \in \mathbb{N}$ such that for all $n \geq N$:

$$\Pr_{x \sim \mu^n}[A(x) \neq M(x)] < \frac{1}{p(n)}. \quad (1)$$

2. Let A be a probabilistic algorithm, which uses a random string s. A is a *probabilistic heuristic polynomial time* algorithm for the distributional manipulation problem $\langle M, \mu \rangle$ if A always runs in polynomial time, and there exists a polynomial p and $N \in \mathbb{N}$ such that for all $n \geq N$:

$$\Pr_{x \sim \mu^n, s}[A(x) \neq M(x)] < \frac{1}{p(n)}. \quad (2)$$

Probabilistic algorithms have two potential sources of failure: an unfortunate choice of input, or an unfortunate choice of random string s. The success or failure of deterministic algorithms depends only on the choice of input.

We now combine all the definitions introduced in this section in an attempt to establish when a mechanism is susceptible to manipulation in the average case. The following definition abuses notation a bit: M is both used to refer to the manipulation itself, and the corresponding decision problem.

Definition 10. We say that a mechanism is *susceptible* to a manipulation M if there exists a junta distribution μ, such that there exists a deterministic/probabilistic heuristic polynomial time algorithm for $\langle M, \mu \rangle$.

4. SUSCEPTIBILITY TO SCWM

Recall [5, 4] that in Borda and Veto, CWM is $\mathcal{NP}$-hard, even with 3 candidates. Since Borda and Veto are examples of sensitive scoring protocols, we would like to know how resistant this family of protocols really is with respect to coalitional manipulation. In this section we use the methods from the previous section to prove our main result:

Theorem 1. *Let P be a sensitive scoring protocol. Then P, with candidates $C = \{p, c_1, \ldots, c_m\}$, $m = O(1)$, is susceptible to SCWM.*

Intuitively, the instances of CWM (or SCWM) which are hard are those that require a very specific partitioning of the voters in T to subsets, where each subset votes unanimously. These instances are rare in any reasonable distribution; this insight will ultimately yield the theorem.

The following proposition generalizes Theorem 1 of [5] and Theorem 2 of [4], and justifies our focus on the family of sensitive scoring protocols. A stronger version of Proposition 2 has been independently proven in [9]. Nevertheless, we include our proof, since it will be required in proving the hardness property of a junta distribution we shall design.

Proposition 2. *Let P be a sensitive scoring protocol. Then CWM in P is $\mathcal{NP}$-hard, even with 3 candidates.*

Definition 11. In the Partition problem, we are given a set of integers $\{k_i\}_{i \in [t]}$, summing to $2K$, and are asked whether a subset of these integers sum to K.

It is well-known that Partition is $\mathcal{NP}$-complete.

PROOF OF PROPOSITION 2. We reduce an arbitrary instance of PARTITION to the following CWM instance. There are 3 candidates, a, b, and p. In S, there are $K(4\alpha_1-2\alpha_2)-1$ voters voting $a \succ b \succ p$, and $K(4\alpha_1 - 2\alpha_2) - 1$ voters voting $b \succ a \succ p$. In T, for every k_i there is a vote of weight $2(\alpha_1+\alpha_2)k_i$. Observe that from S, both a and b get $(K(4\alpha_1 - 2\alpha_2) - 1)(\alpha_1 + \alpha_2)$ points.

Assume first that a partition exists. Let the voters in T in one half of the partition vote $p \succ a \succ b$, and let the other half vote $p \succ b \succ a$. By this vote, a and b each have

$$(K(4\alpha_1 - 2\alpha_2) - 1)(\alpha_1 + \alpha_2) + 2K(\alpha_1 + \alpha_2)\alpha_2$$
$$= (\alpha_1 + \alpha_2)(4K\alpha_1 - 1)$$

votes, while p has $(\alpha_1 + \alpha_2)4K\alpha_1$ points; thus there is a manipulation.

Conversely, assume that a manipulation exists. Clearly there must exist a manipulation where all the voters in T vote either $p \succ a \succ b$ or $p \succ b \succ a$, because the colluders do not gain anything by not placing p at the top in a scoring protocol. In this manipulation, p has $(\alpha_1+\alpha_2)4K\alpha_1$ points, while a and b already have $(K(4\alpha_1 - 2\alpha_2) - 1)(\alpha_1 + \alpha_2)$ points from S. Therefore, a and b must gain less than $(2\alpha_2K + 1)(\alpha_1 + \alpha_2)$ points from the voters in T. Each voter corresponding to k_i contributes $2(\alpha_1+\alpha_2)\alpha_2k_i$ points; it follows that the sum of the k_i corresponding to the voters voting $p \succ a \succ b$ is less than $K + \frac{1}{2\alpha_2}$, and likewise for the voters voting $p \succ b \succ a$. Equivalently, the sum can be at most K, since all k_i are integers and $\alpha_2 \geq 1$. In both cases the sum must be at most K; hence, this is a partition. □

Since an instance of CWM can be translated to an instance of SCWM in the obvious way, we have:

COROLLARY 3. *Let P be a sensitive scoring protocol. It holds that SCWM in P is $\mathcal{NP}$-hard, even with 3 candidates.*

4.1 A Junta Distribution

Let $w(v)$ denote the weight of voter v, and let W denote the total weight of the votes in T; P is a sensitive scoring protocol. We denote $|T| = n$: the size of T is the size of the instance.

Consider a distribution $\mu^* = \{\mu_n^*\}_{n\in\mathbb{N}}$ over the instances of CWM in P, with $m+1$ candidates $p, c_1, \ldots, c_m$, where each μ_n^* is induced by the following sampling algorithm:

1. $\forall v \in T$: Randomly and independently choose $w(v) \in [0,1]$ (up to $O(\log n)$ bits of precision).
2. $\forall i \in \{1,\ldots,m\}$: Randomly and independently choose $S[c_i] \in [(\alpha_1 - \alpha_2)W, \alpha_1 W]$ (up to $O(\log n)$ bits of precision).

We assume that $S[p] = 0$, i.e., all voters in S rank p last. This assumption is not a restriction. If it holds for a candidate c that $S[c] \leq S[p]$, then candidate c will surely lose, since the colluders all rank p first. Therefore, if $S[p] > 0$, we may simply normalize the scores by subtracting $S[p]$ from the scores of all candidates. This is equivalent to our assumption.

REMARK 4. We believe that μ^* is the most natural distribution with respect to which coalitional manipulation in scoring protocols should be studied. Even if one disagrees with the exact definition of junta distribution, μ^* should satisfy many reasonable conditions one could produce.

We shall, of course, (presently) prove that the distribution possesses the properties of a junta distribution.

PROPOSITION 4. *Let P be a sensitive scoring protocol. Then μ^* is a junta distribution for SCWM in P with $C = \{p, c_1, \ldots, c_m\}$, and $m = O(1)$.*

PROOF. We first observe that the dichotomy and symmetry conditions are obviously satisfied.

The proof of the hardness property relies on the reduction from PARTITION in Proposition 2. The reduction generates instances x of CWM in P with 3 candidates, where $W = 4(\alpha_1 + \alpha_2)K$, and

$$S[a] = S[b]$$
$$= (K(4\alpha_1 - 2\alpha_2) - 1)(\alpha_1 + \alpha_2)$$
$$= (\alpha_1 - \alpha_2/2)W - (\alpha_1 + \alpha_2),$$

for some K that originates in the PARTITION instance. These instances satisfy $(\alpha_1 - \alpha_2)W \leq S[a], S[b] \leq \alpha_1 W$. It follows that $\mu^*(x) > 0$ (after scaling down the weights).[3]

We now prove μ^* has the balance property. If for all i, $S[c_i] > (\alpha_1 - \alpha_2/m)W$, then clearly there is no manipulation, since at least $\alpha_2 W$ points are given by the voters in T to the undesirable candidates $c_1, \ldots, c_m$. This happens with probability at least $\frac{1}{m^m}$.

On the other hand, consider the situation where for all i,

$$S[c_i] < (\alpha_1 - \frac{m^2-1}{m^2}\alpha_2)W; \quad (3)$$

this occurs with probability at least $\frac{1}{(m^2)^m}$. Intuitively, if the colluders could distribute the votes in T in such a way that each undesirable candidate is ranked last in exactly $1/m$-fraction of the votes, this would be a successful manipulation: each undesirable candidate would gain at most an additional $\frac{m-1}{m}\alpha_2 W$ points. Unfortunately, this is usually not the case, but the following condition is sufficient for a successful manipulation (assuming condition (3) holds). Partition the voters in T to m disjoint subsets $p_1, \ldots, p_i$ (w.l.o.g. of size n/m), and denote by W_{p_i} the total weight of the votes in p_i. The condition is that for all $i \in \{1,\ldots,m\}$:

$$(1 - 1/m) \cdot 1/2 \cdot n/m \leq W_{p_i} \leq (1 + 1/m) \cdot 1/2 \cdot n/m. \quad (4)$$

This condition is sufficient, because if the voters in p_i all rank c_i last, the fraction of the votes in T which gives c_i points is at most:

$$\frac{(m-1)(1+1/m)}{(m-1)(1+1/m)+1-1/m} = \frac{m^2-1}{m^2+m-2}.$$

Hence the number of points c_i gains from the colluders is at most:

$$\frac{m^2-1}{m^2+m-2}\alpha_2 \leq \frac{m^2-1}{m^2}\alpha_2 < \alpha_1 W - S[c_i].$$

Furthermore, by Lemma 1 and the fact that the expected total weight of n/m votes is $1/2 \cdot n/m$, the probability that condition (4) holds is at least $1 - 2e^{-\frac{2n}{m^3}}$. Since m is a constant, this probability is larger than $1/2$ for a large enough n.

[3]It seems the reduction can be generalized for a larger number of candidates. The hard instances are the ones where all undesirable candidates but two have approximately $(\alpha_1 - \alpha_2)W$ initial points, and two problematic candidates have approximately $(\alpha_1 - \alpha_m/2)W$ points. These instances have a positive probability under μ^*.

Algorithm 1 Decides SCWM

```
1: procedure GREEDY(S, w⃗, p)
2:     for all c ∈ C do                          ▷ Initialization
3:         S_0[c] ← S[c]
4:     end for
5:     for i = 1 to n do                         ▷ All voters in T
6:         Let j_1, j_2, ..., j_m s.t. ∀l, S_{i-1}[c_{j_{l-1}}] ≤ S_{i-1}[c_{j_l}]
7:         Voter t_i votes p ≻ c_{j_1} ≻ c_{j_2} ≻ ... ≻ c_{j_m}
8:         for l = 1 to m do                     ▷ Update score
9:             S_i[c_{j_l}] ← S_{i-1}[c_{j_l}] + w(t_i)α_{l+1}
10:        end for
11:        S_i[p] ← S_{i-1}[p] + w(t_i)α_1
12:    end for
13:    if argmax_{c∈C} S_n[c] = {p} then          ▷ p wins
14:        return true
15:    else
16:        return false
17:    end if
18: end procedure
```

Finally, it can easily be seen that μ^* has the refinement property: if all colluders rank p first and candidate c second, then p gets $\alpha_1 W$ points, and c gets $\alpha_2 W + S[c]$ points. But $S[c] \geq (\alpha_1 - \alpha_2)W$, and thus p surely loses. □

4.2 A Heuristic Polynomial Time Algorithm

We now present our algorithm GREEDY for SCWM, given as Algorithm 1. $\vec{w}$ denotes the vector of the weights of voters in $T = \{t_1, \ldots, t_n\}$.

The voters in T, according to some order, each rank p first, and the rest of the candidates by their current score: the candidate with the lowest current score is ranked highest. GREEDY accepts if and only if p wins this election.

This algorithm, designed specifically for scoring protocols, is a realization of an abstract greedy algorithm: at each stage, voter t_i ranks the undesirable candidates in an order that minimizes the highest score that any undesirable candidate obtains after the current vote. If there is a tie between several permutations, the voter chooses the option such that the second highest score is as low as possible, etc. In any case, every colluder always ranks p first.

REMARK 5. This abstract scheme might also be appropriate for protocols such as Maximin and Copeland. Similarly to scoring protocols, in these two protocols the colluders are always better off by ranking p first. In addition, the abstract greedy algorithm can be applied to Maximin and Copeland since the result of an election is based on the score each candidate has (unlike STV, for example).

In the following lemmas, a *stage* in the execution of the algorithm is an iteration of the for loop.

LEMMA 2. *If there exists a stage i_0 during the execution of* GREEDY*, and two candidates $a, b \neq p$, such that*

$$|S_{i_0}[a] - S_{i_0}[b]| \leq \alpha_2, \tag{5}$$

then for all $i \geq i_0$ it holds that $|S_i[a] - S_i[b]| \leq \alpha_2$.

PROOF. The proof is by induction on i. The base of the induction is given by equation (5). Assume that $|S_i[a] - S_i[b]| \leq \alpha_2$, and without loss of generality: $S_i[a] \geq S_i[b]$. By the algorithm, voter t_{i+1} ranks b higher than a, and therefore:

$$S_{i+1}[b] - S_{i+1}[a] \geq -\alpha_2. \tag{6}$$

Since p is always ranked first, and the weight of each vote is at most 1, b gains at most α_2 points. Therefore:

$$S_{i+1}[b] - S_{i+1}[a] \leq \alpha_2. \tag{7}$$

Combining equations (6) and (7) completes the proof. □

LEMMA 3. *Let $p \neq a, b \in C$, and suppose that there exists a stage i_0 such that $S_{i_0}[a] \geq S_{i_0}[b]$, and a stage $i_1 \geq i_0$ such that $S_{i_1}[b] \geq S_{i_1}[a]$. Then for all $i \geq i_1$ it holds that $|S_i[a] - S_i[b]| \leq \alpha_2$.*

PROOF. Assume that there exists a stage i_0 such that $S_{i_0}[a] \geq S_{i_0}[b]$, and a stage $i_1 \geq i_0$ such that $S_{i_1}[b] \geq S_{i_1}[a]$; w.l.o.g. $i_1 > i_0$ (otherwise at stage i_0 it holds that $S_{i_0}[b] = S_{i_0}[a]$, and then we finish by Lemma 2). Then there must be a stage i_2 such that $i_0 \leq i_2 < i_1$ and $S_{i_2}[a] \geq S_{i_2}[b]$ but $S_{i_2+1}[b] \geq S_{i_2+1}[a]$. Since the weight of each vote is at most 1, b gains at most α_2 points by voter t_{i_2+1}. Hence the conditions of Lemma 2 hold for stage i_2, which implies that for all $i \geq i_2$: $|S_i[a] - S_i[b]| \leq \alpha_2$. In particular, $i_1 \geq i_2$. □

LEMMA 4. *Let P be a sensitive scoring protocol, and assume* GREEDY *errs on an instance of SCWM in P which has a successful manipulation. Then there is $d \in \{2, 3, \ldots, m\}$, and a subset of candidates $D = \{c_{j_1}, \ldots, c_{j_d}\}$, such that:*

$$\begin{aligned} &\sum_{i=1}^{d}(\alpha_1 W - S[c_{j_i}]) - \sum_{i=1}^{d-1}(i \cdot \alpha_2) \leq W \sum_{i=1}^{d} \alpha_{m+2-i} \\ &\leq \sum_{i=1}^{d}(\alpha_1 W - S[c_{j_i}]). \end{aligned} \tag{8}$$

PROOF. For the right inequality, for any d candidates, even if all voters in T rank them last in every vote, the total points distributed among them is $W \sum_{i=1}^{d} \alpha_{m+2-i}$. If this inequality does not hold, there must be some candidate c_i that gains at least $\alpha_1 W - S[c_i]$ points from the colluders, implying that this candidate has at least $\alpha_1 W$ points. However, p also has at most $\alpha_1 W$ points, and we assumed that there is a successful manipulation — a contradiction.

For the left inequality, assume the algorithm erred. Then at some stage i_0, there is a candidate c_{j_0} who has a total of at least $\alpha_1 W$ points (w.l.o.g. only one candidate passes this threshold simultaneously). Denote $T_0 = \{t_1, t_2, \ldots, t_{i_0}\}$, and let W_{T_0} be the total weight of the voters in T_0. Voter t_{i_0} did not rank c_{j_0} last, since $\alpha_{m+1} = 0$, and thus ranking a candidate last gives it no points. We have that there is another candidate c_{j_1}, such that: $S_{i_0-1}[c_{j_1}] \geq S_{i_0-1}[c_{j_0}]$. By Lemma 3, $S_{i_0}[c_{j_0}] - S_{i_0}[c_{j_1}] \leq \alpha_2$, and thus $S_{i_0}[c_{j_1}] \geq \alpha_1 W - \alpha_2$. If these candidates were not always ranked last by the voters of T_0, there must be another candidate c_{j_2} who was ranked strictly higher by some voter in T_0, w.l.o.g. higher than c_{j_1}. Therefore, we have from Lemma 3 that: $S_{i_0}[c_{j_1}] - S_{i_0}[c_{j_2}] \leq \alpha_2$, and so c_{j_2} has a total of at least $\alpha_1 W - 2\alpha_2$ points. By inductively continuing this reasoning, we obtain a subset D of d candidates (possibly $d = m$), who were always ranked in the d last places by the voters in T_0, and for the l'th candidate it holds that: $S_{i_0}[c_{j_l}] \geq \alpha_1 W - (l-1)\alpha_2$. The total points gained by this l'th candidate until stage i_0 must be at least $\alpha_1 W - (l-1)\alpha_2 - S[c_{j_l}]$.

Since the total points distributed by the voters in T_0 to the d last candidates is $W_{T_0}\sum_{i=1}^{d}\alpha_{m+2-i}$, we have:

$$\sum_{i=1}^{d}(\alpha_1 W - S[c_{j_i}]) - \sum_{i=1}^{d-1}(i\cdot\alpha_2) \le W_{T_0}\sum_{i=1}^{d}\alpha_{m+2-i} \le W\sum_{i=1}^{d}\alpha_{m+2-i}.$$

□

LEMMA 5. *Let M be SCWM in a sensitive scoring protocol P with $C=\{p,c_1,\ldots,c_m\}$, m=O(1). Then* GREEDY *is a deterministic heuristic polynomial time algorithm for $\langle M,\mu^*\rangle$.*

PROOF. It is obvious that if the given instance has no successful manipulation, then the greedy algorithm would indeed answer that there is no manipulation, since the algorithm is constructive (it actually selects specific votes for the colluders).

We wish to bound the probability that there is a manipulation and the algorithm erred. By Lemma 4, a necessary condition for this to occur is as specified in equation (8), or equivalently:

$$W\sum_{i=1}^{d}\alpha_1 - W\sum_{i=1}^{d}\alpha_{m+2-i} - \frac{d(d-1)}{2}\alpha_2 \le \sum_{i=1}^{d}S[c_{j_i}] \le W\sum_{i=1}^{d}\alpha_1 - W\sum_{i=1}^{d}\alpha_{m+2-i}. \quad (9)$$

In this case the algorithm may err; but what is the probability of equation (9) holding? Fix a subset D of size $d\in\{2,\ldots,m\}$. $\sum_{i=1}^{d}S[c_{j_i}]$ is a random variable that takes values in $[d(\alpha_1-\alpha_2)W, d\alpha_1 W]$. By fixing values for the $S[c_{j_i}]$, we have that the probability of $\sum_{i=1}^{d}S[c_{j_i}]$ taking values in some interval $[a,b]$ is at most the chance of $S[c_{j_d}]$ taking a value in an interval of size $b-a$, which is at most $\frac{b-a}{\alpha_1 W-(\alpha_1-\alpha_2)W}$, since $S[c_{j_d}]$ is uniformly distributed. By Lemma 1, $W<n/4$ with probability at most $\epsilon(n)=e^{-\frac{n}{8}}$. On the other hand, if $W\ge n/4$, then (9) holds for D with probability at most

$$\frac{\frac{d(d-1)}{2}\alpha_2}{\alpha_1 W-(\alpha_1-\alpha_2)W} = \frac{d(d-1)}{2W} \le \frac{2d(d-1)}{n} = \frac{1}{p^D(n)},$$

for some polynomial p^D. We complete the proof by showing that equation (1) holds:

$$\begin{aligned}&\Pr_{x\sim\mu_n^*}[\text{GREEDY}(x)\neq M(x)]\\ &\le \Pr[W\ge n/4\wedge(\exists D\subset C\ s.t.\ |D|\ge 2\wedge(9)\text{ holds})] + \Pr[W<n/4]\\ &\le \sum_{D\subset C:|D|\ge 2}\frac{1}{p^D(n)}+\epsilon(n)\\ &\le \frac{1}{\text{poly}\,n}\end{aligned}$$

The last inequality follows from the assumption that $m=O(1)$. □

Clearly, Theorem 1 directly follows.

Algorithm 2 Decides UVWM

```
1: procedure SAMPLE-AND-MANIPULATE(w⃗, ν, p, r)
2:     for all permutations of the m+1 candidates do
3:         π ← next permutation
4:         ν* ← the manipulator votes π
5:             ▷ others' votes are always distributed w.r.t. ν
6:         if SAMPLE(w⃗, ν*, p, r) then
7:             return true
8:         end if
9:     end for
10:    return false
11: end procedure
```

5. SUSCEPTIBILITY TO UVWM

In this section we shall prove:

THEOREM 5. *Let P be a voting protocol such that there exists a junta distribution μ^P over the instances of UVWM in P, with the following property: r is uniformly distributed in $[0,1]$. Then P, with candidates $C=\{p,c_1,\ldots,c_m\}$, $m=O(1)$, is susceptible to UVWM.*

The existence of a junta distribution with r uniformly distributed is a very weak requirement (it is even quite natural to have r uniformly distributed). In fact, the following claim is very likely to be true:

CONJECTURE 6. *Let P be a voting protocol. Then there exists a junta distribution μ^P over the instances of UVWM in P, with r uniformly distributed in $[0,1]$.*

If this conjecture is indeed true, we have that all voting protocols are susceptible to UVWM. If for some reason the conjecture is not true with respect to our definition of junta distributions, then perhaps the definition is too restrictive and should be modified accordingly.

To prove Theorem 5, we require a procedure named SAMPLE, which decides UVWE. SAMPLE samples the given distribution on the votes n^3 times, and calculates the winner of the election each time. If p won more than an r-fraction of the elections then the procedure accepts, otherwise it rejects. We omit the details of the procedure, as well as the proof of the following lemma, due to lack of space.

LEMMA 6. *Let P be a voting protocol, and E be UVWE in P with $C=\{p,c_1,\ldots,c_m\}$. Furthermore, let μ be a distribution over the instances of E, with r uniformly distributed in $[0,1]$. Then there exists N such that for all $n\ge N$:*

$$\Pr_{x\sim\mu_n}[\text{SAMPLE}(x)\neq E(x)] \le \frac{1}{\text{poly}n}.$$

We now present an algorithm, SAMPLE-AND-MANIPULATE that decides UVWM; it is given as Algorithm 2. Here, $\vec{w}$ denotes the weights of all voters including the manipulator, and ν is the given distribution over the others' votes.

Given an instance of UVWM, SAMPLE-AND-MANIPULATE generates $(m+1)!$ instances of the UVWE problem, one for each of the manipulator's possible votes, and executes SAMPLE on each instance. SAMPLE-AND-MANIPULATE accepts if and only if SAMPLE accepts one of the instances.

LEMMA 7. *Let P be a voting protocol, and M be UVWM in P with $C=\{p,c_1,\ldots,c_m\}$, $m=O(1)$. Furthermore,*

let μ be a distribution over the instances of UVWM, with r uniformly distributed in $[0,1]$. It holds that SAMPLE-AND-MANIPULATE *is a probabilistic heuristic polynomial time algorithm for $\langle M, \mu \rangle$.*

PROOF. For each independent call to SAMPLE, the chance of failure is inverse-polynomial. By applying the union bound we have that the probability of SAMPLE failing on any of the $(m+1)!$ invocations is at most $(m+1)!\frac{1}{\text{poly}n}$, which is still inverse-polynomial since m is constant. The lemma now follows from the fact that there is a manipulation if and only if there is a permutation of candidates, such that if the manipulator votes according to this permutation, the chance of p winning is greater than r.

Notice that SAMPLE-AND-MANIPULATE is indeed polynomial by the fact that $m = O(1)$, and we assumed that the given distribution over the votes can be sampled in polynomial time. □

6. FUTURE RESEARCH

The issue of resistance of mechanisms to manipulation is important, particularly in the context of voting protocols. Most results on this issue use $\mathcal{NP}$-hardness as the complexity measure. One of this paper's main contributions has been introducing tools that can be utilized in showing that manipulating mechanisms is *easy* in the average case. We were concerned with the likely case of coalitional manipulation, and showed that sensitive scoring protocols are susceptible to such manipulation when the number of candidates is constant.

These results suggest that scoring protocols cannot be safely employed. More importantly, this paper should be seen as a starting point for studying the average case complexity of other types of manipulations, in other protocols. In addition, the definitions in Section 3 are deliberately general, and can be applied to manipulations of mechanisms that are not voting mechanisms. One such mechanism of which we are aware, whose manipulation is $\mathcal{NP}$-hard, is presented in [1, 2].

There is still room for debate as to the exact definition of a junta distribution, especially if Conjecture 6 turns out to be false. It may also be the case that there are "unconvincing" distributions that satisfy all of the (current) conditions of a junta distribution. It might prove especially fruitful to show that a heuristic polynomial time algorithm with respect to a junta distribution also has the same property with respect to some easy distributions, such as the uniform distribution.

An issue of great importance is coming up with natural criteria to decide when a manipulation problem is *hard* in the average-case. The traditional definition of average-case completeness is very difficult to work with in general; is there a satisfying definition that applies specifically to the case of manipulations? Once the subject is more fully understood, this understanding can be used to design mechanisms that are hard to manipulate in the average-case.

7. ACKNOWLEDGMENT

This work was partially supported by grant #039-7582 from the Israel Science Foundation.

8. REFERENCES

[1] Y. Bachrach and J. S. Rosenschein. Achieving allocatively-efficient and strongly budget-balanced mechanisms in the network flow domain for bounded-rational agents. In *The Nineteenth International Joint Conference on Artificial Intelligence*, pages 1653–1654, Edinburgh, Scotland, August 2005.

[2] Y. Bachrach and J. S. Rosenschein. Achieving allocatively-efficient and strongly budget-balanced mechanisms in the network flow domain for bounded-rational agents. The Seventh International Workshop on Agent-Mediated Electronic Commerce (AMEC 2005), Utrecht, The Netherlands, July 2005.

[3] J. Bartholdi and J. Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

[4] V. Conitzer, J. Lang, and T. Sandholm. How many candidates are needed to make elections hard to manipulate? In *Proceedings of the International Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 201–214, Bloomington, Indiana, 2003.

[5] V. Conitzer and T. Sandholm. Complexity of manipulating elections with few candidates. In *Proceedings of the National Conference on Artificial Intelligence*, pages 314–319, Edmonton, Canada, July 2002.

[6] V. Conitzer and T. Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 781–788, Acapulco, Mexico, August 2003.

[7] E. Elkind and H. Lipmaa. Small coalitions cannot manipulate voting. In *International Conference on Financial Cryptography*, Lecture Notes in Computer Science. Springer-Verlag, Roseau, The Commonwealth of Dominica, 2005.

[8] A. Gibbard. Manipulation of voting schemes. *Econometrica*, 41:587–602, 1973.

[9] E. Hemaspaandra and L. A. Hemaspaandra. Dichotomy for voting systems. University of Rochester Department of Computer Science Technical Report 861, 2005.

[10] M. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.

[11] L. Trevisan. Lecture notes on computational complexity. Available from http://www.cs.berkeley.edu/~luca/notes/complexitynotes02.pdf, 2002. Lecture 12.

The Communication Complexity of Coalition Formation among Autonomous Agents

Ariel D. Procaccia Jeffrey S. Rosenschein
School of Engineering and Computer Science
The Hebrew University of Jerusalem
Jerusalem, Israel
{arielpro, jeff}@cs.huji.ac.il

ABSTRACT

It is self-evident that in numerous Multiagent settings, selfish agents stand to benefit from cooperating by forming coalitions. Nevertheless, negotiating a *stable* distribution of the payoff among agents may prove challenging. The issue of coalition formation has been investigated extensively in the field of cooperative n-person game theory, but until recently little attention has been given to the complications that arise when the players are software agents. The bounded rationality of such agents has motivated researchers to study the computational complexity of the aforementioned problems.

In this paper, we examine the *communication* complexity of coalition formation, in an environment where each of the n agents knows only its own initial resources and utility function. Specifically, we give a tight $\Theta(n)$ bound on the communication complexity of the following solution concepts in unrestricted games: Shapley value, the nucleolus and the modified nucleolus, equal excess theory, and the core. Moreover, we show that in some intuitively appealing restricted games the communication complexity is constant, suggesting that it is possible to achieve sublinear complexity by constraining the environment or choosing a suitable solution concept.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity;
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Theory

Keywords

Communication complexity, Coalition formation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

In an environment teeming with autonomous agents, it is only natural for agents to cooperate in order to achieve goals, even under the assumption that the agents are selfish. Cooperation is realized through the formation of *coalitions*: the members of each coalition share their resources, and ultimately divide their payoff. In the second half of the 20th century, some of the most prominent researchers in game theory have studied cooperative games; the focus of most research was determining which coalitions would form in a given game, and how coalitions should divide their payoff among their members.

The trick is to divide the payoff in a way that keeps the coalition structure stable: the agents should be motivated to remain loyal to their respective coalitions, instead of deviating and forming new coalitions that might guarantee them a higher expected payoff. Different notions of stable solutions have been proposed. The strongest, named the *core* of the game, is sometimes empty. Other (weaker) solutions have different desirable properties.

In the past few years, a new layer has been added to the problem. The "players" in our cooperative game are software agents, which are limited in various (practical) ways. This complication has motivated researchers to study the computational complexity of different solution concepts for cooperative games [7, 1, 2, 4].

Another issue, that has so far received little attention in multiagent systems research, is the *communication complexity* of cooperative games in multiagent environments. In the multiparty communication complexity model, each of the players (agents[1]) holds some part of the input, and the players wish to jointly compute some function on the input. In this model, we assume the players have unlimited computational power; we are only interested in the worst-case number of bits of information they must pass among themselves. Although the study of communication complexity is not natural in the context of some problems, it seems especially appropriate in the context of multiagent systems in general[2] (see for example [3]), and cooperative games in particular.

Shehory and Kraus [8] analyzed the computational and communication complexity of two algorithms for payoff di-

[1]Throughout the paper, we use the terms "players" and "agents" interchangeably.

[2]As mentioned above, computational complexity issues are also important (perhaps even more so) in multiagent systems. These issues are not taken into account in the communication complexity model, but are studied separately.

vision in a multiagent environment. The environment introduced by Shehory and Kraus induces a partitioning of the information about the game among the agents — a fact that makes this environment an obvious candidate for a study of communication complexity.

In this paper, we analyze the communication complexity of computing the payoff, in different solution concepts, of an arbitrary player. This restricted problem (as opposed to computing the payoffs for all the players) is important in its own right: it is reasonable that an agent would like to know *a priori* its expected payoff from a game, in order to decide whether to participate in the game at all.[3] It seems possible that, for some solution concepts or specific types of games, it may be sufficient for an agent to elicit information only from a small subset of the other agents in order to compute its payoff.

We focus on two categories of solution concepts:

- Singleton Solutions: Shapley value, the nucleolus (and modified nucleolus), and equal excess theory.
- Solutions that are possibly empty: the core.

For such solution concepts, one can establish hardness of suitable decision questions, such as determining whether the payoff of a certain agent is greater than 0, or determining whether the solution set is empty. Deciding these problems is clearly easier than computing an agent's payoff in a solution (*the* solution in the case of singleton solutions). Studying division schemes that do not fall into one of the two categories is less straightforward, since there are no apparent decision problems that are easier than computing the payoff in a solution. For example, one could ask whether a given payoff configuration is a solution, but even if this problem is hard, it may still be easy to generate just one solution (out of a possibly large solution set). The insistence on having such decision problems will become apparent in Section 2.

We investigate an environment in which there are n agents, each holding a constant amount of information. We show a tight bound of $\Theta(n)$ on the communication complexity of computing the payoff for an arbitrary player under all solution concepts mentioned above. Furthermore, we show that in restricted (but common) types of cooperative games, the communication complexity of finding the payoff of a player in a solution is 1.

The rest of the paper is organized as follows. In Section 2 we explain the basics of cooperative games and communication complexity. In Section 3 we prove our results, and discuss the communication complexity of restricted games. In Section 4 we propose directions for future research.

2. PRELIMINARIES

2.1 Cooperative Games

We start by explaining the basic ideas of cooperative n-person games in characteristic form, and defining restricted types of games that will be needed later on. We next specify the different solution concepts. Last, we describe our environment and connect this representation with games in characteristic form.

[3]This becomes even more explicit when the agent has to pay so as to join the game.

In the next few paragraphs, we follow Chapter 2 of [5].

A *cooperative n-person game in characteristic form with side payments* is a pair $(N; v)$, where $N = \{1, 2, \ldots, n\}$ is a set of players, and v is the *characteristic function*, which assigns a real number $v(S)$ to each $S \subseteq N$. $v(S)$ is the *value* of S: the payoff the players in S can obtain by cooperating. It always holds that $v(\emptyset) = 0$. The collection of payoffs to the players is expressed as a payoff vector: $\vec{x} = \{x_1, x_2, \ldots, x_n\}$.

A *coalition structure* is a partition of N, of the form $\phi = \{S_1, S_2, \ldots, S_r\}$, which specifies how the players in N divide themselves into coalitions. A *Payoff Configuration* is a pair

$$(\vec{x}; \phi) = \{x_1, x_2, \ldots, x_n; S_1, S_2, \ldots, S_r\},$$

where $\vec{x}$ is a payoff vector and ϕ is a coalition structure, such that:

$$\forall j \in [r] : \; x(S_j) =_{\text{def}} \sum_{i \in S_j} x_i = v(S_j).$$

We shall refer to the coalition of all players as the *grand coalition*.

EXAMPLE 1. Consider the following 3-person game known as "Odd Man Out":

$$v(\{1\}) = v(\{2\}) = v(\{3\}) = v(N) = 0;$$
$$v(\{1,2\}) = 4;\; v(\{1,3\}) = 5;\; v(\{2,3\}) = 6.$$

The payoff configuration $(2, 2, 0; \{1, 2\}, \{3\})$ would be obtained if players 1 and 2 formed a coalition and split their payoff equally.

A game is *superadditive* if:

$$\forall S, T \subseteq N \; s.t. \; S \cap T = \emptyset : \; v(S \cup T) \geq v(S) + v(T).$$

Superadditivity is a reasonable assumption in many games, since the union $S \cup T$ of two coalitions may, in the worst case, act as two separate coalitions and receive the payoff $v(S) + v(T)$.

We next discuss special types of games. In *weighted majority* games, the players are assigned weights $w_1, w_2, \ldots, w_n$, and a criterion number q is specified, such that:

$$\forall S \subseteq N : \; v(S) = \begin{cases} 1 & \sum_{i \in S} w_i \geq q \\ 0 & \sum_{i \in S} w_i < q \end{cases}$$

Such a game is represented by the following shorthand notation:

$$[q; w_1, w_2, \ldots, w_n].$$

LEMMA 1. *Let $[q; w_1, w_2, \ldots, w_n]$ be a weighted majority game, where for all i, $w_i \in \{0, 1\}$. If $q \geq \lfloor n/2 \rfloor + 1$, then the game is superadditive.*

PROOF. Let $S, T \subseteq N$ such that $S \cap T = \emptyset$. If $|S| < q$ and $|T| < q$, then clearly $v(S \cup T) \geq v(S) + v(T)$. Otherwise, assume without loss of generality that $|S| \geq q$. Since $q \geq \lfloor n/2 \rfloor + 1$ and $S \cap T = \emptyset$, it holds that $|T| < q$; it follows that $v(T) = 0$. Therefore,

$$v(S \cup T) = 1 = 1 + 0 \geq v(S) + v(T).$$

□

A different type of restricted game is an *apex* game. In such games, the only coalitions that have nonzero value are those that contain a distinguished player called the *apex*

(and are of size at least 2), or the remaining $n-1$ players (called the *base*) are included. Constant-sum apex games have the weighted majority representation:

$$[n-1; 1, \ldots, 1, n-2, 1, \ldots, 1],$$

where $n-2$ is the weight of the apex player.

Similarly to apex games, in *veto* games a distinguished player is a necessary (but not sufficient) member of any coalition with nonzero value. The general homogeneous weighted majority representation of the veto game in which the veto player needs only one ally is:

$$[n; 1, \ldots, 1, n-1, 1, \ldots, 1],$$

where $n-1$ is the weight of the veto player.

2.1.1 Solution Concepts

In this part of the paper, we follow Chapters 3 and 6 of [5], as well as [9].

Over the years, many different solutions to characteristic function games with side payments have been proposed. These solutions differ in their notion of stability: given a coalition structure, the payoff division should be such that agents are not motivated to deviate, thus breaking down coalitions. For example, if $v(\{i\}) > 0$, any payoff configuration where $x_i = 0$ could not be stable, since player i would prefer to receive the payoff he could get by himself.

2.1.1.1 The Core.

The *core* C of a game $(N; v)$ is the following set of payoff configurations:

$$C = \{(\vec{x}, \phi) : \forall S \subseteq N, x(S) \geq v(S)\}.$$

Less formally, the core is the set of payoff configurations that leave no coalition in a position to improve the payoffs to all of its members.

The core is the strongest of all solution concepts; in fact, it is so strong that in some cases the core is empty. In such a case, at least one coalition will be dissatisfied with any payoff configuration.

2.1.1.2 Shapley Value.

The *Shapley value* is a payoff division scheme that is characterized axiomatically, and hence satisfies some important desiderata. Player i is called a *dummy* if $v(S \cup \{i\}) = v(S)$ for all coalitions S that do not include i; players i and j are *interchangeable* if $v((S - \{i\}) \cup \{j\}) = v(S)$ for every coalition S that includes i but not j. The axioms of the Shapley value are:

- *Symmetry:* If i and j are interchangeable, then $x_i = x_j$.
- *Dummies:* If i is a dummy, then $x_i = v(\{i\})$.
- *Additivity:* For any two games $(N; v)$ and $(N; w)$, x_i in $(N; v + w)$ equals x_i in $(N; v)$ plus x_i in $(N; w)$.

It is well known that the Shapley value, defined by:

$$x_i = \sum_{S \subseteq N} \frac{(|N| - |S|)!(|S| - 1)!}{|N|!}(v_S - v_{S-\{i\}}),$$

is the *unique* payoff division scheme that satisfies the axioms listed above.

2.1.1.3 The Nucleolus and the Modified Nucleolus.

The *excess* of a coalition S with respect to the payoff vector $\vec{x}$ is: $e(S, \vec{x}) = v(S) - x(S)$. Given a payoff configuration $(\vec{x}; \phi)$, an excess $e(S, \vec{x})$ can be constructed for any coalition S; there are 2^n such coalitions. Let $\theta(\vec{x})$ be a vector of length 2^n, whose components are all possible excesses, sorted in non-increasing order:

$$\begin{aligned}\theta(\vec{x}) &= \langle \theta_1(\vec{x}), \theta_2(\vec{x}), \ldots, \theta_{2^n}(\vec{x}) \rangle \\ &= \langle e(S_1, \vec{x}), e(S_2, \vec{x}), \ldots, e(S_{2^n}, \vec{x}) \rangle,\end{aligned}$$

where for all $i < j$: $e(S_i, \vec{x}) \geq e(S_j, \vec{x})$. $\theta(\vec{x})$ is said to be *lexicographically greater* than $\theta(\vec{y})$, denoted $\theta(\vec{x}) \succ \theta(\vec{y})$, if there exists an integer $q \in [2^n]$ such that $\theta_p(\vec{x}) = \theta_p(\vec{y})$ for all $p < q$, and $\theta_q(x) > \theta_q(y)$. If $\theta(\vec{x})$ is not lexicographically greater than $\theta(\vec{y})$, we write $\theta(\vec{x}) \not\succ \theta(\vec{y})$.

Example 2. Consider the following 3-person game:

$$v(\{1\}) = v(\{2\}) = v(\{3\}) = 0;\ v(N) = 105$$
$$v(\{1,2\}) = 90;\ v(\{1,3\}) = 80;\ v(\{2,3\}) = 70.$$

Define a payoff configuration $(\vec{x}, \phi) = (15, 55, 35; N)$. The 2^3 excesses with respect to $\vec{x}$ are

$$(0, 0, -15, -55, -35, 20, 30, -20)$$

for coalitions $(\emptyset, N, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\})$, respectively. Therefore,

$$\vec{\theta}(\vec{x}) = (30, 20, 0, 0, -15, -20, -35, -55).$$

Similarly, for $(\vec{y}; \phi) = (45, 15, 45; N)$ we have:

$$\vec{\theta}(\vec{y}) = (30, 10, 0, 0, -10, -15, -45, -45).$$

It follows that $\vec{\theta}(\vec{x}) \succ \vec{\theta}(\vec{y})$.

The *nucleolus* of a game is the set of all payoff configurations for which the sorted vector of excesses is lexicographically minimal:

$$\{(\vec{x}; \phi) : \theta(\vec{x}) \not\succ \theta(\vec{y}) \text{ for all } \vec{y}, \text{ given } \phi\}.$$

The *modified nucleolus* is defined identically, except that instead of a vector $\vec{\theta}$ with 2^n excesses, a sorted vector with $2^n - n - 2$ excesses is constructed, with components $e(S, \vec{x})$ for all coalitions S such that $1 < |S| < n$.

The nucleolus is unique for each coalition structure in any characteristic function game. It is also known that the modified nucleolus for the grand coalition consists of a unique payoff vector.

2.1.1.4 Equal Excess.

Equal excess theory yields a payoff vector that is the result of a bargaining process. At each stage, a player lodges a claim for a share of the value of each coalition of which he is a member. As a starting point, each player expects an equal share of the value of each such coalition.

More formally, the bargaining process consists of discrete *rounds*. In round r, player i has an *expectation* of the payoff he will obtain from coalition S of which he is a member; this expectation is denoted $E^r(i, S)$. Let $A^r(i, S) = \max_{T \neq S}[E^r(i, T)]$; this is player i's highest expectation from the alternative coalitions to S. Player i's expectation for round $r+1$ is created by:

$$E^{r+1}(i, S) = A^r(i, S) + \frac{v(S) - \sum_{j \in S} A^r(j, S)}{|S|}.$$

EXAMPLE 3. Consider the following 3-person game:

$$v(\{1\}) = v(\{2\}) = v(\{3\}) = v(N) = 0;$$
$$v(\{1,2\}) = 90;\ v(\{1,3\}) = 80;\ v(\{2,3\}) = 70.$$

In round 0, we have: $E^0(1,\{1,2\}) = E^0(2,\{1,2\}) = 45$, $E^0(1,\{1,3\}) = E^0(3,\{1,3\}) = 40$, and finally $E^0(2,\{2,3\}) = E^0(3,\{2,3\}) = 35$. If the game terminates at round 0 and the coalition structure is $\{\{1,2\},\{3\}\}$, for instance, then the payoffs would be $(45, 45, 0)$. If negotiation continues to round 1, then for example: $A^0(1,\{1,2\}) = 40$, $A^0(2,\{1,2\}) = 35$, and thus:

$$E^1(1,\{1,2\}) = 40 + \frac{90 - 40 - 35}{2} = 47.5.$$

For the same coalition structure as before, player 1's payoff after a round of negotiation would be 47.5.

If the bargaining process ends after m rounds, we say it is the solution generated by *m-round* equal excess. As far as we know, for all games examined, the sequence of expectations converges to an asymptotic value, but there is no proof of convergence. Assuming such a limit always exists, we say it is the solution generated by ∞-*round* equal excess.

2.1.2 Environment Description

We follow the presentation of the environment in [8] (with some simplifications). An exact description of the environment is important for our purposes, since it induces a distribution of the information about the game (or input) among the agents.

Our environment consists of autonomous agents $1, 2, \ldots, n$, with tasks to fulfill. Each agent has a given amount of the resources $l_1, \ldots, l_s$, which are required in order to deal with tasks (we denote by Q the set of all possible vectors of quantities of resources). An agent receives a payoff for fulfilling tasks. These concepts can be formalized as a *payoff function* $U^i : Q \to \mathbb{R}^+$, which gives the payoff of agent i for some arbitrary resources.

We assume that resources can be traded among agents. We also assume that payoff can be transferred from one agent to another (using money, or some other divisible desirable commodity).[4] The agents may try to tackle the tasks alone, but may also prefer to form coalitions, thus pooling (and redistributing) their resources. If S is a coalition, we say that its value is $v(S)$ if $v(S) = \sum_{i \in S} U^i(\overline{q}_i)$, where $\overline{q}$ is the vector of resources after optimal redistribution in the coalition.

It is important to recognize that such an environment can be represented using a characteristic function game, and sometimes vice versa; hence the concepts from the previous subsections can be used. However, the representation of the game proposed in this subsection is more realistic. In essence, the information about the game is distributed among the agents; each agent holds a *constant* amount of information, namely its resources and payoff function.

EXAMPLE 4. Consider the weighted majority game defined by $[q; w_1, w_2, \ldots, w_n]$. There is only one resource, of which each agent is given the initial quantity w_i; the utility function for all agents is:

$$U^i(z) = \begin{cases} 1 & z \geq q \\ 0 & z < q \end{cases}$$

In such games, the private information agent i has — its resources and production function — essentially defines its weight w_i (which can be deduced from its resources), and the value of q (which can be deduced from the common utility function, and is therefore common knowledge).

We do not assume that the environment is superadditive (although this is implied by the above definitions), and our results also apply to environments that do not have this property. However, note that in a superadditive game, we can assume that the grand coalition forms at some stage, and concern ourselves only with the payoff division (and not the coalition structure).

2.2 Communication Complexity

In this subsection, we present the multiparty communication complexity model with which we shall deal. A recommended, more detailed overview of communication complexity theory appears in [6].

There are several ways to generalize the two-party communication complexity model (introduced by Yao in [10]) to a multiparty setting. In our model,[5] player i ($i \in N$) holds an input $z_i \in \{0,1\}^k$. The players wish to compute together a function $f : (\{0,1\}^k)^n \to A$. They communicate via a "public blackboard" — all the players can see any bit a player sends. We are only interested in the amount of communication among the players, and therefore we allow the players to have unlimited computational power.

A *deterministic protocol* P is a binary tree where each internal node v is labeled by a function $g_{i,v}\{0,1\}^k \to \{0,1\}$, for a single $i \in N$, and each leaf is labeled with $a \in A$. The *value* of the protocol P on input $(z_1, z_2, \ldots, z_n)$ is the label of the leaf reached by starting from the root, and walking on the tree: at each internal node v labeled by $g_{i,v}$ walking left if $g_{i,v}(z_i) = 0$, and right if $g_{i,v}(z_i) = 1$. The *cost* of the Protocol P on input $(z_1, z_2, \ldots, z_n)$ is the length of the path taken on this input. The *cost* of the protocol P is the height of the tree. The *deterministic communication complexity* of a function f is the minimum cost of P, over all protocols P that compute f.

Intuitively, every internal node v labeled by $g_{i,v}$ is associated with a bit sent by player i: 0 if $g_{i,v}(z_i) = 0$, and 1 otherwise. At each point in the protocol, the current node in the tree is determined by the previous bits sent by the players. In this way, players take into account the bits communicated so far.

It is also possible to consider the *nondeterministic communication complexity* of f. An all-powerful prover is trying to convince the players that $f(z_1, z_2, \ldots, z_n) = a_0$. If it is indeed true that $f(z_1, z_2, \ldots, z_n) = a_0$, then the prover should be able to convince the players (by posting the answer and a proof to the blackboard). However, if this is not the case, the players should be able to detect the lie, regardless of what the prover says. It is obvious that a lower bound on the

[4]Such a transfer of payoff is called a *side payment*. Recall that we assumed side payments are possible when we described characteristic function games in Section 2.1.

[5]Another interesting model is the "number on the forehead" model, where each player holds a bit that all other players can see, but he cannot.

nondeterministic communication complexity of a problem is also a lower bound on the deterministic communication complexity of the problem. Furthermore, it is also a lower bound on the *randomized* communication complexity of the problem.

REMARK 1. Many problems in the multiparty model have a nondeterministic communication complexity of $o(n)$. For example, each player holds one bit, and

$$f = \max(z_1, z_2, \ldots, z_n).$$

If $f(z_1, z_2, \ldots, z_n) = 1$, the prover can post to the blackboard an index i of the player with $z_i = 1$ ($\log n$ bits); this player checks whether indeed $z_i = 1$.

There are several techniques for obtaining lower bounds on communication complexity. The most popular technique is the *fooling set*.

DEFINITION 1. Let $f : (\{0,1\}^k)^n \to \{0,1\}$. A subset H of $(\{0,1\}^k)^n$ is called a *fooling set* (for f) if there exists a value $f_0 \in \{0,1\}$ such that:

- For every $(z_1, z_2, \ldots, z_n) \in H$, $f(z_1, z_2, \ldots, z_n) = f_0$.
- For every two distinct vectors

$$(z_1^1, z_2^1, \ldots, z_n^1), (z_1^2, z_2^2, \ldots, z_n^2) \in H,$$

there exist $r_1, r_2, \ldots, r_n \in \{1,2\}$ such that

$$f(z_1^{r_1}, z_2^{r_2}, \ldots, z_n^{r_n}) = 1 - f_0.$$

Less formally, by "mixing" the coordinates of any two vectors in the fooling set, we can obtain an input vector whose value under f is $1 - f_0$.

It is known that the existence of a fooling set of size m for f entails a lower bound of $\log m$ on the nondeterministic communication complexity of f [6].

REMARK 2. In the context of cooperative games, f is the function which, given the resources and utility function for all the players and a distinguished player, outputs that player's payoff in some solution, according to a fixed solution concept. However, when proving our lower bounds, we will deal with boolean functions[6] (so that we can use the fooling set technique).

3. RESULTS

This section is devoted to proving a tight bound of $\Theta(n)$ on the communication complexity of computing a player's expected payoff in the following solution concepts (in this order): Shapley value, the nucleolus and modified nucleolus, m-round equal excess and ∞-round equal excess, and the core.

3.1 Upper Bound

Obtaining an upper bound on the communication complexity of any solution concept in our environment is immediate:

[6]Determining whether a given player's payoff is greater than 0, or determining whether the solution set is nonempty. This strengthens our results, since these decision problems are weaker than computing payoff.

PROPOSITION 1. *The deterministic communication complexity of any solution concept (even when computing the payoff of all players) is $O(n)$.*

PROOF. Recall that each agent holds a constant amount of information. Therefore, all n agents can communicate their entire part of the input (resources and utility function), and then compute a solution. □

3.2 Lower Bounds

The following lemma will soon be essential in the proof of Lemma 3.

LEMMA 2. $log\binom{n}{\lfloor \frac{n}{2} \rfloor + 1} = \Omega(n)$.

PROOF. Without loss of generality, assume n is even (for an odd n small changes are required, but the proof is similar). Observe that

$$\binom{n}{\frac{n}{2}+1} = \frac{n!}{(\frac{n}{2}+1)!(\frac{n}{2}-1)!} = \frac{n \cdot (n-1) \cdots (n/2+2)}{(n/2-1) \cdot (n/2-2) \cdots 1}.$$

We associate each factor in the denominator which is greater than or equal to $n/4+1$, with a factor in the numerator that is exactly twice as large. For example, $n/4+1$ is associated with $n/2+2$. We have $n/4-1$ such pairs, each with a ratio of 2. The other $n/4$ factors in the numerator are all greater than the other $n/4$ factors in the denominator. Therefore, we have: $\binom{n}{\frac{n}{2}+1} \geq 2^{n/4-1}$, and hence:

$$\log\binom{n}{\frac{n}{2}+1} \geq \log\left(2^{n/4-1}\right) = n/4 - 1 = \Omega(n).$$

□

The next definition and lemma are a part of the proof of Lemma 4. They yield a somewhat roundabout proof for Lemma 4, which is meant to provide intuition for the correctness of some of the main propositions.

DEFINITION 2. The *majority function*, denoted by *maj*, returns 1 if at least $\lfloor \frac{n}{2} \rfloor + 1$ players have the bit 1, and 0 otherwise.

Let E be the set of input vectors such that the number of ones is at most $\lfloor \frac{n}{2} \rfloor + 1$, and denote by $maj_{|E}$ the majority function restricted to the games in E.

LEMMA 3. *The nondeterministic communication complexity of $maj_{|E}$ is $\Omega(n)$.*

PROOF. We exhibit a fooling set of size $\binom{n}{n'}$, where $n' = \lfloor \frac{n}{2} \rfloor + 1$; the result follows from Lemma 2. The fooling set consists of all binary vectors of length n with exactly n' ones. Clearly, for any vector there is a majority of ones, and thus it remains to show that for any two vectors $\vec{w^1}$ and $\vec{w^2}$ in the fooling set, we can create a vector $\vec{w}$ where $w_i = w_i^1$ or $w_i = w_i^2$ for all $i \in N$, in such a way that $maj(\vec{w}) = 0$. Indeed, for any two such vectors, there must be $i_0 \in N$ such that $w_{i_0}^1 = 1$ but $w_{i_0}^2 = 0$. Let $w_i = w_i^1$ for all $i \neq i_0$, and $w_{i_0} = w_{i_0}^2 = 0$. $\vec{w}$ has exactly $n' - 1$ ones, as required. □

LEMMA 4. *Let G be the set of weighted-majority games $[q; w_1, w_2, \ldots, w_n]$ where $q = \lfloor \frac{n-1}{2} \rfloor + 2$, such that $w_i \in \{0,1\}$ for all i, $w_1 = 1$, and $\#\{i : i \geq 2 \wedge w_i = 1\} \leq \lfloor \frac{n-1}{2} \rfloor + 1$.*

Assume that some singleton solution concept satisfies the following property:

$$\forall g \in G,\ maj_{|_E}(w_2, w_3, \ldots, w_n) = 1 \implies x_1(g) > 0 \text{ for the grand coalition.} \quad (1)$$

Then the nondeterministic communication complexity of computing the payoff of an arbitrary agent in this solution concept is $\Omega(n)$ (even to decide whether the payoff of a given agent is greater than 0).

PROOF. Notice that all games in G are superadditive by Lemma 1, and thus we can assume the grand coalition forms. There is an obvious reduction from $maj_{|_E}$ with $n-1$ players to games in G: given an input $\langle w_2, w_3, \ldots, w_n\rangle$ for $maj_{|_E}$, complete the vector of weights with $w_1 = 1$, and set $q = n' + 1$, where $n' = \lfloor \frac{n-1}{2} \rfloor + 1$. Assuming property (1) holds, then $maj(w_2, w_3, \ldots, w_n) = 1$ if and only if $x_1(g) > 0$: "only if" follows from the fact that if $maj(w_2, w_3, \ldots, w_n) = 0$, then $\sum_{i\geq 2} w_i < n'$, and so the value of all coalitions is 0.

We have that if the nondeterministic communication complexity of deciding whether $x_1 > 0$ is $f(n)$ for some function f, then the nondeterministic communication complexity of $maj_{|_E}$ with $n-1$ players is at most $f(n)$. From Lemma 3, $f(n) = \Omega(n-1) = \Omega(n)$. □

We now prove our lower bounds. We start with the solution concepts that correspond to a singleton set of stable payoff configurations. These proofs rely on Lemma 4.

PROPOSITION 2. *The nondeterministic communication complexity of computing the payoff of an arbitrary agent according to the Shapley value is $\Omega(n)$ (even to decide whether the payoff of a given agent is greater than 0).*

PROOF. Let G be the set of weighted-majority games $[q; w_1, w_2, \ldots, w_n]$ where $q = \lfloor \frac{n-1}{2} \rfloor + 2$, such that $w_i \in \{0, 1\}$ for all i, $w_1 = 1$, and $\#\{i : i \geq 2 \wedge w_i = 1\} \leq \lfloor \frac{n-1}{2} \rfloor + 1$. We show that the Shapley value has property (1); this is sufficient to complete the proof by Lemma 4.

Fix a game from G, and let $S^* = \{i : w_i = 1\}$ (this coalition includes player 1). If we assume that

$$maj_{|_E}(w_2, w_3, \ldots, w_n) = 1,$$

then $|S^*| = q$; it follows that $x_1 > 0$, since:

$$\begin{aligned} x_1 &= \sum_{S \subseteq N} \frac{(|N| - |S|)!(|S| - 1)!}{|N|!}(v_S - v_{S-\{1\}}) \\ &\geq \frac{(|N| - |S^*|)!(|S^*| - 1)!}{|N|!}(v_{S^*} - v_{S^*-\{1\}}) \\ &= \frac{(n-q)!(q-1)!}{n!} \cdot (1 - 0) \\ &> 0. \end{aligned}$$

In fact, since all q players with $w_i = 1$ are interchangeable, and the rest are null players, we have from the axioms that characterize the Shapley value that $x_1 = \frac{1}{q}$. □

PROPOSITION 3. *The nondeterministic communication complexity of computing the payoff of an arbitrary agent according to the nucleolus[7] and the modified nucleolus is $\Omega(n)$ (even to decide whether the payoff of a given agent is greater than 0).*

[7]Recall that the nucleolus is a singleton for a given coalition structure. Here we investigate the nucleolus of the grand coalition.

PROOF. Let G be the set of weighted-majority games $[q; w_1, w_2, \ldots, w_n]$ where $q = \lfloor \frac{n-1}{2} \rfloor + 2$, such that $w_i \in \{0, 1\}$ for all i, $w_1 = 1$, and $\#\{i : i \geq 2 \wedge w_i = 1\} \leq \lfloor \frac{n-1}{2} \rfloor + 1$. We show that the nucleolus has property (1); this is sufficient to complete the proof by Lemma 4.

Fix a game from G, and let $\vec{x}$ be the nucleolus of the grand coalition. We assume that $maj(w_2, \ldots, w_n) = 1$; it follows that $\#\{i : w_i = 1\} = q$. We wish to show that $x_1 > 0$. Consider the payoff division $\vec{x^*}$, where $x^*_i = \frac{1}{q}$ for all i such that $w_i = 1$, and $x^*_i = 0$ otherwise. $\mathrm{e}(R, \vec{x^*}) = 0$ for coalitions R that contain all players with $w_i = 1$, and coalitions that do not contain any of these players. Moreover, for any other coalition, $\mathrm{e}(R, \vec{x^*}) < 0$: if the coalition contains $k < q$ players with $w_i = 1$, then $\mathrm{e}(R, \vec{x^*}) = -\frac{k}{q}$.

Now, assume $x_1 = 0$. If there exists i such that $w_i = 0$ but $x_i > 0$, then the coalition $S^* = \{i : w_i = 1\}$ must have $\mathrm{e}(S^*, \vec{x}) > 0$, since $v(S^*) = 1$ but $x(S^*) < 1$. But this means $\vec{\theta}(\vec{x}) \succ \vec{\theta}(\vec{x^*})$, and thus we can assume the payoff is distributed only among the players with $w_i = 1$. $\mathrm{e}(R, \vec{x}) = 0$ for coalitions R that contain all players with $w_i = 1$, and all coalitions that do not contain any of these players. Additionally, it holds that $\mathrm{e}(\{1\}, \vec{x}) = 0$. The number of zeros in $\vec{\theta}(\vec{x})$ is greater than the number of zeros in $\vec{\theta}(\vec{x^*})$, and thus $\vec{\theta}(\vec{x}) \succ \vec{\theta}(\vec{x^*})$; this is a contradiction to the assumption that $\vec{x}$ is the nucleolus.

For the modified nucleolus we have to slightly change the end of the proof, because the excess of coalitions of size 1 is no longer considered. However, observe that if $x_1 = 0$ (and we still have that $x_i = 0$ for all players with weight 0), there must be a player i_0 with $w_i = 1$ and $x_i \leq \frac{1}{q-1}$; hence, $\mathrm{e}(\{1, i_0\}, \vec{x}) \geq -\frac{1}{q-1}$. It holds that the number of 0's in $\vec{\theta}(\vec{x})$ and $\vec{\theta}(\vec{x^*})$ is equal, but the next smallest coordinate in $\vec{\theta}(\vec{x})$ is at least $-\frac{1}{q-1}$, while in $\vec{\theta}(\vec{x^*})$ it is $-\frac{2}{q}$. Hence, $\vec{\theta}(\vec{x}) \succ \vec{\theta}(\vec{x^*})$ — a contradiction to our assumption that $\vec{x}$ is the modified nucleolus of the grand coalition. □

PROPOSITION 4. *The nondeterministic communication complexity of computing the payoff of an arbitrary agent according to m-round and ∞-round equal excess is $\Omega(n)$ (even to decide whether the payoff of a given agent is greater than 0).*

PROOF. Let G be the set of weighted-majority games $[q; w_1, w_2, \ldots, w_n]$ where $q = \lfloor \frac{n-1}{2} \rfloor + 2$, such that $w_i \in \{0, 1\}$ for all i, $w_1 = 1$, and $\#\{i : i \geq 2 \wedge w_i = 1\} \leq \lfloor \frac{n-1}{2} \rfloor + 1$. We show that m-round equal excess has property (1); this is sufficient to complete the proof by Lemma 4.

Fix a game from G, and let $S^* = \{i : w_i = 1\}$; we assume that $|S^*| = q$. Observe that for all rounds r and players $i \in S^*$: $E^r(i, S^*) = 1/q$, by the symmetry of all players in S^*. Moreover, clearly it holds (again, by the symmetry of the players in S^*) that for all rounds r, $i \in S^*$ and $T \subseteq N$: $E^r(i, T) \leq 1/q$. Therefore,

$$\forall r, \forall i \in S^*, \forall T \subseteq N \text{ s.t. } S^* \subsetneq T :\ A^r(i, N) = 1/q. \quad (2)$$

For all players $i \notin S^*$ it holds that:

$$\forall T \subseteq N :\ E^0(i, T) \leq E^0(i, T^*) = \frac{1}{q+1}, \quad (3)$$

where $T^* = S^* \cup \{i\}$. In subsequent rounds r, we claim that:

$$\forall i \notin S^*, \forall T \subseteq N :\ E^r(i, T) \leq \frac{1}{q+1}. \quad (4)$$

This can be proven by induction: the base is given by equation (3). For the induction step, we have that $A^r(i,T) \leq \frac{1}{q+1}$ from the assumption. Any coalition T with non-zero value of which player i is a member also contains S^*. For such coalitions:

$$\begin{aligned} E^{r+1}(i,T) &= A^r(i,T) + \frac{v(s) - \sum_{j\in T} A^r(j,T)}{|T|} \\ &\leq \frac{1}{q+1} \\ &\quad + \frac{1 - \sum_{j\in S^*} A^r(j,T) - \sum_{j\in T-S^*} A^r(j,T)}{|T|} \\ &\overset{(2)}{=} \frac{1}{q+1} - \frac{\sum_{j\in T-S^*} A^r(j,T)}{|T|} \\ &\leq \frac{1}{q+1} \end{aligned}$$

Consequently, for all rounds r:

$$\begin{aligned} E^{r+1}(1,N) &= A^r(1,N) + \frac{v(N) - \sum_{j\in N} A^r(j,N)}{n} \\ &\overset{(2)}{=} \frac{1}{q} \\ &\quad + \frac{1 - \sum_{j\in S^*} A^r(j,N) - \sum_{j\in N-S^*} A^r(j,N)}{n} \\ &\overset{(2)}{=} \frac{1}{q} - \frac{\sum_{j\in N-S^*} A^r(j,N)}{n} \\ &\overset{(4)}{\geq} \frac{1}{q} - \frac{n-q}{n}\frac{1}{q+1} \\ &> 0. \end{aligned}$$

For m-round equal excess it holds that $x_1 = E^m(1,N)$; this completes the proof. Observe that we have from the proof that even for $m = \infty$, $E^\infty(1,N) > 0$. □

We now wish to analyze the communication complexity of payoff division according to the core. The core is not necessarily a singleton, but may be empty. Clearly, determining whether the core is empty is easier than computing the payoff of a player in some payoff configuration that is in the core.

PROPOSITION 5. *The nondeterministic communication complexity of computing the payoff of an arbitrary agent according to the core is $\Omega(n)$ (even to decide whether the core is empty).*

PROOF. We exhibit a fooling set of size $\binom{n}{n'}$, where $n' = \lfloor \frac{n}{2} \rfloor + 1$; the result follows from Lemma 2. All the inputs in the fooling set correspond to weighted majority games with $q = n' - 1 = \lfloor \frac{n}{2} \rfloor$; thus, an input in the fooling set can be fully represented by the vector of the agents' weights. The vectors $\langle w_1, w_2, \ldots, w_n \rangle$ in the fooling set have 1 in exactly n' coordinates, and 0 in the other $n - n'$ coordinates. There are $\binom{n}{n'}$ such vectors. Fix a game $\vec{w}$ in the set; we wish to show the core is empty. Since the game is superadditive,[8] it is sufficient to show that there is no stable payoff division for the grand coalition. Indeed, let $\vec{x}$ be a payoff vector, and let $i_0 = \text{argmax}_i\{x_i : w_i = 1\}$. The coalition S of q players i with $w_i = 1$ and $i \neq i_0$ benefits by deviating, since $v(S) = 1 = x(S) + x(N-S) > x(S)$. It follows that $\vec{x}$ cannot be in the core.

We still need to show that for any two vectors $\vec{w^1}$ and $\vec{w^2}$ in the fooling set, we can create a vector $\vec{w}$ where $w_i = w_i^1$ or $w_i = w_i^2$ for all $i \in N$, in such a way that the core is non-empty. For any two such vectors, there must be $i_0 \in N$ such that $w_{i_0}^1 = 1$ but $w_{i_0}^2 = 0$. Let $w_i = w_i^1$ for all $i \neq i_0$, and $w_{i_0} = w_{i_0}^2 = 0$. $\vec{w}$ has 1 in exactly q coordinates. It is clear that the following payoff distribution is in the core: $x_i = 1/q$ for all i such that $w_i = 1$, and $x_i = 0$ for all other players. □

[8]The proof of superadditivity is similar to Lemma 1, and relies on the fact that if $v(T) = 1$, T contains at least q players with $w_i = 1$, and so $N - T$ contains at most one such player.

3.3 Restricted Games

In Section 2, we defined some restricted types of games. In this subsection, we shall look more closely at apex and veto games. These two types of games have natural, real-world interpretations: in an apex game, the apex player may be described as a limited monopolist (who requires only one ally), while veto games are similar to such political bodies as the United Nations security council.

Apex and veto games are particularly interesting in the context of communication complexity, since a great deal of information about the game can be communicated by simply naming the apex or veto player.

PROPOSITION 6. *In constant-sum apex games, the* deterministic *communication complexity of computing the payoff of an arbitrary agent in any solution concept is 1.*

PROOF. Recall that constant-sum apex games have the weighted majority representation

$$[n-1; 1, \ldots, 1, n-2, 1, \ldots, 1],$$

where $n-2$ is the weight of the apex agent, and suppose w.l.o.g. that the goal is to compute the payoff of agent 1. It is sufficient that agent 1 simply announce (using one bit) whether it is the apex player; it can do this by determining whether its weight is $n-2$ (in other words, checking that it has $n-2$ resource units). By the symmetry between the non-apex agents, this knowledge is sufficient to compute the payoff of agent 1. □

Moreover, many important veto games have the weighted majority representation:

$$[n; 1, \ldots, 1, n-1, 1, \ldots, 1],$$

where $n-1$ is the weight of the veto player. Obviously, the deterministic communication complexity of payoff division in these games is also 1.

4. CONCLUSIONS AND FUTURE WORK

In an environment with n-agents, each holding a constant amount of information (the agent's utility function and initial resources), we have shown that the communication complexity of computing the payoff of an arbitrary player is $\Theta(n)$ in the following solution concepts: Shapley value, the nucleolus and modified nucleolus, equal excess theory, and the core. Additionally, we have shown that in constant-sum apex games and certain veto games, the deterministic communication complexity is 1.

As our upper bound is trivial, clearly the significance of the results lies in the lower bounds. Fortunately, a communication complexity lower bound of $\Omega(n)$ is usually not an obstacle. Nevertheless, when interpreted negatively, our results show that solving cooperative games may be infeasible in scenarios where the communication is severely restricted, or the number of agents is very large.

Our approach has some limitations. The results in this paper are relevant to coalition formation in our specific environment model; the bounds on communication complexity may be different in other models, although the model which was our focus is an intuitive and well-known one.

Another possible criticism of this work is that, even if a payoff division among the n players is known in advance, in general it takes $O(n)$ communication to broadcast this solution to the players. Nevertheless, recall that our goal here was to determine the value of a specific player in a given solution concept. As we noted in the introduction, this problem is also important in its own right. Our analysis of restricted games shows that $\Omega(n)$ communication is not a lower bound for this problem in certain games.

There are several appealing directions in which this research can be extended. An important task is to characterize the "interesting" games in which payoff division has a communication complexity of $o(n)$ (*small* O). A related issue is to determine whether there exist reasonable singleton solution concepts with communication complexity of $o(n)$ in general characteristic function games. It may also be the case that the communication complexity may be lowered for certain solutions, but only in specific games (where other solutions are still hard), or particular environments.

Our methods of obtaining lower bounds have prevented us from investigating such important solutions as the kernel and the bargaining set (see [5]). These solutions should also be examined in the future.

Another direction is to augment our model by adding another parameter m, which is the amount of information each agent holds (in our environment $m = O(1)$, since we assumed that each agent holds a constant amount of information). We expect that the bounds for many of the solution concepts can be generalized to $\Theta(n \cdot \log m)$, but the same issues we studied here may be even more explicit in this augmented model.

5. ACKNOWLEDGMENT

This work was partially supported by grant #039-7582 from the Israel Science Foundation.

6. REFERENCES

[1] V. Conitzer and T. Sandholm. Complexity of determining nonemptiness of the core. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 613–618, 2003.

[2] V. Conitzer and T. Sandholm. Computing Shapley values, manipulation value division schemes, and checking core-membership in multi-issue domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 219–225, 2003.

[3] V. Conitzer and T. Sandholm. Communication complexity of common voting rules. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 78–87, 2005.

[4] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, pages 257–266, 1994.

[5] J. P. Kahan and A. Rapoport. *Theories of Coalition Formation*. Lawrence Erlbaum Associates, 1984.

[6] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[7] T. Sandholm and V. Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1):99–137, 1997. Special issue on Economic Principles of Multiagent Systems.

[8] O. Shehory and S. Kraus. Coalition formation among autonomous agents: Strategies and complexity. In *From Reaction to Cognition*, Lecture Notes in Artificial Intelligence Number 957, pages 57–72, 1995.

[9] T. Sandholm. Distributed rational decision making. In G. Weiß, editor, *Multiagent Systems: A Modern Introduction to Distributed Artificial Intelligence*, chapter 5. MIT Press, 1999.

[10] A. C. Yao. Some complexity questions related to distributed computing. In *Proceedings of the 11th ACM Symposium on Theory of Computing (STOC)*, pages 209–213, 1979.

The Cost of Social Agents*

Guido Governatori
School of ITEE
The University of Queensland
Brisbane, Australia
guido@itee.uq.edu.au

Antonino Rotolo
CIRSFID
University of Bologna
Bologna, Italy
rotolo@cirsfid.unibo.it

Vineet Padmanabhan
School of ITEE
The University of Queensland
Brisbane, Australia
vnair@itee.uq.edu.au

ABSTRACT

In this paper we follow the BOID (Belief, Obligation, Intention, Desire) architecture to describe agents and agent types in Defeasible Logic. We argue that the introduction of obligations can provide a new reading of the concepts of intention and intentionality. Then we examine the notion of social agent (i.e., an agent where obligations prevail over intentions) and discuss some computational and philosophical issues related to it. We show that the notion of social agent either requires more complex computations or has some philosophical drawbacks.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods

General Terms

Theory, Legal Aspects, Algorithms, Performance

Keywords

Defeasible logic, Intention, Obligation, Social Agents, Computational Complexity

1. INTRODUCTION

Recent works on cognitive agents combine two apparently independent perspectives [6, 14, 8, 7, 9, 10]: (a) a classical cognitive account of agents that specifies their mental attitudes; (b) modelling agents' behaviour by means of normative concepts. For the first approach, the background is the belief-desire-intention (BDI) architecture, where mental attitudes are taken as primitives to give rise to a set of Intentional Agent Systems [20, 4]. This view is interesting especially when the behaviour of agents is the outcome of a rational balance among their (possibly conflicting) mental states. The normative aspect is rather based on the assumption that normative concepts play a role to characterise the idea of social co-ordination of autonomous agents [19]. The nice result of this combination of perspectives is that of leading to an account of agents' deliberation and behaviour in terms of the interplay between mental attitudes and normative (external) factors such as obligations.

Following [14, 8, 7] we discuss how this combination can be framed in Defeasible Logic (DL). As is well-known, DL is based on a logic programming-like language and it is a simple, efficient but flexible non-monotonic formalism able to deal with many different intuitions of non-monotonic reasoning and recently applied in many fields. In addition, several efficient implementations have been developed [17, 3]. Here we discuss and extend some aspects of a non-monotonic logic of agency, based on the framework of [1], developed in [8, 7]. Indeed, DL is one of the most expressive languages that allows for the definition of large sets of patterns called agent types. Here, the focus will be in particular on philosophical and computational aspects of the notion of social agent, by which we mean a norm-complying agent.

The layout of the paper is as follows. Section 2 provides the theoretical background of our system. In particular, since the concept of social agent focuses on the interplay between obligations and intentions, we will discuss which kind of intentions have to be considered in this regard. Section 3 will present our logical framework, based on DL, which will embed our intuitions and permit to deal with social agency. Some first conclusions will be provided with reference to the problematic nature of social agents. Section 4 deals with the computational complexity of social agency.

2. SOCIAL AGENTS: OBLIGATION AND INTENTION

This section provides some theoretical background for the rest of the paper. Our focus is on the so-called policy-based attitudes. The term was coined by Bratman [5] with specific reference to the idea of intention. The intuition behind policy-based intention is based on Bratman's view regarding *future directed intention* and *general intention*. Bratman terms general intentions as *general/personal policies*. Along with general policies go policy-based intentions. For example, I have a general policy to patch up and reboot the Unix server in the department once every month. This morning, on the basis of this policy, I form the intention to reboot the machine at 7.00 pm in the evening. My intention this morning to reboot the machine this evening is a *policy-based* intention. This specific intention will play a major part in my planning process for the day as it will pose problems about means and constrain my other options. Based on this distinction Bratman makes the following classification of intentions: *deliberative, non-deliberative* and *policy-based*.

The difference between the three is the following: When an agent i has an intention of the form $\mathrm{INT}_i^{t_1}\varphi, t_2$ (read as *agent i intends at*

*This work is supported by the Australian Research Council under the Discovery Project DP0558854 on "A Formal Approach to Resource Allocation in Service Oriented Marketplaces".

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

t_1 *to* φ *at* t_2) as a process of *present* deliberation, then it is called *deliberative intention*. On the other hand, if the agent comes to have such an intention not on the basis of present deliberation, but at some earlier time t_0 and has retained it from t_0 to t_1 without reconsidering it, then this intention it is called *non-deliberative*. The third case arises when intentions are general and concern potentially recurring circumstances in an agent's life. Such general intentions constitute *policy-based intentions*. A policy-based intention is not a non-deliberative intention because it is not simply a case of retaining an intention previously formed. Neither is it a deliberative intention since it is not based on a full-blown deliberation where an attempt is made to weigh pros and cons for and against conflicting options. It also differs from an intention in favour of necessary means, i.e., intention in favour of a specific end, in the sense that the defeasibility of general policies makes it possible to *block* the application of the policy to the particular case without *abandoning* the policy. Otherwise one could abandon the intention in favour of the end. The difference here is that in each case the policy concerns not just a single future situation, but a kind of circumstance that is expected to recur in the agent loop and in each case the agent might well have a general intention to act in the particular circumstances. Whether the agent is able to perform that action or not depends on the circumstances.

As argued in detail elsewhere [13], it may happen that a policy-based intention needs to be *re-considered* if not *blocked* for the application to particular cases. But this does not mean that the agent should know all such conditions in a scenario, but only those she considers necessary for the intended outcome and that she is not confident of their being satisfied. To intend the necessary consequence the agent has to make sure that all the evidence to the contrary has been defeated, which is basically a defeasible conclusion.

The starting point of this paper is to extend the policy-based approach to other attitudes and motivational factors such as beliefs and obligations. In this way, all motivational factors are represented within a rule-based system: intentions and beliefs are viewed as constituting the internal constraints (based on policies) of an agent while obligations are her external constraints (based on rules). As constraints they are defeasible. Notice, in particular, that such an extension to obligations can capture the well-known defeasible character of deontic reasoning. In this last case, a policy-based obligation –conceived of as an external motivational attitude– turns out to be simply a conditional obligation, namely, a rule that allows for the inference of an obligation whenever the antecedent of this rule holds [18, 21].

2.1 Intentions and Normativity

According to Bratman, rational agents can be basically modelled as follows:

- agents are goal-directed without being necessarily aware of their activity;
- intentions are used to choose partial plans for the realisation of a goal;
- not all consequences are intended but only some initial intentions and the goal as a result of the plan; if some side-effects occur, they are never intended.

It is worthy discussing here the notion of side-effect. This well-known problem has to deal with several variants of logical omniscience: the problem arises when the agent is required to know all the truths defined by her logic, or when the logic that depicts the agent automatically includes all the logical truths of classical logic, or, finally, if the agent knows all the logical consequences of the known propositions [12]. Indeed, the problem is usually referred to as the *expected side-effects* problem [5], a problem which depends on the interactions between the reasoning mechanism for the propositional inferences and the mechanism ruling the introduction and the behaviour of the modal operators. A simple and rather unsatisfactory solution would be to consider two completely unrelated consequence relations, one for the propositional part and the second one for the modal operators. The consequence relation for a modal operator is meant to give the condition under which one can prove a modal formula. For example the pair $\Gamma \mid\sim_{\mathbf{X}} \alpha$, where $\mathbf{X}$ is a modal operator, means that if we can prove all the formulas in Γ then we can deduce $\mathbf{X}\alpha$. In what follows we will develop a system for mental states and motivational attitudes based on this idea. However, we will allow the consequence relation for intentions and obligations to interact with the propositional module and we will also consider possible interactions between the modal operators. To this end we have to show that the expected side-effects phenomenon is not a drawback for policy-based agents: such a kind of agents must accept the expected-side effects unless they have some reasons to reject the consequences corresponding to them.

In effect, though our proposed theory does not entertain many of the properties leading to logical omniscience, some aspects of the side-effects problem are accepted. Consider

$$\mathrm{INT}\mathit{Smoke},\ \mathit{Smoke} \rightarrow \mathit{Cancer} \mid\sim \mathrm{INT}\mathit{Cancer} \quad (1)$$

$$\mathrm{INT}\mathit{GoToRome},\ \mathit{GoToRome} \rightarrow \mathit{GoToItaly} \mid\sim \mathrm{INT}\mathit{GoToItaly} \quad (2)$$

Actually, whereas the first case is clearly unacceptable, the second should be accepted by a rational agent. In this perspective the side-effects problem is similar to the substitution of indiscernible in opaque contexts. An agent may have the intention to visit Rome and not to visit Italy. But if the agent knows that Rome is the capital of Italy then it would be irrational of the agent not to have the intention to go to Italy given the intention to visit Rome.

Accordingly, some cases of the side-effects problem are not necessarily a weakness of a theory. This holds in particular if we assume that our agents are *aware* of their activities. In our view, modelling rational agents corresponds to the following assumptions:

- agents are aware of their activities, of their policies;
- some cases of the side-effects problem can be accepted;
- if a case has to be rejected this means that its unpleasant consequences should not be intended;
- when unpleasant consequences are not intended, this only means that they are blocked by conflicting attitudes or facts.

The theory an agent is equipped with can be understood as the specification of the behaviour of the agent. If the agent is *aware* that B is an unavoidable/indisputable consequence of A and the agent intends A, then B is a consequence of the agent's intentions and the agent must accept it as part of her intentions. Suppose we have that "raising one's hand at an auction counts as making a bid". Thus if the agent (aware of this policy) intends to raise her hand, then she intends to bid in the auction, and her action will be understood as making a bid. In other words, in our system we will try to balance and moderate some unpleasant aspects of the side-effects problem with the equally important need for modelling rational agents. Of course, according to our view, we may have that something is intended even if it is causally distant with respect to the original derived intentions. But this is not necessarily a drawback if we conceive agents as rational and, as such, being aware of the policies which are related with the environment and with their interests: even a causally distant behaviour can be rationally

intended unless it is removed in the meantime from deliberation. But this case is indeed considered within our analysis because we may have concrete contexts in which some policy-based intentions, as soon as they are applicable, turn out to be overridden by other policies: we may have reasons to argue that, if an agent intends A and believes that B is a consequence of A, this is not a reason for necessarily intending B; in fact, the derivation of B as an intention may be blocked, in our view, by competing attitudes or made non-applicable by concrete facts.

According to the previous discussion it should be clear that, though inspired by Bratman's [5] analysis, the notion of intention we study in this paper is slightly different, as it focuses on the idea of *intentionality*. In Bratman's view intentions are used to choose partial plans for the realisation of a goal; in this way they have a close relation to means-ends. In our view intentions should be related not only to means-ends but also to the their consequences.

This concept of intention is particularly relevant in conjunction with deontic and normative notions, for example if we want to say that an agent is legally responsible for A if the agent did A with the intention to do A. In such cases the agent has to include in the set of her intentions not only her intentions in Bratman's sense but also some of their consequences. Our intuition is compatible with von Wright's [22] classical theory of normative actions. Von Wright's problem is to identify what should be the content of norms. He argues that norms should deal with actions. Roughly, actions can be described in terms of state transitions and as the sets of all changes of world that follow from them. It is not our purpose discussing here von Wright's theory of action. It should be noted, however, that he considers the related problem of intentions. On the one hand, von Wright is clear when he says that any action may have an arbitrary number of consequences and not all of them are intended. On the other hand, he provides a very broad concept of action, according to which all actions in norms, strictly speaking, are intentional. If so, what are the boundaries of intentions to be considered when they interplay with obligations?

Let us see how to recast Bratman's Strategic Bomber scenario [5] in this perspective. The basic scenario runs as follows: Strategic Bomber intends to bomb a munition plant of the enemy being aware that the resulting explosion will kill innocent children in a nearby school. Bratman argues that Strategic Bomber does not have the intention to kill the children. Let us expand the scenario by supposing that despite the bombing, Strategic Bomber loses the war, and that there is a process for war crimes against him. Civil casualties are a sad but almost unavoidable consequence of war, but usually the killing of civilians does not constitute a war crime if there was no intention to kill. According to Bratman, Strategic Bomber did not commit a war crime since he did not have such an intention. However, let us assume that Strategic Bomber did not do anything to prevent or minimise civil casualties (let us say by a movement of troops that might have resulted in an evacuation of the area surrounding the munition plant). In this extended scenario the killing of children is brought about by a (successful) intentional act of Strategic Bomber. Accordingly, he must be held responsible for the killing of innocent civilians.

Given this interpretation of intentions, we will see in the rest of this paper that some standard accounts of agent types, and of social agents in particular, are not satisfactory. We provide now some brief comments on the notion of agent type and social agent.

Classically, agent types are characterised by stating conflict resolution types in terms of orders of overruling between rules [6, 14, 7, 8]. In this perspective, agent types are meaningful within a nonmonotonic setting and are nothing but general strategies to detect and solve conflicts between the different components of the cognitive profiles of agent's deliberation. In [6] 24 possible types are identified while, in [8], based on a different framework, 20 combinations are proposed. Typically, rational agents are assumed to be at least *realistic*: a realistic agent, in fact, is such that rules for beliefs override all other components, as beliefs correspond to agent's account of how the environment is. If the realistic condition is abandoned, we may have situations where intentions and desires override beliefs, thus leading to various forms of wishful thinking.

Given the minimal assumption that a rational agent should be realistic, we may further constrain agent's deliberation in order not to violate obligations: a *social agent* type requires that obligations are stronger than the other motivational components with the exception of beliefs. Various forms of social agency correspond to assuming axiom schemata such as

$$\mathrm{OBL}\phi \rightarrow \neg\mathrm{INT}\neg\phi \tag{3}$$

$$\mathrm{OBL}\phi \rightarrow \neg\mathrm{DES}\neg\phi \tag{4}$$

$$(\mathrm{OBL}\phi \wedge \mathrm{DES}\neg\phi) \rightarrow \neg\mathrm{INT}\neg\phi \tag{5}$$

In this paper we will consider only the interaction between intentions and obligations. But, even confining the problem to these components, the question at stake is: How to deal with social agents? The simplest solution is the classical one, corresponding to adopting schema (3): when we have two rules, one leading to $\mathrm{INT}\phi$ and the other to $\mathrm{OBL}\neg\phi$, the former is blocked. Are we sure that this classical view is sufficient, given the account of policy-based attitudes we previously discussed?

3. BIO AGENTS IN DEFEASIBLE LOGIC

We focus on how beliefs, intentions and obligations jointly interplay in modelling agent's deliberation and behaviour. In particular, the system is meant to infer the goals the agent has to achieve.

The formal language contains modal literals and preferences, and is defined as follows:

DEFINITION 1. *Let $M = \{\mathrm{BEL}, \mathrm{INT}, \mathrm{OBL}\}$ be a set of modal operators, and P a set of propositional atoms. The set of literals is defined as $L = P \cup \{\neg p | p \in P\}$. If q is a literal, $\sim q$ denotes the complementary literal (if q is a positive literal p then $\sim q$ is $\neg p$; and if q is $\neg p$, then $\sim q$ is p).*

- *The language $\mathscr{L}$ is the smallest set including L and containing modal literals Xl and $\neg Xl$ when $l \in L$ is a literal and $X \in M$ is a modal operator.*

For $X \in \{\mathrm{BEL}, \mathrm{INT}, \mathrm{OBL}\}$, we have that $\phi_1, \ldots, \phi_n \rightarrow_X \psi$ is a *strict rule* such that whenever the premises $\phi_1, \ldots, \phi_n$ are indisputable so is the conclusion ψ. $\phi_1, \ldots, \phi_n \Rightarrow_X \psi$ is a *defeasible rule* that can be defeated by contrary evidence. A rule $\phi_1, \ldots, \phi_n \rightsquigarrow_X \psi$ is a *defeater* that is used to defeat some defeasible rules by supporting evidence to the contrary.

DEFINITION 2. *A* rule *r consists of its* antecedent *(or* body*) $A(r)$ ($A(r)$ may be omitted if it is the empty set), an arrow ($\rightarrow$ for a strict rule, $\Rightarrow$ for a defeasible rule, and $\rightsquigarrow$ for a defeater), and its* consequent *$C(r)$ (or* head*).*

- *The arrow is labelled with a modal operator $X \in \{\mathrm{BEL}, \mathrm{INT}, \mathrm{OBL}\}$. If the arrow is labelled with* BEL *the rule is for belief, and similarly for the other modal operators.*
- *Given a rule r, $A(r)$ is a set of literals or modal literals, and $C(r)$ is a literal.*
- *Given a set R of rules, we denote the set of all strict rules in R by R_s, the set of strict and defeasible rules in R by R_{sd}, the*

set of defeasible rules in R by R_d, and the set of defeaters in R by R_{dft}. $R[q]$ denotes the set of rules in R with consequent q. If any R is labelled with X, that is R^X, all expressions just defined will refer to rules for X.

The purpose of the system is to derive modalised literals (goals), with the exception of rules for beliefs. As we shall see, provability for beliefs will not generate goals, as in our view beliefs concern the knowledge an agent has about the world: they may contribute to derive goals (here, intentions and obligations), but they are not in themselves motivations for action. For example, the application of $p \Rightarrow_{\text{OBL}} q$ permits to infer OBLq. Accordingly, modalities will not occur in the consequents of rules to keep the system manageable.

DEFINITION 3. *A defeasible agent theory is a structure $D = (F, R^{\text{BEL}}, R^{\text{INT}}, R^{\text{OBL}}, >)$ where F is a finite set of facts, R^{BEL} is a finite set of rules for belief, R^{INT} is a finite set of rules for intention, R^{OBL} is a finite set of rules for obligation, and $>$, the superiority relation, is a binary relation over the set of rules.*

The *superiority relation* $>$ says when one rule may override the conclusion of another rule. *Facts* are indisputable statements.

The following example illustrates the agent theory.

EXAMPLE 1. (RUNNING EXAMPLE). Frodo, our Tolkienian agent, is entrusted by Elrond to be the bearer of the ring of power, a ring forged by the dark lord Sauron. Frodo has the task to bring the ring to Mordor, the realm of Sauron, and to destroy it by throwing it into the fires of Mount Doom. However, Frodo loves the place where he was born, the Shire, and intends to go there.

$$
\begin{aligned}
F &= \{\text{INT}\mathit{GoToShire},\ \mathit{EntrustedByElrond}\} \\
R &= \{r_1 : \mathit{EntrustedByElrond} \Rightarrow_{\text{BEL}} \mathit{RingBearer} \\
&\quad\ r_2 : \mathit{RingBearer} \Rightarrow_{\text{OBL}} \mathit{DestroyRing} \\
&\quad\ r_3 : \text{INT}\mathit{GoToShire} \Rightarrow_{\text{INT}} \neg\mathit{GoToMordor} \\
&\quad\ r_4 : \neg\mathit{GoToMordor} \Rightarrow_{\text{BEL}} \neg\mathit{DestroyRing}\} \\
> &= \{r_4 > r_2\}
\end{aligned}
$$

3.1 Inferences with Social Agents

DEFINITION 4. *Given an agent theory D, a proof in D is a linear derivation, i.e, a sequence of labelled formulas of the type $+\Delta_X q$, $-\Delta_X q$, $+\partial_X q$ and $-\partial_X q$, where the proof conditions defined in the rest of this section hold.*

The meaning of the proof tags $+\Delta_X$, $-\Delta_X$, $+\partial_X$ and $-\partial_X$ is as follows: $+\Delta_X q$ means that q is definitly provable using only facts and strict rules for X, $-\Delta_X q$ means that it has been proved that q is not definitely provable, $+\partial_X q$ that q is defeasibly provable in D and $-\partial_X q$ that q is not defeasibly provable.

We start with some terminology. As was explained, the following definition states the special status of belief rules, and that an introduction of a modal operator corresponds to being able to derive the associated literal using the rules for the modal operator.

DEFINITION 5. *Let $\# \in \{\Delta, \partial\}$, and $P = (P(1), \ldots, P(n))$ be a proof in D. A (modal) literal q is* #-provable *in P if there is a line $P(m)$ of P such that either*

1. *q is a literal and $P(m) = +\#_{\text{BEL}} q$ or*
2. *q is a modal literal Xp and $P(m) = +\#_X p$ or*
3. *q is a modal literal $\neg Xp$ and $P(m) = -\#_X p$.*

A literal q is #-rejected *in P if there is a line $P(m)$ of P such that*

1. *q is a literal and $P(m) = -\#_{\text{BEL}} q$ or*
2. *q is a modal literal Xp and $P(m) = -\#_X p$ or*
3. *q is a modal literal $\neg Xp$ and $P(m) = +\#_X p$.*

The definition of Δ_X describes just forward chaining of strict rules:

$+\Delta_X$: If $P(n+1) = +\Delta_X q$ then
(1) $q \in F$ if $X = \text{BEL}$ or $Xq \in F$ or
(2) $\exists r \in R_s^X[q]\ \forall a \in A(r)$ a is Δ-provable or
(3) $\exists r \in R_s^{\text{BEL}}[q]\ \forall a \in A(r)$ Xa is Δ-provable.

$-\Delta_X$: If $P(n+1) = -\Delta_X q$ then
(1) $q \notin F$ if $X = \text{BEL}$ and $Xq \notin F$ and
(2) $\forall r \in R_s^X[q]\ \exists a \in A(r) : a$ is Δ-rejected and
(3) $\forall r \in R_s^{\text{BEL}}[q]\ \exists a \in A(r)$ Xa is Δ-rejected.

For a literal q to be definitely provable we need to find a strict rule with head q, whose antecedents have all been definitely proved previously. And to establish that q cannot be definitely proven we must establish that for every strict rule with head q there is at least one antecedent which has been shown to be non-provable. Condition (3) says that a belief rule can be used as a rule for a different modal operator in case all literals in the body of the rule are modalised with the modal operator we want to prove. Thus given the rule $p, q \rightarrow_{\text{BEL}} s$, we can derive $+\Delta_Y s$ if we have $+\Delta_Y p$ and $+\Delta_Y q$.

Conditions for ∂_X are more complicated. We define when a rule is applicable or discarded. A rule for a belief is applicable if all the literals in the antecedent of the rule are provable with the appropriate modalities, while the rule is discarded if at least one of the literals in the antecedent is not provable. For the other types of rules we have to take complex derivations into account called conversions [14]. In this paper we say there is a conversion from X to Y if a rule fior X can also be used as a rule for Y. We have thus to determine conditions under which a rule for X can be used to directly derive a literal q modalised by Y. Roughly, the condition is that all the antecedents a of the rule are such that $+\partial_Y a$.

We represent all allowed conversions by a conversion relation c.

DEFINITION 6. *Let a conversion relation c be a binary relation over $\{\text{BEL}, \text{INT}, \text{OBL}\}$, such that (1) $c(X,Y)$ stands for the conversion of X rules into Y rules, (2) $Y \neq$ BEL. Given a derivation P, $P(1..n)$ denotes the initial part of the derivation of length n.*

- *A rule r in R^{BEL} is* applicable *iff $\forall a \in A(r)$, $+\partial_{\text{BEL}} a \in P(1..n)$ and $\forall Za \in A(r)$, where Z is a modal operator, $+\partial_Z a \in P(1..n)$.*
- *A rule r in R^{BEL} is* discarded *iff $\exists a \in A(r)$ such that $-\partial_{\text{BEL}} a \in P(1..n)$ or $\exists Za \in A(r)$ such that $-\partial_Z a \in P(1..n)$.*
- *A rule $r \in R_{sd}$ is* applicable *in the condition for $\pm\partial_Y$ iff*
  1. *$r \in R^Y$ and $\forall a \in A(r)$, $+\partial_{\text{BEL}} a \in P(1..n)$ and $\forall Za \in A(r)$ $+\partial_Z a \in P(1..n)$, or*
  2. *$r \in R^X$ and $\forall a \in A(r)$, $+\partial_Y a \in P(1..n)$.*
- *A rule r is* discarded *in the condition for $\pm\partial_Y$ iff we prove either $-\partial_{\text{BEL}} a$ or $-\partial_X a$ for some $a \in A(r)$.*
  1. *$r \in R^Y$ and $\exists a \in A(r)$ such that $-\partial_{\text{BEL}} a \in P(1..n)$ or $\exists Za \in A(r)$ such that $-\partial_Z a \in P(1..n)$; or*
  2. *$r \in R^X$ and $\exists a \in A(r)$ such that $-\partial_Y a \in P(1..n)$.*

EXAMPLE 2. The rule $a, \text{INT}b \Rightarrow_{\text{BEL}} c$ is applicable if we can prove both $+\partial_{\text{BEL}} a$ and $+\partial_{\text{INT}} b$.

EXAMPLE 3. If we have a type of agent that allows a deontic rule to be converted into a rule for intention, $c(\text{OBL},\text{INT})$, then the definition of applicable in the condition for $\pm\partial_{\text{INT}}$ is as follows: a rule $r \in R_{sd}[q]$ is applicable iff (1) $r \in R^{\text{INT}}$ and $\forall a \in A(r), +\partial_{\text{BEL}} a \in P(1..n)$ and $\forall Xa \in A(r), +\partial_X a \in P(1..n)$, (2) or $r \in R^{\text{OBL}}$ and $\forall a \in A(r), +\partial_{\text{INT}} a \in P(1..n)$. In this second case, for example, given the rule $p, q \Rightarrow_{\text{OBL}} s$, we can derive $+\partial_{\text{INT}} s$ if we have $+\partial_{\text{INT}} p$ and $+\partial_{\text{INT}} q$.

As a corollary of the definition of applicability, we can establish when a literal is supported (see Section 4 for the use of this notion):

DEFINITION 7. *Given a theory $D = (F, R^{\text{BEL}}, R^{\text{INT}}, R^{\text{OBL}}, >)$, a literal l is* supported *in D iff there exists a rule $r \in R[l]$ such that r is applicable, otherwise l is not supported. For $X \in \{\text{BEL}, \text{INT}, \text{OBL}\}$ we use $+\Sigma_X l$ and $-\Sigma_X l$ to indicate that l is supported / not supported by rules for X.*

We are now ready to provide proof conditions for $\pm\partial_X$:

$+\partial_X$:If $P(n+1) = +\partial_X q$ then
  (1)$+\Delta_X q \in P(1..n)$ or
    (2.1) $-\Delta_X \sim q \in P(1..n)$ and
    (2.2) $\exists r \in R_{sd}[q]$ such that r is applicable; and
    (2.3) $\forall s \in R[\sim q]$, either s is discarded, or
      (2.3.1) $\exists t \in R[q]$ such that r is applicable and $t > s$

$-\partial_X$: If $P(n+1) = -\partial_X q$ then
  (1) $-\Delta_X q \in P(1..n)$ and either
    (2.1) $+\Delta_X \sim q \in P(1..n)$ or
    (2.2) $\forall r \in R_{sd}[q]$, either r is discarded, or
    (2.3) $\exists s \in R[\sim q]$, such that s is applicable, and
      (2.3.1) $\forall t \in R[q]$ either t is discarded, or $t \not> s$

To show that q is defeasibly provable we have two choices: (1) We show that q is already definitely provable; or (2) we need to argue using the defeasible part of a theory D. For this second case, three (sub)conditions must be satisfied. First, we require that there must be a strict or defeasible rule for q which can be applied (2.1). Second, we need to consider possible reasoning chains in support of $\sim q$, and show that $\sim q$ is not definitely provable (2.2). Third, we must consider the set of all rules which are not known to be inapplicable and which permit to get $\sim q$ (2.3). Essentially, each such a rule s attacks the conclusion q. For q to be provable, s must be counterattacked by a rule t for q with the following properties: (i) t must be applicable, and (ii) t must be stronger than s. Thus each attack on the conclusion q must be counterattacked by a stronger rule. In other words, r and the rules t form a team (for q) that defeats the rules s. $-\partial_X q$ is defined in an analogous manner.

EXAMPLE 4. (RUNNING EXAMPLE; CONTINUED). Below is the set C of all conclusions we get using the rules in R:

$$C = \{\textit{RingBearer}, \text{INT}\neg\textit{GoToMordor}, \text{INT}\neg\textit{DestroyRing}\}$$

As facts, we know that Frodo has the primitive intention to go to the Shire and that he has been entrusted by Elrond. These facts make applicable rules r_3 and r_1, which permit to derive that Frodo is the ring bearer and that he has the intention not to go to Mordor. At this point we have a conflict if we assume $c(\text{BEL},\text{INT})$. In effect, given this conversion, r_4 permits to derive that Frodo has the intention not to destroy the ring while rule r_2 should lead to the obligation to destroy it. However, r_4 is stronger than r_2 and so we only get $+\partial_{\text{INT}}\neg\textit{DestroyRing}$.

3.2 The Problem of Social Agents

As suggested in [7, 8], agent types can be characterised in DL as follows:

DEFINITION 8. *An agent type is defined by a set of pairs (X,Y), $X,Y \in \{\text{BEL},\text{OBL},\text{INT}\}$, such that for every r and r' such that $r \in R^X[q]$ and $r' \in R^Y[\sim q]$, we have that $r > r'$.*

While realistic agents are such that $X = \text{BEL}$ and $Y \in \{\text{INT},\text{OBL}\}$, social agents are such that $X = \text{OBL}$ and $Y = \text{INT}$.

Unfortunately, this definition –adopted also in [6]– does not guarantee that agent's deliberation is oriented to fully complying with obligations. This drawback is mainly due to the introduction of conversions. Indeed, the notion of conversion should not sound strange. In many formalisms we can convert from one type of conclusion into a different one. Take for example the right weakening rule of non-monotonic consequence relations, where it is possible to combine non-monotonic with classical consequences: $B \vdash C$ and $A \mid\sim B$ imply $A \mid\sim C$ [15]. Here, conversions simply allow to obtain conclusions modalised by a certain X through the application of rules which are not modalised by X. In particular, they are fundamental in order to capture the fact that some side-effects should be accepted insofar as they are consequences of policies of which the agent is aware. Finally, some conversions seem useful to integrate the basic idea of social agency. For example, we may have agent types for which, given $p \Rightarrow_{\text{OBL}} q$ and $+\partial_{\text{INT}} p$, we can obtain $+\partial_{\text{INT}} q$. Of course, this is possible only if we assume a kind of norm regimentation, by which we impose that all agents intend what is prescribed by deontic rules.

It is clear that our system admits three different types of intentions and obligations. First, we have *primitive* intentions and obligations when these are facts of the theory. But we can also have what we may call *primary* and *secondary* intentions and obligations, depending on whether we accept at least basic conversions via belief rules.

Let us consider Example 1. INT*GoToShire* is a primitive intention. On the other hand, OBL*DestroyRing* –if it were derived from rule r_2– and INT$\neg$*GoToMordor* are primary obligations and intentions as they would be obtained without the use of conversions (see Example 4). Finally, INT$\neg$*DestroyRing* is a secondary intention because it is obtained from the rule $r_4 : \neg\textit{GoToMordor} \Rightarrow_{\text{BEL}} \neg\textit{DestroyRing}$ and from $+\partial_{\text{INT}}\neg\textit{GoToMordor}$ (again, see Example 4). It should be noted that OBL*DestroyRing* cannot be derived because $r_4 > r_2$, but this just amounts to assuming that the agent is realistic: r_4 is a belief rule whereas r_2 is a deontic rule. In other words, when we have in general that

$$\begin{array}{ll} a \Rightarrow_{\text{OBL}} q & b \Rightarrow_{\text{BEL}} \sim q \\ +\partial_{\text{BEL}} a & +\partial_{\text{INT}} b \end{array}$$

we are doomed to have social agents who cannot be truly social since some of their (primitive) intentions lead to behaviours against what would be otherwise obligatory for the agents. However, this issue is not a matter of a direct conflict between rules for intentions and obligations. Thus, to deal with norm-complying agents in these scenarios and to restore their sociality we are required to change the notion of agent type. We cannot anymore define it in terms of an order of overruling between rules, but we have to focus on how the conflicting literals are derived during the proof. Indeed, this is feasible, but has a high computational cost, and even then we cannot guarantee the sociality of an agent.

4. THE COST OF SOCIAL AGENTS

In this section we investigate the complexity of the defeasible logic for BIO agents where we assume the conversion $c(\text{BEL},\text{OBL})$ and $c(\text{BEL},\text{INT})$ and then we turn our attention to the complexity of social agents. We first introduce some notions to make precise the definition of the issues at hand.

DEFINITION 9. *Let # be one of the proof tags. Given a theory D, $D \vdash \pm\#p$ iff there is a derivation P in D such that for some n $P(n) = \pm\#p$.*

DEFINITION 10. *Given a theory D, the* universe *of D (U^D) is the set of all the atoms occurring in D; the* extension *of D (E^D), is defined as follows:*

$$E^D = (\Delta^+, \Delta^-, \partial^+, \partial^-)$$

where for $X \in \{\text{BEL}, \text{INT}, \text{OBL}\}$

$$\Delta^+ = \{Xl : D \vdash +\Delta_X l\};$$
$$\Delta^- = \{Xl : D \vdash -\Delta_X l\};$$
$$\partial^+ = \{Xl : D \vdash +\partial_X l\};$$
$$\partial^- = \{Xl : D \vdash -\partial_X l\}.$$

Two theories D and D' are *equivalent* if and only if they have the same extension, namely $D \equiv D'$ iff $E^D = E^{D'}$.

We now prove the main theorem about the complexity of our defeasible logic. We show that the logic has linear complexity if we compute the whole set of conclusions, i.e., the extension, of a given theory.

THEOREM 1. *For every theory D, E^D can be computed in time linear to the size of the theory, i.e., $O(|U^D| * |R|)$.*

PROOF. The proof is based on a modification of the algorithm given by Maher [16] to show that propositional defeasible logic has linear complexity.

The main idea of the proof is to build appropriate data structure to implement a series of transformations reducing the complexity of the rules, and where each literal and modal literal is examined only once. The focal point of the transformations is based on the following properties:

- Let $D \vdash +\partial p$ then
$$D \cup \{r : p_1, \ldots, p_n, p \Rightarrow q\} \equiv D \cup \{r : p_1, \ldots, p_n \Rightarrow q\}.$$
- Let $D \vdash -\partial p$ then $D \cup \{r : p_1, \ldots, p_n, p \Rightarrow q\} \equiv D$.

The properties allow us (1) to remove already proved literals from the body of rules and (2) to remove rules which have been discarded.

The algorithm has three phases. (1) A pre-processing phase where we use the transformations given in [2] to transform a theory into an equivalent theory without superiority relation and defeaters; the transformation is linear. (2) A *rule loader* that parses the theory obtained in the first phase and generates the data structure that encodes the theory. (3) The *inference engine* applies transformations to the data structure, where at every step it reduces the complexity of the data structure.

The rule loader builds a data structure as follows: for every atom $\alpha \in U^D$ we create three entries α, INTα and OBLα. Each entry has associated to it a list of hash tables:

For α we have

- $+h$ is a list of (pointers to) rules in R^{BEL} where α appears in the head;
- $-h$ is the list of rules in R^{BEL} where $\sim\alpha$ appears in the head;
- $+b$ is the list of rules in R where α occurs in the body;
- $-b$ is the list of rules in R where $\sim\alpha$ occurs in the body.

For $X\alpha$, $X \in \{\text{INT}, \text{OBL}\}$ we have

- $+h$ is a list of rules in R^X where α appears in the head;
- $-h$ is the list of rules in R^X where $\sim\alpha$ appears in the head;
- $+h^{\mathcal{B}}$ is a list of rules in R^{BEL} where α appears in the head;
- $-h^{\mathcal{B}}$ is a list of rules in R^{BEL} where $\sim\alpha$ appears in the head;
- $+b$ is the list of rules in R where $X\alpha$ occurs in the body;
- $-b$ is the list of rules in R where $X\sim\alpha$ occurs in the body.
- $+b_\sim$ is the list of rules in R where $\sim X\alpha$ occurs in the body;
- $-b_\sim$ is the list of rules in R where $\sim X\sim\alpha$ occurs in the body.

To each rule in R^X, $X \neq$ BEL, we associate a structure consisting of a (modal) literal (the head of the rule) and a set of pointers to the modal literals in the body of the rule, implemented as an hash table; while for belief rules we create the same structure as the other types of rules plus two other structures one for INT and one for OBL, the single pointer refers to the modal literal and the set of pointers corresponds to the literals in the body modalised, respectively, with INT and OBL.

The Inference Engine is based on an extension of the *Delores* algorithm/implementation proposed in [17] as a computational model of Basic Defeasible Logic. In turn

1. It asserts each fact (as an atom) as a conclusion and removes the atom from the rules where the atom occurs positively in the body, and it "deactivates" the rules where either the atom occurs negatively in the body, or incompatible modal literals occur in the body.
2. It scans the list of active rules for rules where the body is empty. It takes head and searches for rule (of the appropriate type) where the head is the negation of the atom or a modal literal incompatible with it. If there are no such rules then, the atom is appended to the list of facts, and removed from the rules
3. It repeats the first step.
4. The algorithm terminates when one of the two steps fails. On termination the algorithm outputs the set of conclusions.[1]

It is immediate to see that the algorithm runs in linear time. Each (modal) atom/literal in a theory is processed exactly once and every time we have to scan the set of rules, thus the complexity of the above algorithm is $O(|U^D| * |R|)$. □

Given the above result it might seem that social agents are computationally feasible. However, as we have seen in Section 3.1 there are situations (let us call them deviant situations) where social agents do not behave as expected. First of all, we have to identify when we have a deviant situation and what are the reasons why we have them, and what kind of control an agent has over them. Here we assume that a deviant situation depends on some primitive intentions of an agent (i.e., intentions given as facts). Since these intentions are independent of the policy the theory describe the only alternative a social agent has is to give up some of them. In the rest of the section we study whether this is possible and what price an agent has to pay to be social. The answer is negative; we will provide a theory that is essentially deviant, and we will show that social agents are (computationally) expensive.

First of all we have to give a precise definition of the problem.

[1] This algorithm outputs ∂^+; ∂^- can be computed by an algorithm similar to this with the "dual actions". For Δ^+ we have just to consider similar constructions where we examine only the first parts of step 1 and 2. Δ^- follows from Δ^+ by taking the dual actions.

Restoring Sociality Problem

INSTANCE:
Let I be a finite set of primitive intentions, OBLp a primary obligation, and D a theory such that $I \subseteq F$, $D \vdash -\partial_{\text{OBL}} p$, $D \vdash -\Sigma_{\text{OBL}} \sim p$, $D \vdash +\partial_{\text{INT}} \sim p$, $D \vdash +\Sigma_{\text{OBL}} p$ and $D \vdash -\Sigma_{\text{BEL}} \sim p$.

QUESTION:
Is there a theory D' equal to D apart from containing only a proper subset I' of I instead of I, such that $\forall q$ if $D \vdash +\partial_{\text{OBL}} q$ then $D' \vdash \partial_{\text{OBL}} q$ and $D' \vdash +\partial_{\text{OBL}} p$?

The specification of the problem is meant to formalise the situation we have described in the previous sections. The combination of the proof tags in the specification of the instance is only possible in case there is an applicable deontic rule for p ($+\Sigma_{\text{OBL}} p$) which would be otherwise unchallenged, i.e., there are no deontic rules to support $\sim p$ ($-\Sigma_{\text{OBL}} \sim p$) and there are no reasons to believe the opposite, is defeated, against the sociality of the agent, by the intentionality of $\sim p$ obtained as a consequence of an intention of the agent (this means it has been obtained by converting a belief rule into an intention rule). In other terms a potentially valid obligation is blocked by a consequence of an intentional behaviour.

EXAMPLE 5. Let us the consider the theory consisting of

$$F = \{\text{INT}p, \text{INT}s\}$$
$$R = \{r_1 : p, s \Rightarrow_{\text{BEL}} q \quad r_2 : \Rightarrow_{\text{OBL}} \sim q \quad r_3 : \Rightarrow_{\text{BEL}} s\}$$
$$> = \{r_1 > r_2\}$$

r_1 is a belief rule and so the rule is stronger than the deontic rule r_2. In addition we have that the belief rule is not applicable (i.e., $-\Sigma_{\text{BEL}} q$) since there is no way to prove $+\partial_{\text{BEL}} p$. There are no deontic rules for q, so $-\partial_{\text{OBL}} q$. However, rule r_1 behaves as an intention rule since all its antecedent can be proved as intentions, i.e., $+\partial_{\text{INT}} p$ and $+\partial_{\text{INT}} s$. Hence, since r_1 is stronger than r_2, the derivation of $+\partial_{\text{OBL}} \sim q$ is prevented against the sociality of the agent.

The related decision problem is whether it is possible to avoid the "deviant" behaviour by giving up some primitive intentions, retaining all the (primary) obligations, and maintaining a set of primitive intentions as close as possible to the original set of intentions.

EXAMPLE 5. (CONTINUED). When we examine the theory we notice that both primitive intentions concur to the prevention of the derivation of $+\partial_{\text{OBL}} \sim q$. These intentions are under the control of the agent. The agent has the opportunity to avoid the deviant behaviour if she gives up at least one of her primitive intentions. Accordingly, the agent has three alternatives: to give up INTp, to give up INTs, or to give up both. The first two options minimise the difference between the original theory and the resulting theory.

There could be cases where, no matter what intentions are removed, the theory will result in a deviant situation. The simplest case is where there are intentions that are at the same time primitive and primary.

EXAMPLE 6. Let the theory D be

$$F = \{\text{INT}p\}$$
$$R = \{r_1 : \Rightarrow_{\text{INT}} p \quad r_2 : p \Rightarrow_{\text{BEL}} q \quad r_3 : \Rightarrow_{\text{OBL}} \sim q\}$$
$$> = \{r_2 > r_3\}$$

In this theory we have only one primitive intention and therefore the only way to see whether it is possible to avoid the problem is to give up that intention. However, we have that r_1 is an intention rule for p, and thus we can use it to derive $+\partial_{\text{INT}} p$, which allows r_2 to be used to derive an intention instead of a belief, and consequently to prevent the derivation of an obligation against the sociality of the agent.

Notice that, given the non-monotonic nature of defeasible logic, it is possible that a solution to the problem is given by a superset of the original set of intentions instead of a subset.

EXAMPLE 7. Given a theory D as follows

$$F = \{\text{INT}a, \text{INT}b\}$$
$$R^{\text{BEL}} = \{r_1 : \text{INT}a \Rightarrow_{\text{BEL}} d, \quad r_2 : \text{INT}b \Rightarrow_{\text{BEL}} d,$$
$$r_3 : \text{INT}c \Rightarrow_{\text{BEL}} \sim d, \quad r_4 : d \Rightarrow_{\text{BEL}} e\}$$
$$R^{\text{INT}} = \{r_5 : \Rightarrow_{\text{INT}} a, \quad r_6 \Rightarrow_{\text{INT}} b\}$$
$$R^{\text{OBL}} = \{r_7 : \Rightarrow_{\text{OBL}} \sim e\}$$
$$> = \{r_3 > r_1, r_3 > r_2, r_4 > r_7\}$$

As we have seen in the previous example, throwing away the two primitive intentions is of no avail, they are reinstated by the intention rules r_5 and r_6. However, to block the side effect d of the two intentions we can introduce a further primitive intention, INTc.

If we replace the theory D by a theory D' obtained from D by emptying the set of intention rules, then we have two alternatives to avoid the deviance. The first is to drop both the primitive intentions INTa and INTb, or we can form a new primitive intention INTc. In this case the theory obtained from adding the new intention is, intuitively, more similar to the original theory than the theory obtained from dropping the two primitive intentions.

Variations of the problem can be obtained by changing other parameters of the specification. Some of these can define new types of agents. For example a *pro-active social agent* might try to recover from a deviant situation by changing the raw facts (facts that are neither primitive intentions nor primitive obligations). Thus a pro-active social agent tries to adapt the environment to her goals (intentions). A legalistic social agent, on other the hand, might change the set of primitive obligations, while a cheating social agent might change the rules. However, it is important to realise that all these variations have a structure isomorphic to the specification we discuss in this paper. In addition it is possible to generalise the problem to the case of multiple deviant behaviours.

THEOREM 2. *The Restoring Sociality Problem is NP-complete.*

PROOF. We have to show that the problem is both NP and NP-hard. For the NP part all we have to do is to notice that we can guess a theory, we compute the extension of the theory in linear time (Theorem 1) and then verify in linear time whether the restore conditions are satisfied.

For the NP-hard part we have to map a known NP-complete problem to the Restoring Sociality Problem. Here we use the *knapsack problem* [11, Problem MP9].

Knapsack Problem

INSTANCE:
Given a finite set U, for each $u \in U$ a size $s(u) \in \mathbb{Z}^+$ and a value $v(u) \in \mathbb{Z}^+$, and integer B and K.

QUESTION:
Is there a subset $U' \subseteq U$ such that $\sum_{u \in U'} s(u) \leq B$ and $\sum_{u \in U'} v(u) \geq K$?

The knapsack problem is encoded by a defeasible theory D where R is as follows:

- $\text{INT}load(u) \Rightarrow_{\text{BEL}} load(u)$ for each $u \in U$.
- $\sum_{s(u):D \vdash +\partial_{\text{BEL}} load(u)} s(u) > B \Rightarrow_{\text{INT}} overload$
- $\sum_{s(u):D \vdash +\partial_{\text{BEL}} load(u)} v(u) < K \Rightarrow_{\text{INT}} undervalue$
- $overload \Rightarrow_{\text{BEL}} \neg good$
- $undervalue \Rightarrow_{\text{BEL}} \neg good$
- $\Rightarrow_{\text{OBL}} good$

F is given by the relationship $\text{INT}load(u) \in F$ iff $u \in U'$.

The theory of the above construction has several interesting properties. First of all $D \vdash +\partial_{\text{BEL}} load(u)$ iff $\text{INT}load(u) \in F$, which means $u \in U'$; then $D \vdash +\partial_{\text{OBL}} good$ iff either of the two conditions of the knapsack problem are satisfied; notice that since there are no literals for $\neg load(u)$, the computation of the rule $\text{INT}load(u) \Rightarrow_{\text{BEL}} load(u)$ can be computed independently of the rest of the theory thanks to the modularity of DL [2], thus the sums in the antecedent of the second and third rule can be considered as "facts" in the theory. In case one of the condition of the knapsack problem is not satisfied we have exactly a deviant situation as in the restoring sociality problem. The encoding of the knapsack problem in DL is clearly linear, thus any algorithm that solves the restoring sociality problem in polynomial time will solve the knapsack problem in polynomial time. Therefore the restoring sociality problem is NP-complete. □

5. REVISING SOCIAL AGENTS

A first solution to the complexity of social agents is to avoid conversions. However, we believe, that this is a rather unsatisfactory approach for agents with both internal (intentions) and external (obligations) motivational attitudes. It is not possible to capture the notion of intentionality which is of paramount importance when we deal with agents situated in legal contexts.

A second solution would be to assume that belief rules behaving as intention rules (i.e., obtained from the conversion $c(\text{BEL}, \text{INT})$) are always weaker than deontic rules or belief rules behaving as deontic rules (i.e., where the conversion $c(\text{BEL}, \text{OBL})$ applies). In this case the problem is with theory like

$$\begin{array}{ll} r_1 : a \Rightarrow_{\text{BEL}} q & r_2 : b \Rightarrow_{\text{BEL}} \sim q \\ +\partial_{\text{INT}} a & +\partial_{\text{OBL}} b \\ r_1 > r_2 & \end{array}$$

where r_1 is at the same time stronger and weaker than r_2.

6. SUMMARY

The contribution of this paper is manifold. We extend the analysis of policy-based cognitive agents with the notion of obligation and we argue that in such case side-effects do not endanger the logical analysis but on the contrary are beneficial to explain notions, e.g., intentionality, of paramount importance for agents situated in legal contexts.

Policy based agents are represented in defeasible logic extended with the modalities of belief, intention and obligation. This choice was motivated by the computational feasibility of the logic. We have demonstrated that the logic has linear complexity. As far as we know this is the first result of this kind for cognitive agents.

Finally we have studied the notion of social agent and we have proved that a proper and philosophically sound treatment of this notion leads to an increase of the computational complexity of the problem. Again this is the first result of this kind we are aware of.

7. REFERENCES

[1] G. Antoniou, D. Billington, G. Governatori, and M.J. Maher. A flexible framework for defeasible logics. In *Proc. AAAI-2000*, pages 401–405. AAAI/MIT Press, 2000.

[2] G. Antoniou, D. Billington, G. Governatori, and M.J. Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2(2):255–287, 2001.

[3] N. Bassiliades, G. Antoniou, and I. Vlahavas. DR-DEVICE: A defeasible logic system for the Semantic Web. In H.J. Ohlbach and S. Schaffert, editors, *Proc 2nd PPSWR*, LNCS 3208, pages 134–148. Springer, 2004.

[4] M.E. Bratman, D.J. Israel, and M.E Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.

[5] M.E. Bratman. *Intentions, Plans and Practical Reason*. Harvard University Press, 1987.

[6] J. Broersen, M. Dastani, J. Hulstijn, and L. van der Torre. Goal generation in the BOID architecture. *Cognitive Science Quarterly*, 2(3-4):428–447, 2002.

[7] M. Dastani, G. Governatori, A. Rotolo, and L. van der Torre. Preferences of agents in defeasible logic. In S. Zhang and R. Jarvis, editors, *Proc. Australian AI05*, LNAI 3809, pages 695–704. Springer, 2005.

[8] M. Dastani, G. Governatori, A. Rotolo, and L. van der Torre. Programming cognitive agents in defeasible logic. In G. Sutcliffe and A. Voronkov, editors, *Proc. LPAR 2005,*, LNAI 3835, pages 621–636. Springer, 2005.

[9] F. Dignum. Autonomous agents with norms. *Artificial Intelligence and Law*, 7(1):69–79, 1999.

[10] F. Dignum, D. Morley, L. Sonenberg, and L. Cavedon. Towards socially sophisticated BDI agents. In *Proc 4th ICMAS*, pages 111–118, 2000.

[11] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[12] R. Girle. *Modal Logic and Philosophy*. Acumen, 2000.

[13] G. Governatori, V. Padmanabhan, A. Rotolo, and A. Sattar. A defeasible logic for modelling policy-based intentions and motivational attitudes. *submitted*, 2005.

[14] G. Governatori and A. Rotolo. Defeasible logic: Agency, intention and obligation. In A. Lomuscio and D. Nute, editors, *Deontic Logic in Computer Science*, LNAI 3065, pages 114–128. Springer, 2004.

[15] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.

[16] M.J. Maher. Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming*, 1(6):691–711, 2001.

[17] M.J. Maher, A. Rock, G. Antoniou, D. Billignton, and T. Miller. Efficient defeasible reasoning systems. *International Journal of Artificial Intelligence Tools*, 10(4):483–501, 2001.

[18] D. Nute, editor. *Defeasible Deontic Logic*. Kluwer, 1997.

[19] J. Pitt, editor. *Open Agent Societies*. Wiley, 2005.

[20] A.S. Rao and M.P. Georgeff. Modelling rational agents within a BDI-architecture. In *Proc. KR'91*, pages 473–484. Morgan Kaufmann, 1991.

[21] G. Sartor. *Legal Reasoning: A Cognitive Approach to the Law*. Springer, 2005.

[22] G.H. von Wright. *Norm and Action*. Routledge, 1963.

Failures of the VCG Mechanism in Combinatorial Auctions and Exchanges*

Vincent Conitzer
Carnegie Mellon University
Computer Science Department
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
conitzer@cs.cmu.edu

Tuomas Sandholm
Carnegie Mellon University
Computer Science Department
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
sandholm@cs.cmu.edu

ABSTRACT

The *VCG mechanism* is the canonical method for motivating bidders in combinatorial auctions and exchanges to bid truthfully. We study two related problems concerning the VCG mechanism: the problem of revenue guarantees, and that of collusion. The existence of these problems even in one-item settings is well-known; in this paper, we lay out their full extent in multi-item settings. We study four settings: combinatorial forward auctions with free disposal, combinatorial reverse auctions with free disposal, combinatorial forward (or reverse) auctions without free disposal, and combinatorial exchanges. In each setting, we give an example of how additional bidders (colluders) can make the outcome much worse (less revenue or higher cost) under the VCG mechanism (but not under a first price mechanism); derive necessary and sufficient conditions for such an effective collusion to be possible under the VCG mechanism; and (when nontrivial) study the computational complexity of deciding whether these conditions hold.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*; F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Algorithms, Economics, Theory

*This material is based upon work supported by the National Science Foundation under CAREER Award IRI-9703122, Grant IIS-9800994, ITR IIS-0081246, ITR IIS-0121678, and ITR IIS-0427858, a Sloan Fellowship, and an IBM Ph.D. Fellowship.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Keywords

Combinatorial Auctions and Exchanges, VCG Mechanism

1. INTRODUCTION

Combinatorial markets are important mechanisms for allocating tasks and resources in multiagent systems. In a *combinatorial auction*, there are multiple items for sale, and bidders are allowed to place a bid on a *bundle* of these items rather than just on the individual items. A rapidly growing body of literature is devoted to the study of combinatorial auctions, and, to a lesser extent, variations of it, such as combinatorial reverse auctions (where the auctioneer seeks to procure certain items) and combinatorial exchanges (where bidders can offer goods for sale as well as express demand for goods—even within the same bid). One important problem is the *winner determination problem* [16, 6, 19], which is to label bids as accepted or rejected to maximize the total value of the bids accepted (or, in the case of a reverse auction, to minimize their total value), under the constraint that the corresponding allocation of items does not require more items than are available (or, in the case of a reverse auction, under the constraint that all the desired items are procured).

Another key problem in auctions and exchanges (combinatorial or not) is that in general, the bidders may not bid their true valuations for the goods. The problem with untruthful bidding is that the winner determination algorithm can only base the final allocation of the goods on the reported valuations, and thus the final allocation may not be optimal relative to the bidders' true valuations. Thus, economic efficiency may be lost. However, by changing the payment rule, it is possible to motivate bidders to bid their true valuations. The best-known such payment rule is the Vickrey-Clarke-Groves (VCG) mechanism [20, 5, 9]. If the setting is general enough, given certain requirements, the VCG mechanism (or its generalization to *Groves mechanisms*) is in fact the only mechanism promoting truthful bidding [8, 15, 10, 21].[1] Because of this, the VCG mechanism constitutes the standard approach to promoting truthful bidding, and variations such as anytime VCG mechanisms [13] as well as distributed computation of VCG mechanisms [14] have been proposed.

Unfortunately, there are also many problems with the

[1]It should be noted, however, that when the agents' valuations are more restricted, the characterization of truthful mechanisms becomes less stringent [1, 11, 12, 10, 17, 4].

VCG mechanism [18, 3]. We will focus on two related problems: the VCG mechanism is vulnerable to collusion, and may lead to low revenue/high payment for the auctioneer. It is well-known that these problems occur even in single-item auctions (where the VCG mechanism specializes to the *Vickrey* or second-price sealed-bid auction). However, in the single-item setting, these problems are not as severe. For example, in a Vickrey auction, it is not possible for colluders to obtain the item at a price less than the bid of any other bidder. Additionally, in a Vickrey auction, various types of revenue equivalence with (for example) first-price sealed-bid auctions hold. As we will show, in the multi-item setting these properties do not hold and can be violated to an arbitrary extent. Some isolated examples of such problems with the VCG mechanism in multi-item settings have already been noted in the literature [3, 22, 2] (these will be discussed later in the paper). In contrast, our goal in this paper is to give a *comprehensive* characterization of how severe these problems can be and when these severe problems can occur. For the various variants of combinatorial auctions and exchanges, we study the following single problem that relates both issues under consideration: *Given some of the bids, how bad can the remaining bidders make the outcome?* Informally, "bad" here means that the remaining bidders are paid an inordinately large amount, or pay an inordinately small amount, relative to the goods they receive and/or provide. This is closely related to the problem of making revenue guarantees to the auctioneer. But it is also the collusion problem, if we conceive of the remaining bidders as colluders. (The collusion problem can become more difficult if the collusion is required to be *self-enforcing*. A collusion is self-enforcing when none of the colluders have an incentive to unilaterally deviate from the collusion. We will also study how this extra requirement affects our results.)

As it turns out, our fundamental problem is often computationally hard. Computational hardness here is a double-edged sword. On the one hand, if the problem is hard, collusion may not occur (or to a lesser extent) because the colluders cannot find a beneficial collusion. On the other hand, if the problem is hard, it is difficult to make strong revenue guarantees to the auctioneer. Of course, in either case, the computational hardness may be overcome in practice if the stakes are high enough.

2. THE VCG MECHANISM—REVIEW

All the results in this paper hold even when all bidders are *single-minded*, that is, they bid only on a single bundle of items. (Hence, we do not need to discuss bidding languages.) The VCG mechanism proceeds as follows. First, solve the winner determination problem to maximize the sum of the agents' utilities before payments (the sum of the values of the accepted bids, or, in the case of a reverse auction, the negative thereof). Call this sum of utilities a. Then, to determine winning bidder i's payment, remove that bidder's bid, and see what the maximum sum of the utilities before payments would have been with only the remaining bids. Call this sum of utilities b_i. Winning bidder i must pay $b_i - a + v_i$ where v_i is the value of winning bidder i's bid. Effectively, it is the externality the bidder imposed on the other bidders (before payments). We observe that this payment is negative if the bidder's presence makes the other bidders better off (before payments)—for example, in reverse auctions.

3. COMBINATORIAL (FORWARD) AUCTIONS

In a *combinatorial auction*, there is a set of items $I = \{s_1, s_2, \ldots, s_m\}$ for sale. A bid takes the form $b = (B, v)$, where $B \subseteq I$ and $v \in \mathbb{R}$. The winner determination problem is to label bids as accepted or rejected, to maximize the sum of the values of the accepted bids, under the constraint that no item occurs in more than one accepted bid. (This is assuming *free disposal*: items do not have to be allocated to anyone.)

3.1 Motivating example

(A similar example to the one described in this subsection has been given before [3], and examples of vulnerability to false-name bidding in combinatorial auctions [22] can in fact also be used to demonstrate the basic point. We include this subsection for completeness.) Consider an auction with two items, s_1 and s_2. Suppose we have collected two bids (from different bidders), both $(\{s_1, s_2\}, N)$. If these are the only two bids, one of the bidders will be awarded both the items and, under the VCG mechanism, will have to pay N. However, suppose two more bids (by different bidders) come in: $(\{s_1\}, N+1)$ and $(\{s_2\}, N+1)$. Then these bids will win. Moreover, neither winning bidder will have to pay anything! (This is because a winning bidder's item would simply be thrown away if that winning bidder were removed.)

This example demonstrates a number of issues. First, the addition of more bidders can actually decrease the auctioneer's revenue from an arbitrary amount to 0. Second, the VCG mechanism is not revenue-equivalent to the sealed-bid first-price mechanism in combinatorial auctions, even when all bidders' true valuations are common knowledge[2]—unlike in the single-item case. Third, even when the other bidders by themselves would generate nonnegative revenue for the auctioneer under the VCG mechanism, it is possible that two colluders can bid so as to receive all the items without paying anything.

The following sums up the properties of this example.

PROPOSITION 1. *In a forward auction (even with only 2 items), the following can hold simultaneously: 1. The winning bidders pay nothing under the VCG mechanism; 2. If the winning bids are removed, the remaining bids generate revenue N under the VCG mechanism; 3. If these bids were*

[2]Consider the above example with $N \geq 9$ and suppose that the four bids reflect the bidders' true valuations—since bidding truthfully is a weakly dominant strategy in the VCG mechanism. Running a first-price sealed bid auction in this setting, when all bidders' valuations are common knowledge, will not generate expected revenue less than $\frac{N}{8}$. For suppose the expected revenue is less than this. Then the probability that the revenue is at least $\frac{N}{4}$ must be less than $\frac{1}{2}$ by Markov's inequality. So, bidding $(\{A, B\}, \frac{N}{4})$ will win any bidder both items with probability at least $\frac{1}{2}$, leading to an expected utility of at least $\frac{1}{2}(N - \frac{N}{4}) = \frac{3N}{8}$. Because at most one of the three bidders with valuations $(\{A\}, N+1)$ or $(\{A, B\}, N)$ can win its desired bundle, it follows that at least one of these bidders has a probability of at most $\frac{1}{3}$ of winning its desired bundle, and thus has an expected utility of at most $\frac{N+1}{3}$. Because $N \geq 9$, $\frac{3N}{8} > \frac{N+1}{3}$, so this bidder would be better off bidding $(\{A, B\}, \frac{N}{4})$—contradicting the assumption that we are in equilibrium.

truthful (as we would expect under VCG), then if we had run a first-price sealed-bid auction instead (and the bidders' valuations were common knowledge), any equilibrium would have generated revenue $\Theta(N)$.

3.2 Characterization

We now characterize the settings where, given the noncolluders' bids, the colluders can receive all the items for free.

LEMMA 1. *If the colluders receive all the items at cost 0, then for any positive bid on a bundle B of items by a noncolluder, at least two of the colluders receive an item from B.*

PROOF. Suppose that for some positive bid b on a bundle B by a noncolluder i, one of the colluders c receives all the items in B (and possibly others). Then, in the auction where we remove that colluder's bids, one possible allocation gives every remaining bidder all the goods that bidder received in the original auction; additionally, it gives i all the items in bundle B; and it disposes of all the other items c received in the original auction. With this allocation, the total value of the accepted bids by bidders other than c is at least $v(b)$ more than in the original auction. Because the total value obtained in the new auction is at least the value of this particular allocation, it follows that c imposes a negative externality of at least $v(b)$ on the other bidders, and will pay at least $v(b)$. □

LEMMA 2. *Suppose all the items in the auction can be divided among the colluders in such a way that, for any positive bid on a bundle of items B by a noncolluder, at least two of the colluders receive an item from B. Then the colluders can receive all the items at cost 0.*

PROOF. For the given partition of items among the noncolluders, let each colluder place a bid with an extremely large value on the bundle consisting of the items assigned to him in the partition. (For instance, twice the sum of the values of all noncolluders' bids.) Then, the auction will clear awarding each colluder the items assigned to him by the partition. Moreover, if we remove the bids of one of the colluders, all the remaining colluders' bids will still win—and thus none of the noncolluders' bids will win, because each such bid requires items assigned to at least two colluders by the partition (and at least one of them is still in the auction and wins these items). Thus, each colluder (individually) imposes no externality on the other bidders. □

Combining these two lemmas, we get:

THEOREM 1. *The colluders can receive all the items at cost 0 if and only if it is possible to divide the items among the colluders in such a way that, for any positive bid B by a noncolluder, at least two colluders receive an item from B.*

3.3 Self-enforcing collusion

It turns out that requiring that the collusion is self-enforcing (i.e., no colluder has an incentive to unilaterally deviate) is no harder for the colluders:

THEOREM 2. *Whenever the colluders can receive all the items for free, they can also receive them all for free in a self-enforcing way.*

PROOF. Let each colluder bid on the same bundle as before; but, increase the bid value of each colluder by an amount that exceeds the utility that any colluder can get from any bundle of items. The colluders will continue to receive all the items at a cost of 0. Now, the only reason that a colluder may wish to deviate from this is that the colluder wishes to obtain items outside of the colluder's assigned bundle. However, doing so would prevent one of the other bundles from being awarded to its designated colluder. This would cause a decrease in the total value of bids awarded to bidders other than the deviating colluder that exceeds the utility of the deviating colluder for any bundle, and the deviating colluder would have to pay for this decrease under the VCG mechanism. Therefore, there is no incentive for the colluder to deviate. □

3.4 Complexity

In order to collude in the manner described above, the n colluders must solve the following computational problem.

DEFINITION 1 (DIVIDE-SUBSETS). *Suppose we are given a set I, as well as a collection $R = \{S_1, \ldots, S_q\}$ of subsets of it. We are asked whether I can be partitioned into n parts $T_1, T_2, \ldots, T_n$ so that no subset $S_i \in R$ is contained in one of these parts.*

THEOREM 3. *DIVIDE-SUBSETS is NP-complete, even when $n = 2$.*

PROOF. The problem is technically identical to HYPERGRAPH-2-COLORABILITY, which is known to be NP-complete [7]. □

This hardness result only states that it is hard to identify the *most* beneficial collusion, and one may wonder whether it is perhaps easier to find *some* beneficial collusion. It turns out that the hardness of the former problem implies the hardness of the latter problem: the utility functions of the colluders can always be such that only the most beneficial collusion actually benefits them, in which case the two problems are the same. This observation can also be applied to hardness results presented later in the paper.

4. COMBINATORIAL REVERSE AUCTIONS

In a *combinatorial reverse auction*, there is a set of items $I = \{s_1, s_2, \ldots, s_m\}$ to be procured. A bid takes the form $b = (B, v)$, where $B \subseteq I$ and $v \in \mathbb{R}$. (Here, v represents the value that the bidder must be compensated by in order to provide the goods B.) The winner determination problem is to label bids as accepted or rejected, to minimize the sum of the values of the accepted bids, under the constraint that each item occurs in at least one accepted bid. (This is assuming free disposal.)

4.1 Motivating example

Consider a reverse auction with m items, $s_1, s_2, \ldots, s_m$. Suppose we have collected two bids (from different bidders), both $(\{s_1, s_2, \ldots, s_m\}, N)$. If these are the only two bids, one of the bidders will be chosen to provide all the goods, and, under the VCG mechanism, will be paid N. However, suppose m more bids (by different bidders) come in: $(\{s_1\}, 0), (\{s_2\}, 0), \ldots, (\{s_m\}, 0)$. Then, these m bids will

win. Moreover, each bidder will be paid N under the VCG mechanism. (This is because without this bidder, we would have had to accept one of the original bids.) Thus, the total payment that needs to be made is mN.[3]

Again, this example demonstrates a number of issues. First, the addition of more bidders may actually increase the total amount that the auctioneer needs to pay. Second, the VCG mechanism requires much larger payments than a first-price auction in the case where all bidders' valuations are common knowledge. (The first-price mechanism will not require a total payment of more than N for these valuations in any pure-strategy equilibrium.[4][5]) Third, even when the other bidders by themselves would allow the auctioneer to procure the items at a low cost under the VCG mechanism, it is possible for m colluders to get paid m times as much for all the items.

The following sums up the properties of this example.

PROPOSITION 2. *In a reverse auction, the following can hold simultaneously: 1. The winning bidders are paid mN under the VCG mechanism; 2. If the winning bids are removed, the remaining bids allow the auctioneer to procure everything at a cost of only N under the VCG mechanism; 3. If these bids were truthful (as we would expect under VCG), then if we had run a first-price sealed-bid reverse auction instead (and the bidders' valuations were common knowledge), any equilibrium in pure strategies would have required total payment of at most N. (However, there are also mixed-strategy equilibria with arbitrarily large expected total payment.)*

4.2 Characterization

Letting N be the sum of the values of the accepted bids when all the colluders' bids are taken out, it is clear that no colluder can be paid more than N. (With the colluder's bid, the sum of the values of others' accepted bids is still at least 0; without it, it can be at most N, because in the worst case the auctioneer can accept the bids that would be accepted if none of the colluders are present.) In this subsection, we will identify a necessary and sufficient condition for the colluders to be able to each receive N.

LEMMA 3. *If a colluder receives N, then the items that it has to provide cannot be covered by a subset of the noncolluders' bids with cost less than N.*

PROOF. If they could be covered by such a set, we could simply accept this set of bids (including those that were accepted already) rather than the colluder's bid, and increase the total cost by less than N. Thus, the colluder's VCG payment is less than N. □

Thus, in order for each of the n colluders to be able to receive N, it is necessary that there exist n disjoint subsets of the items, each of which cannot be covered with a subset of the noncolluders' bids with total value less than N. The next lemma shows that this condition is also sufficient.

LEMMA 4. *If there are n disjoint sets of items $R_1, \ldots, R_n$, each of which cannot be covered by a subset of the noncolluders' bids with cost less than N, then n colluders can be paid N each.*

PROOF. Let colluder i (for $i < n$) bid $(R_i, 0)$, and let colluder n bid $(R_n \cup (S - \bigcup_i R_i), 0)$. Then the total cost of all accepted bids with all the colluders is 0; but when one colluder is omitted, the items it won cannot be covered at a cost less than N (because its bid contained one of the R_i). Thus, each colluder's VCG payment is N. □

The next lemma shows that the necessary and sufficient condition above is equivalent to being able to *partition* all the items into n sets, so that no element of the partition can be covered by a subset of the noncolluders' bids with total value less than N. That is, we can restrict our attention to the case where the subsets exhaust all the items.

LEMMA 5. *The condition of Lemma 4 is satisfied if and only if it is possible to partition the items into $T_1, \ldots, T_n$ such that no T_i can be covered by a subset of the noncolluders' bids with cost less than N.*

PROOF. The "if" part is trivial: given T_i that satisfy the condition of this lemma, simply let $R_i = T_i$. For the "only if" part, given R_i that satisfy the condition of Lemma 4, let $T_i = R_i$ for $i < n$, and $T_n = R_n \cup (S - \bigcup_i R_i)$. We observe that this last set can also not be covered at a cost of less than N because it contains R_n. □

Combining all the lemmas, we get:

THEOREM 4. *The n colluders can receive a payment of N each (simultaneously), where N is the sum of the values of the accepted bids when all the colluders' bids are removed, if and only if it is possible to partition the items into $T_1, \ldots, T_n$ such that no T_i can be covered by a subset of the noncolluders' bids with cost less than N.*

4.3 Self-enforcing collusion

Unlike the case of combinatorial forward auctions, in reverse auctions, a stronger condition is required if the collusion is also required to be self-enforcing.

[3]Similar examples have been discovered in the context of purchasing paths in a graph [2]. However, in that setting, the buyer does not seek to procure all of the items, and hence the examples cannot be applied directly to combinatorial reverse auctions.

[4]Consider the above example and suppose that the $n+2$ bids reflect the bidders' true valuations—since bidding truthfully is a weakly dominant strategy in the VCG mechanism. Supposing that a pure-strategy equilibrium is being played, let the total payment to be made in this equilibrium be π. (We observe that the final allocation can still be uncertain, e.g. if there is a random tie-breaking rule.) Suppose $\pi > N$. Then, the expected utility for either one of the bidders interested in providing the whole bundle can never exceed $\pi - N$ (because the bidder will be paid 0 whenever none of its bids are accepted, and providing any items at all will cost it N). Moreover, it is not possible for both of these bidders to simultaneously have an expected utility of $\pi - N$ (as this would mean that both are paid π with certainty, contrary to the fact that the total payment is π). It follows at least one has an expected utility of $\pi - N - \epsilon$ for some $\epsilon > 0$. But then this bidder would be better off bidding $\pi - \frac{\epsilon}{2}$ for the whole bundle, which would be accepted with certainty and give an expected utility of $\pi - N - \frac{\epsilon}{2}$. It follows that the total payment in a pure-strategy equilibrium cannot exceed N.

[5]Perhaps surprisingly, the first-price combinatorial reverse auction for this example (with commonly known true valuations corresponding to the given bids) actually has mixed-strategy equilibria with arbitrarily high expected payments. We omit the proof because of space constraint.

THEOREM 5. *The n colluders can receive a payment of N each (simultaneously), where N is the sum of the values of the accepted bids when all the colluders' bids are removed, if and only if it is possible to partition the items into $T_1, \ldots, T_n$ such that 1) no T_i can be covered by a subset of the noncolluders' bids with cost less than N, 2) for no colluder i, the following holds: there exists a subset $T_i' \subseteq T_i$ such that T_i' can be covered by a set of noncolluders' bids with total cost less than $v_i(T_i) - v_i(T_i - T_i')$ (the marginal savings to agent i of not having to provide T_i').*

PROOF. Omitted due to space constraint. □

4.4 Complexity

In order to collude in the manner described above, the n colluders must solve the following computational problem.

DEFINITION 2 (CRITICAL-PARTITION). *We are given a set of items I, a collection of bids (S_i, v_i) where $S_i \subseteq I$ and $v_i \in \mathbb{R}$, and a number n. Say that the* cost *of a subset of these bids is the sum of their v_i; and that the cost $c(T)$ of a subset $T \subseteq I$ is the lowest cost of any subset of the bids whose S_i cover T. We are asked whether there exists a partition of I into n disjoint subsets $T_1, T_2, \ldots, T_n$, such that for any $1 \leq i \leq n$, $c(T_i) = c(I)$.*

THEOREM 6. *Even when the bids are so that a partition $T_1, \ldots, T_n$ is a solution if and only if no set $I - T_i$ covers all items in a bid, CRITICAL-PARTITION is NP-complete (even with $n = 2$).*

PROOF. The problem is in NP in this case because given a partition $T_1, \ldots, T_n$, it is easy to check if any set $I - T_i$ covers all items in a bid.

To show NP-hardness, we reduce an arbitrary NAESAT[6] instance (given by a set of clauses C over a set of variables V, with each variable occurring at most once in any clause) to the following CRITICAL-PARTITION instance with $n = 2$ (where we are trying to partition into T_1 and T_2). Let I be as follows. For every variable $v \in V$, there are two items labeled s_{+v} and s_{-v}. Let the bids be as follows. For every variable $v \in V$, there is a bid $(\{s_{+v}, s_{-v}\}, 2)$. For every clause $c \in C$, there are two bids $(\{s_l : l \in c\}, 2m_c - 1)$ and $(\{s_l : -l \in c\}, 2m_c - 1)$ where m_c is the number of literals occurring in c.

First we show that this instance satisfies the condition that a partition $T_1, \ldots, T_n$ is a solution if and only if no set $I - T_i$ covers all items in a bid. First, we observe that $c(I) = |I|$ (we can use all the bids of the form $(\{s_{+v}, s_{-v}\}, 2)$, getting a per-item cost of 1; no other bid gives a lower per-item cost).

Now, if some set $I - T_i$ covers all the items in a bid of the form $(\{s_{+v}, s_{-v}\}, 2)$, then $c(T_i) \leq 2|I| - 2$ (because we can simply omit this bid from the solution for all the items). If some set $I - T_i$ covers all the items in a bid of the form $(\{s_l : l \in c\}, 2m_c - 1)$, then $c(T_i) = |I| - 1$. (This is because we can now accept the "complement" bid $(\{s_l : -l \in c\}, 2m_c - 1)$, and we will have covered all the items s_{+v} and s_{-v} in T_i such that v occurs in c (precisely $2m_c$ items, because variables do not reoccur within a clause); for any other item s_{+v} or s_{-v}, we can accept the bid $(\{s_{+v}, s_{-v}\}, 2)$, and we need to accept at most $|V| - m_c$ such bids, leading to a total cost of $2m_c - 1 + 2(|V| - m_c) = |I| - 1$.)

On the other hand, suppose there is no set $I - T_i$ that covers all the items in a bid. Then, either T_i must include at precisely one of s_v and s_{-v}. (Otherwise one T_i would include neither and $I - T_i$ would cover all items in the bid $(\{s_{+v}, s_{-v}\}, 2)$.) Thus, when we are trying to cover T_i, covering items in it with bids of the form $(\{s_{+v}, s_{-v}\}, 2)$ would result in a per-item cost of 2. On the other hand, covering items in it with bids of the form $(\{s_l : l \in c\}, 2m_c - 1)$ or $(\{s_l : -l \in c\}, 2m_c - 1)$ would result in a per-item cost of at least $\frac{2m_c - 1}{m_c - 1} > 2$ (because at most $m_c - 1$ of the m_c items in the bid can be in T_i, otherwise T_i would cover all the items in the bid; but $T_i = I - T_{3-i}$ which by assumption does not cover all the items in any bid). It follows that $C(T_i) = 2|V| = |I| = c(I)$.

Now we show that the two instances are equivalent. First suppose there exists a solution to the NAESAT instance. Then partition the elements as $T_1 = \{s_l : l = \textit{true}\}$ and $T_2 = \{s_l : l = \textit{false}\}$, according to this solution. Clearly neither of $I - T_i = T_{3-i}$ covers a bid of the form $(\{s_{+v}, s_{-v}\}, 2)$. Also, because no clause has all its literals set to the same value (we have a NAESAT solution), the items in a corresponding bid $(\{s_l : l \in c\}, 2m_c - 1)$ or $(\{s_l : -l \in c\}, 2m_c - 1)$ are not all in the same set. By the previously proved property, it follows that this partition is a solution to the CRITICAL-PARTITION instance.

On the other hand, suppose that there exists a solution to the CRITICAL-PARTITION instance. Then label a literal *true* if $s_l \in T_1$, and *false* otherwise. By the previously proved property, because $(\{s_{+v}, s_{-v}\}, 2)$ is a bid, only one of s_{+v} and s_{-v} can be in $T_1 = I - T_2$, so this provides a consistent setting of the literals. Additionally, because $(\{s_l : l \in c\}, 2m_c - 1)$ is a bid, not all the s_l in that bid can be in $T_1 = I - T_2$. It follows that some of the literals $l \in c$ are set to *false*. Similarly, not all the s_l in that bid can be in $T_2 = I - T_1$, so some of the literals $l \in c$ are set to *true*. It follows that this assignment of truth values to variables is a solution to the NAESAT instance. □

5. COMBINATORIAL FORWARD (OR REVERSE) AUCTIONS WITHOUT FREE DISPOSAL

A combinatorial forward auction *without free disposal* is exactly the same as one with free disposal, with the exception that every item must be allocated to some bidder. Here, bids with a *negative* value may still be useful, as they allow us to remove some of the items—which may allow us to accept better bids for the remaining items.

Similarly, a combinatorial reverse auction without free disposal is exactly the same as one with free disposal, with the exception that no additional items can be procured. Here, bids with a *negative* value may occur—the (nondisposable) item may be a liability to the bidder.

In both cases, we seek to identify a subset of the bids that constitutes an exact cover of the items (no item covered more than once), and to maximize the bidders' total utility under this constraint. Therefore, the settings are technically identical, and in the rest of this section, we can restrict our attention to forward auctions without free disposal.

[6]The goal in NAESAT is to assign truth values to all variables in such a way that there is no clause with all its literals set to *true*, and no clause with all its literals set to *false*.

5.1 Motivating example

Consider a forward auction with two nondisposable items, s_1 and s_2. Suppose we have collected two bids (from different bidders), both $(\{s_1, s_2\}, N)$. If these are the only two bids, one of the bidders will be awarded both the items and, under the VCG mechanism, will have to pay N. However, suppose two more bids (by different bidders) come in: $(\{s_1\}, N+M)$ and $(\{s_2\}, N+M)$, with $M > 0$. Then these bids will win. Moreover, because without free disposal, we cannot accept either of these bids without the other, each of these bidders will be *paid* M under the VCG mechanism!

Again, this example demonstrates a number of issues. First, additional bidders may change the auctioneer's revenue from an arbitrarily large positive amount to an arbitrarily large negative amount (an arbitrarily large cost). Second, the VCG mechanism may require arbitrarily large payments from the auctioneer even in cases where a first-price auction would actually generate revenue for the auctioneer, in the case where all bidders' valuations are common knowledge. (The first-price mechanism will generate a revenue of at least N for these valuations in any pure-strategy equilibrium.[7][8]) Third, even when the other bidders by themselves would generate positive revenue for the auctioneer under the VCG mechanism, it is possible that two colluders can make the auctioneer pay each of them an arbitrarily large amount.

The following sums up the properties of this example.

PROPOSITION 3. *In a forward auction without free disposal (even with only two items), the following can hold simultaneously: 1. Each winning bidder is paid an arbitrary amount M under the VCG mechanism (where M depends only on the winners' bids); 2. If the winning bids are removed, the remaining bids actually generate revenue N to the auctioneer under the VCG mechanism; 3. If these bids were truthful (as we would expect under VCG), then if we had run a first-price sealed-bid auction instead (and the bidders' valuations were common knowledge), any equilibrium in pure strategies would have generated revenue N. (However, there are mixed-strategy equilibria with arbitrarily large cost to the auctioneer.)*

[7]Consider the above example and suppose that the four bids reflect the bidders' true valuations—since bidding truthfully is a weakly dominant strategy in the VCG mechanism. Supposing that a pure-strategy equilibrium is being played, let the total revenue to the auctioneer be ρ, where ρ is possibly negative. (We observe that the final allocation can still be uncertain, e.g. if there is a random tie-breaking rule.) Suppose $\rho < N$. Then the expected utility for either of the bidders interested in providing the whole bundle is at most $N - \rho$. (If the bidder receives a singleton item, its utility is $-\infty$; if it receives nothing, its utility is 0; if it receives both items, its utility is $N - \rho$.) Moreover, it is not possible for both of these bidders to both have an expected utility of $N - \rho$, as this would mean they both receive both items with probability 1. It follows that at least one of them has an expected utility of $N - \rho - \epsilon$ where $\epsilon > 0$. But then this bidder would be better off bidding $\rho + \frac{\epsilon}{2}$, as this bid would be accepted with certainty and give an expected utility of $N - \rho - \frac{\epsilon}{2}$. It follows that the expected revenue in a pure-strategy equilibrium cannot be less than N.

[8]Similarly to the case of the combinatorial reverse auction with free disposal, there are mixed-strategy equilibria in the first-price auction where the auctioneer is forced to make arbitrarily large payments—we omit the proof because of space constraint.

5.2 Characterization

In this subsection, we will identify a necessary and sufficient condition for the colluders to be able to each receive an arbitrary amount. Let $v(b)$ denote the value of bid b.

LEMMA 6. *If each colluder receives a payment of more than $2\sum_d |v(b_d)|$ (where d ranges over the noncolluders), then for each colluder c, the set of all items awarded to either that colluder or a noncolluder (that is, $s_c \cup \bigcup_d s_d$, where s_b is the set of items awarded to bidder b and d ranges over the noncolluders) cannot be covered exactly with bids from the noncolluders.*

PROOF. Say that the sum of the values of accepted noncolluder bids is D (which may be negative). Suppose that for one colluder c, the set of all items awarded to either her or a noncolluder (that is, $s_c \cup \bigcup_d s_d$) can be covered by a set of noncolluder bids of combined value C (which may be negative). Then removing colluder c can make the allocation at most $D - C$ worse to the other bidders (relative to their reported valuations), because we could simply accept the bids of combined value C and no longer accept the bids of combined value D, and keep the rest of the allocation the same. Thus, under VCG, that colluder should be rewarded at most $D - C \leq 2\sum_d |v(b_d)|$. □

Thus, in order for each colluder to be able to receive an arbitrarily large payment, it is necessary that there are n disjoint subsets of the items such that no such subset taken together with the remaining items can be covered exactly by the noncolluders' bids. Also, the set of remaining items must be exactly coverable by the noncolluders' bids (otherwise we cannot accept all the colluders' bids). The next lemma shows that this condition is also sufficient.

LEMMA 7. *If it is possible to partition the items into $R_1, \ldots, R_n, R_{n+1}$ such that for no $1 \leq i \leq n$, $R_i \cup R_{n+1}$ can be covered exactly with bids from the noncolluders; and such that R_{n+1} can be covered exactly with bids from the noncolluders; then for any $M > 0$, n colluders can place additional bids such that each of them receives at least M.*

PROOF. Let colluder i place a bid $(R_i, M + 3\sum_d |v(b_d)|)$ (where d ranges over the noncolluders). All these bids will be accepted, because it is possible to do so by also accepting the noncolluder bids that cover R_{n+1} exactly; and these noncolluder bids will have a combined value of at least $-\sum_d |v(b_d)|$, so that the sum of the values of all accepted bids is at least $(3n-1)\sum_d |v(b_d)| + nM$. (We observe that if we do not accept all of the colluder bids, the sum of the values of all accepted bids is at most $(3(n-1)+1)\sum_d |v(b_d)| + (n-1)M = (3n-2)\sum_d |v(b_d)| + (n-1)M$, which is less.) Now, if the bid of colluder i is removed, it is no longer possible to accept all the remaining $n-1$ colluder bids, because $R_i \cup R_{n+1}$ cannot be covered exactly with noncolluder bids. It follows that the total value of all accepted bids when i's bid is removed can be at most $(3(n-2)+1)\sum_d |v(b_d)| + (n-2)M$. When i's bid is not omitted, the sum of the values of all accepted bids other than i's is at least $(3(n-1)-1)\sum_d |v(b_d)| + (n-1)M$. Subtracting the former quantity from this, we get that the VCG payment to i is at least $\sum_d |v(b_d)| + M$. □

The next lemma shows that the necessary and sufficient condition above is equivalent to being able to partition all the items into n sets, so that no element of the partition can be covered exactly by a subset of the noncolluders' bids. That is, we can restrict our attention to the case where $R_{n+1} = \emptyset$.

LEMMA 8. *The condition of Lemma 7 is satisfied if and only if the items can be partitioned into $T_1, \ldots, T_n$ such that no T_i can be covered exactly with bids from the noncolluders.*

PROOF. For the "if" part: given T_i that satisfy the condition of this lemma, let $R_i = T_i$ for $i \leq n$, and $R_{n+1} = \emptyset$. Then no $R_i \cup R_{n+1} = T_i$ can be covered exactly with bids from the noncolluders, and $R_{n+1} = \emptyset$ can trivially be covered exactly with noncolluder bids. For the "only if" part: given R_i that satisfy the condition of Lemma 7, let $T_i = R_i$ for $i < n$, and let $T_n = R_n \cup R_{n+1}$. That T_n cannot be covered exactly by noncolluder bids now follows directly from the conditions of Lemma 7. But also, no T_i with $i < n$ can be covered exactly: because if it could, then we could cover $R_i \cup R_{n+1} = T_i \cup R_{n+1}$ using the bids that cover T_i exactly together with the bids that cover R_{n+1} exactly (which exist by the conditions of Lemma 7). □

Combining all the lemmas, we get:

THEOREM 7. *The n colluders can receive a payment of at least M each (simultaneously), where M is an arbitrarily large number, if and only if it is possible to partition the items into $T_1, \ldots, T_n$ such that no T_i can be covered exactly with bids from the noncolluders.*

5.3 Self-enforcing collusion

Again, a stronger condition is required if the collusion is also required to be self-enforcing.

THEOREM 8. *The n colluders can receive a payment of at least M each (simultaneously), where M is an arbitrarily large number, if and only if it is possible to partition the items into $T_1, \ldots, T_n$ such that 1) no T_i can be covered exactly with noncolluder bids, 2) for no colluder i, the following holds: there exists a subset $T_i' \subseteq T_i$ such that T_i' can be covered by a set of noncolluders' bids with total value greater than $v_i(T_i) - v_i(T_i - T_i')$ (the marginal value to agent i of receiving T_i').*

PROOF. Omitted due to space constraint. □

5.4 Complexity

In order to collude in the manner described above, the n colluders must solve the following computational problem.

DEFINITION 3 (COVERLESS-PARTITION). *We are given a set I and a collection of subsets $S_1, S_2, \ldots, S_q \subseteq I$. We are asked whether there is a partition of I into subsets $T_1, T_2, \ldots T_n \subseteq I$ such that no T_i can be covered exactly by some of the S_i.*

THEOREM 9. *Even if there is a singleton S_i for all but two elements a and b, and $n = 2$, COVERLESS-PARTITION is NP-complete.*

PROOF. Omitted due to space constraint. □

5.5 An easier collusion problem

So far in this section, we have formulated the collusion problem so that *each* colluder should receive M, where M is an arbitrary amount. An easier problem for the colluders is to make sure that *together*, they receive M, where M is an arbitrary amount. Such a collusion may be less stable (because some of the colluders may be receiving very little). Nevertheless, as we will show, this type of collusion is possible whenever a weak (and easily verified, given the noncolluders' bids) condition holds: at least one item has no singleton bid on it. (A singleton bid is a bid on only one item.) We first show that this condition is necessary.

LEMMA 9. *If at least one colluder receives a payment of more than $\sum_d |v(b_d)|$ (where d ranges over the noncolluders), then there is at least one item s on which no noncolluder places a singleton bid.*

PROOF. If each item has a singleton noncolluder bid placed on it, then when we remove a colluder's bid, we can simply cover all the items in it with singleton bids (with a combined value of at least $-\sum_d |v(b_d)|$), and leave the rest of the allocation unchanged. It follows that the VCG payment to the colluder can be at most $\sum_d |v(b_d)|$). □

We now show that the condition is sufficient.

LEMMA 10. *If there is at least one item s on which no noncolluder places a singleton bid, then if one colluder bids $(\{s\}, 0)$, and the other colluder bids $(I - \{s\}, M + 2\sum_d |v(b_d)|)$ (for $M > 0$), then the total payment to the colluders is at least M.*

PROOF. The colluders' bids will be the only accepted ones (because colluder 2's bid has a greater value than all other bids combined). If we removed colluder 2's bid, the total value of the accepted bids would be at most $\sum_d |v(b_d)|$, so colluder 2 will pay at most this much under the VCG mechanism. If we removed colluder 1's bid, colluder 2's bid could no longer be accepted (because $\{s\}$ cannot be covered by itself), and thus the total value of the accepted bids could be at most $\sum_d |v(b_d)|$). It follows that colluder 1 is paid at least $M + \sum_d |v(b_d)|$). So the total payment to the colluders is at least M □

Combining the two lemmas, we get the desired result:

THEOREM 10. *Two (or more) colluders can receive a total payment of M, where M is an arbitrarily large number, if and only if there is at least one item that has no singleton bid placed on it by a noncolluder.*

6. COMBINATORIAL EXCHANGES

In a *combinatorial exchange*, there is a set of items $I = \{s_1, s_2, \ldots, s_m\}$ that can be traded. A bid takes the form $b = (\lambda_1, \ldots, \lambda_m, v)$, where $\lambda_1, \ldots, \lambda_m, v \in \mathbb{R}$ (possibly negative). (Each λ_i is the number of units of the ith item that the bidder seeks to procure, and v is how much the bidder is willing to pay.) The winner determination problem is to label bids as accepted or rejected, under the constraint that the sum of the accepted vectors has its first m entries ≤ 0, to

maximize the last entry of the sum of the accepted vectors. (This is assuming free disposal.) We will also use the notation $(\{(s_{i_1}, \lambda_{i_1}), (s_{i_2}, \lambda_{i_2}), \ldots, (s_{i_k}, \lambda_{i_k})\}, v)$ for representing a bid in which λ_{i_j} units of item s_{i_j} are demanded (and 0 units of each item that is not mentioned).

6.1 Characterization

In a combinatorial exchange with at least two items s_1 and s_2, let q_1 (respectively, q_2) be the total number of units of s_1 (respectively, s_2) offered for sale in bids so far (by noncolluders). Now consider the following two bids (by colluders): $(\{(s_1, q_1 + 1), (s_2, -q_2 - 1)\}, M + \sum_d |v(b_d)|)$ and $(\{(s_1, -q_1 - 1), (s_2, q_2 + 1)\}, M + \sum_d |v(b_d)|)$, where $M > 0$ and d ranges over the original (noncolluding) bids. Both these bids will be accepted (for otherwise, the total value of the accepted bids could be at most $M + 2\sum_d |v(b_d)| < 2(M + \sum_d |v(b_d)|)$). Moreover, if we remove one of these two bids, the other cannot be accepted (because its demand cannot be met), so the total value of the accepted bids can be at most $\sum_d |v(b_d)|$). It follows that the VCG payment to each of these two bidders is at least M. This proves the following:

THEOREM 11. *In a combinatorial exchange with at least two items, for any set of bids by noncolluders, two colluders can place bids so that each of them will receive at least M, where M is an arbitrary amount. Moreover, each one receives exactly the items that the other provides, so that their net contribution in terms of items is nothing.*

7. CONCLUSIONS

The VCG mechanism is the canonical mechanism for motivating the bidders to bid truthfully in combinatorial auctions and exchanges; if the setting is general enough, under some requirements, it is the only one. Unfortunately, it also introduces many problems. In this paper, we focused on the related problems of revenue guarantees and bidder collusion. We showed that these problems can be much worse in combinatorial auctions and exchanges than in single-item settings. We gave necessary and sufficient conditions for when these highly undesirable scenarios can occur, and showed how computationally hard it is to verify these conditions. One direction for future research is to characterize when similar but less severe failures of the VCG mechanism can occur. A more interesting (and difficult) direction is to find (possibly untruthful) mechanisms (perhaps for restricted settings) that do not suffer from these problems.

8. REFERENCES

[1] A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *FOCS*, 2001.

[2] A. Archer and E. Tardos. Frugal path mechanisms. In *SODA*, 991–999, 2002.

[3] L. M. Ausubel and P. Milgrom. The lovely but lonely Vickrey auction. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*, chapter 1. MIT Press, 2006.

[4] M. Babaioff, R. Lavi, and E. Pavlov. Mechanism design for single-value domains. In *AAAI*, 2005.

[5] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[6] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *IJCAI*, 548–553, 1999.

[7] M. Garey and D. Johnson. *Computers and Intractability*. W. H. Freeman and Company, 1979.

[8] J. Green and J.-J. Laffont. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica*, 45:427–438, 1977.

[9] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[10] R. Lavi, A. Mu'Alem, and N. Nisan. Towards a characterization of truthful combinatorial auctions. In *FOCS*, 574–583, 2003.

[11] D. Lehmann, L. I. O'Callaghan, and Y. Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.

[12] A. Mu'alem and N. Nisan. Truthful approximate mechanisms for restricted combinatorial auctions. In *AAAI*, 379–384, 2002.

[13] D. Parkes and G. Schoenebeck. GROWRANGE: Anytime VCG-based mechanisms. In *AAAI*, 34–41, 2004.

[14] D. Parkes and J. Shneidman. Distributed implementations of generalized Vickrey-Clarke-Groves auctions. In *AAMAS*, 261–268, 2004.

[15] K. Roberts. The characterization of implementable social choice rules. In J.-J. Laffont, editor, *Aggregation and Revelation of Preferences*. North-Holland Publishing Company, 1979.

[16] M. Rothkopf, A. Pekeč, and R. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[17] M. E. Saks and L. Yu. Weak monotonicity suffices for truthfulness on convex domains. In *ACM-EC*, 286–293, 2005.

[18] T. Sandholm. Issues in computational Vickrey auctions. *International Journal of Electronic Commerce*, 4(3):107–129, 2000. Early version appeared in ICMAS-96.

[19] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, Jan. 2002.

[20] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[21] M. Yokoo. The characterization of strategy/false-name proof combinatorial auction protocols: Price-oriented, rationing-free protocol. In *IJCAI*, 733–742, 2003.

[22] M. Yokoo, Y. Sakurai, and S. Matsubara. Robust combinatorial auction protocol against false-name bids. *Artificial Intelligence*, 130(2), 2004.

Agent Interaction in Distributed POMDPs and its Implications on Complexity

Jiaying Shen Raphen Becker Victor Lesser
Department of Computer Science
University of Massachusetts
Amherst, MA 01003-4610
{jyshen, raphen, lesser}@cs.umass.edu

ABSTRACT

The ability to coordinate effectively is critical for agents to accomplish their goals in a multi-agent system. A number of researchers have modeled the coordination problem for multi-agent systems using decision theory. The most general models have proven to be extremely complex to solve optimally (NEXP-complete). Some of the more restricted models have been much more tractable, though still difficult (NP-complete). What is missing is an understanding about why some models are much easier than others. This work fills this gap by providing a condition that distinguishes between problems in NP and those strictly harder than NP. This condition relates to the quantity of information each agent has about the others, and whether this information can be represented in a succinct way. We show that the class of problems that satisfy this condition is NP-complete. We illustrate this idea with two interaction protocols that satisfy the condition. For those problems that do not satisfy this condition we demonstrate how our theoretical results can be used to generate an NP approximation of the original problem.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

DESIGN, THEORY

Keywords

Complexity, Interaction, Distributed POMDP

1. INTRODUCTION

In a multi-agent system, the agents do not act alone and must take into account the behaviors of the other agents in order to coordinate effectively and achieve their goals. Recently, multi-agent coordination problems have received attention from the decision theory community, and several distributed POMDP frameworks have been built to model these problems. These models assume that the agents have partial or noisy observations about the current world state, and that communication between the agents may have a cost. This partial observability of the world state leads to an increased complexity of NEXP-complete [4, 12]. However, two subclasses of distributed POMDPs have been identified whose complexity is much more tractable than the general case. One of them is the class of distributed POMDPs where the agents interact with each other only through a global reward function while their local transitions and observations remain independent [3]. In the other class of problems agents are allowed to communicate with each other, however, when they do they are required to transmit sufficient information such that all agents know the global state [6, 1]. Both subclasses have been proven to be NP-complete [7, 3].

The focus of this paper is to quantify the characteristics of a class of multi-agent coordination problems that determines its complexity. Our key result is that the complexity of the problem depends on the amount of important information each agent has about the other agents, and whether this information can be represented in a succinct way. Information is important if knowing it could allow the agents to achieve a higher expected reward, and by succinct we mean that the set of all possible important information the agents could acquire is polynomial in the number of states. We prove that this criteria is both sufficient and necessary for the class of problems to be NP-complete. We illustrate this idea with two examples from the literature and prove that both have this property.

Our goal in this paper is not to introduce new models or algorithms, but to change the way people view interactions between agents in the context of distributed POMDPs. Multi-agent researchers have long intuitively understood that the interaction between the agents is the cause of their high complexity. The theoretical results of this paper are significant in that they both formally justify this intuition as well as explain how the interaction affects the complexity. This new understanding of interaction and its relationship to complexity will help us to identify new classes of multi-agent systems with a lower complexity.

The significance of this theoretical result also has a more practical side. Most multi-agent systems are provably harder than NP and solving them optimally is very difficult. Much work has been put into developing good algorithms for approximating these problems. This work provides theoretical guidance in understanding how the approximations in a model limit the search space and reduce the complexity.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

We demonstrate this on two problems that does not meet our condition by providing two approximations that impose additional restrictions on the interactions among the agents and reduce the complexity to no harder than NP.

There is existing work that looks at different types of interactions in distributed POMDPs. Synchronizing communication in an otherwise transition and observation independent DEC-MDP has been studied most extensively [6, 1]. Very little research has been done studying the transfer of only partial observation history between agents. Shen et al. [13] studies the problem of what to communicate in a Distributed Sensor Interpretation setting, where at every step only part of the local state is transferred between the agents. Some researchers [8, 5] have studied the case where the agents send their policies or actions to each other. In a sense, exchanging actions or policies between the agents accomplishes the same goal as transferring partial observation history. Both of them change each agent's belief of the global state and/or the other agent's local view of the global state without completely synchronizing the global views. The interaction history in these problems contains information that is not polynomial in the number of states and therefore they are harder than NP.

This work builds on previous work in the distributed POMDP field. Xuan and Lesser [14] proposed one of the first decentralized extensions to the single agent MDP, in which an agent has its local state in addition to the global state. Several more general distributed POMDP models have been independently developed and can be grouped into two categories. Partially Observable Identical Payoff Stochastic Games (POIPSG) [11], Decentralized Partially Observable Markov Decision Processes (DEC-POMDP) and Decentralized Markov Decision Processes (DEC-MDP) [4] do not explicitly model the communication actions but instead view them as a part of the normal action space. In contrast, Communicative Multi-agent Team Decision Problems (COM-MTDP) [12] and DEC-MDPs with Communication (DEC-MDP-Com) [6] make a clear distinction between a domain level action and an explicit communication action. In this paper, we prove that these two classes of models are equivalent. While making a clear distinction between explicit communication and domain level actions allows us to study the effects of communication more easily, this distinction does not increase the expressiveness of the model. Becker et al [2] made an early attempt to identify and exploit structure in the interactions between agents to reduce the complexity. This work explains the complexity results they found.

This paper is organized as follows. Section 2 presents two formal models that we use throughout the paper. Section 3 discusses different types of interactions between agents and presents the condition that determines the complexity of a distributed MDP. Two examples of NP-complete interaction protocols are found in Section 4, and Section 5 contains two examples of how harder problems can be approximated. Section 6 discusses future work and conclusions.

2. FORMAL MODELS

There have been a number of distributed decision-theoretic models for multi-agent systems published in the last few years. Many of these models describe the same class of problems. We will discuss two models, the DEC-MDP-Com and the DEC-MDP. While these two models are equivalent, they do represent interactions between the agents differently and it is therefore useful to examine both.

The models are based on the single agent Partially Observable Markov Decision Process (POMDP) extended to apply to systems with multiple agents. In this paper, we will define them for two agents i and j, but the definitions easily extend to any number of agents. The basic idea is that the state, now known as the global state, encompasses all of the necessary information about the state of each agent (local state) and the environment. Each agent has its own set of actions it takes and observations it makes. The transition function is from world states to world states given the actions taken by each agent, the joint action. The complexity of these models is NEXP-complete [4]. The reason that it is harder than the POMDP is that the agents are all receiving different observations, and only together could they identify the current global state. To predict what action agent i will take requires agent j to maintain not only a belief about the global state (similar to a POMDP) but also a belief about the belief agent i has. Interactions between the agents are built into the transition and observation functions. The likelihood that agent i will see a particular observation depends directly on the actions of both agents as well as the new global state, which in turn depends on the actions of both agents. The DEC-MDP stops there in its definition (see [4] for a formal definition), but the DEC-MDP-Com explicitly represents another form of interaction, communication.

DEFINITION 1 (ADAPTED FROM [7]). *A DEC-MDP-Com is a tuple: $\langle S, A, \Sigma, C_\Sigma, P, R, \Omega, O \rangle$, where:*

- *S is a finite set of global states, with a distinguished initial state s_0.*
- *$A = A_i \times A_j$ is a finite set of joint actions. A_i are the local actions for agent i.*
- *$\Sigma = \Sigma_i \times \Sigma_j$ is the alphabet of messages. $\sigma_i \in \Sigma_i$ denotes an atomic message sent by agent i. $\overline{\sigma}_i$ is a sequence of atomic messages sent by agent i. $\epsilon_\sigma \in \Sigma_i$ denotes not sending a message to the other agent.*
- *$C_\Sigma \rightarrow \Re$ is the cost of transmitting an atomic message.*
- *$P : S \times A \times S \rightarrow \Re$ is the transition function. $P(s'|s, a_i, a_j)$ is the probability of the outcome state s' when the joint action (a_i, a_j) is taken in state s.*
- *$R : S \times A \times S \rightarrow \Re$ is the reward function. $R(s, a_i, a_j, s')$ is the reward obtained from taking joint action (a_i, a_j) in state s and transitioning to state s'.*
- *$\Omega = \Omega_i \times \Omega_j$ is a finite set of joint observations. Ω_i is the set of observations for agent i.*
- *$O : S \times A \times S \times \Omega \rightarrow \Re$ is the observation function. $O(s, a_i, a_j, s', o_i, o_j)$ is the probability of agents i and j seeing observations o_i and o_j after the sequence s, (a_i, a_j), s' occurs.*
- *Joint full observability: the pair of observations made by the agents together fully determine the current state. If $O(s, a_i, a_j, s', o_i, o_j) > 0$ then $P(s'|o_i, o_j) = 1$.*

The policy for an agent is a mapping from all of the available information to a domain action and a communication action. Even though the global state is Markov, the agents base their decisions on the history of observations just like in a POMDP. However, if the agents ever communicate such that they both know the global state, i.e., they exchange their most recent observation, then they do not need to remember the prior history due to the Markov property. In

effect, this synchronization resets the problem but with a new starting state.

DEFINITION 2. *The local policy for agent i, π_i, is a mapping from the history of observations $\overline{\Omega}_i$, the history of messages sent $\overline{\Sigma}_i$, and the history of messages received $\overline{\Sigma}_j$ since the last synchronized world state S to a domain action and a communication action.*

$$\pi_i : S \times \overline{\Omega}_i \times \overline{\Sigma}_i \times \overline{\Sigma}_j \rightarrow A_i \times \Sigma_i.$$

The goal for a DEC-MDP-Com is to find a joint policy $\pi = \langle \pi_i, \pi_j \rangle$ that maximizes the expected value.

An important question is whether or not the communication explicitly represented in the DEC-MDP-Com increases the expressiveness of the model. It turns out that it does not – the communication actions are just special types of domain actions and the message received are just special types of observations.

THEOREM 1. *DEC-MDP-Com is equivalent to DEC-MDP.*

PROOF.

• **DEC-MDP $\leq_p$ DEC-MDP-Com.**

This reduction is trivial. To any DEC-MDP, we add $\Sigma = C_\Sigma = \emptyset$ and we get an equivalent DEC-MDP-Com.

• **DEC-MDP-Com $\leq_p$ DEC-MDP.**

We reduce a DEC-MDP-Com $\langle S, A, \Sigma, C_\Sigma, P, R, \Omega, O \rangle$ to an equivalent DEC-MDP $\langle \hat{S}, \hat{A}, \hat{P}, \hat{R}, \hat{\Omega}, \hat{O} \rangle$.

The basic idea is to introduce the two step process of the DEC-MDP-Com into the DEC-MDP by doubling the state space: $\hat{S} = S \times \{0, 1\}$. The states $\hat{s} = [s, 0]$ are for taking domain actions A_i and receiving observations Ω_i. The states $\hat{s} = [s, 1]$ are for taking communication actions Σ_i and receiving communications Σ_j. The total space of actions is therefore $\hat{A}_i = A_i \cup \Sigma_i$. The observations that agent i receives include both the messages sent by agent i and the messages received from agent j, i.e., $\hat{\Omega}_i = \Omega_i \times \Sigma_i \times \Sigma_j$. When taking domain actions nothing changes in the functions $\hat{P}$, $\hat{R}$ and $\hat{O}$:

$$\hat{P}([s, 0], a_1, a_2, [s', 1]) = P(s, a_1, a_2, s').$$
$$\hat{R}([s, 0], a_1, a_2, [s', 1]) = R(s, a_1, a_2, s').$$
$$\hat{O}([s, 0], a_1, a_2, [s', 1], o_1, o_2) = O(s, a_1, a_2, s', o_1, o_2).$$

When taking the communication actions, they do change:

$$\hat{P}([s, 1], \sigma_1, \sigma_2, [s, 0]) = 1.$$
$$\hat{R}([s, 1], \sigma_1, \sigma_2, [s, 0]) = C_\Sigma(\sigma_1) + C_\Sigma(\sigma_2).$$
$$\hat{O}([s, 1], \sigma_1, \sigma_2, [s, 0], \sigma_1\sigma_2, \sigma_2\sigma_1) = 1.$$

Therefore, the DEC-MDP-Com is equivalent to the DEC-MDP. □

Theorem 1 states that DEC-MDP-Com and DEC-MDP have the same expressiveness. However, the distinction between the communication actions and the domain actions can be very useful, as we will show in the next section.

3. POLYNOMIALLY ENCODABLE INTERACTIONS

The DEC-MDP-Com and related models allow for a very general form of interaction between the agents. The complexity for those problems has been proven to be NEXP-complete [4, 7]. At the other end of the spectrum would be to disallow all interactions between the agents. In effect, each agent would be independently solving a local MDP that represents its part of the system. MDPs are P-complete, so there is something about the interactions between the agents which is the cause of the complexity. As additional evidence, Becker et al [3] defined a class of multi-agent problems in which the agents were almost completely independent. Each agent had a local MDP that described its part of the system. The agents could not communicate in any way nor could they take an action that would influence another agent. However, the system received a reward that depended on the local states and actions of all of the agents, and the goal was to find a joint policy that maximized the sum of the expected local and global rewards. This class of problems in which the agents can only interact through the reward function proved to be NP-complete. Goldman and Zilberstein [7] also showed that by following certain communication protocols the agents from the previous example could communicate with each other and the problem remained at NP-complete.

This section will examine different types of interactions between agents and provide theoretical results to explain how and why the interactions affect the complexity of finding optimal solutions. The next section will elaborate on the two NP-complete examples introduced above and prove that they meet this condition.

We classify the actions agents can take into two groups: non-interacting (or independent) actions and interacting (or dependent) actions. Independent actions are those that do not affect the other agent and neither agent receives any information about the other. Dependent actions are those that affect the other agent in some way. For example, robot i could pick up robot j and move it, which would affect the local state of robot j. Communication is another example: agent i could send a message to agent j, which would change the knowledge agent j has. We can further subdivide dependent actions into explicit and implicit communication. Normally when one thinks about communication, i.e., sending a message, one is talking about explicit communication. This is the communication part of the DEC-MDP-Com. Implicit communication is the information an agent receives by a domain action, like the example of a robot picking up and moving another robot. The robot being picked up gains information about the local state and belief of the other robot, namely the location of the other robot and the fact that the other robot felt this was a useful action to take.

We will illustrate these interactions with a token collecting example. There is a $n \times n$ grid world, which is populated by two agents and a number of tokens. The agents can observe their own locations and the locations of the tokens. When an agent picks up a token, the system gets a positive reward. The goal of the system is to maximize the total reward within time T. This problem can be modeled in a DEC-MDP. The world state includes the locations of both agents, the locations of the tokens and the time left. The agents' observations at each time step include the agents' own location, the location of the tokens, and the time left. At every time step, each agent can either move to an adjacent square or pick up a token at its current location. If an agent moves, its action does not affect the other agent's observations, and therefore the movement actions are independent actions. However, if agent i picks up a token, agent j can observe the fact that one of the tokens just dis-

appeared. By comparing its current observation and the observation at the last time step, agent j can infer the exact location of agent i and therefore has the complete knowledge of the current world state. As a result, the token collecting action is a dependent action even though there is no explicit communication in the system.

We defined a dependent action as an action that affects the observations of the other agent and therefore changes its belief about the global state. If a dependent action is explicitly modeled in a DEC-MDP-Com, its effect is recorded by the communication action σ_i itself. On the other hand, if it is not explicitly modeled in a DEC-MDP-Com, its effect is recorded by the observations of the agents. The observation history $\overline{\Omega}_i$ records the interaction history of agent i in the DEC-MDP. Consequently, in a DEC-MDP-Com where there are communication actions explicitly modeled, the interaction history of agent i is $\overline{\Omega}_i \times \overline{\Sigma}_i \times \overline{\Sigma}_j$.

DEFINITION 3. *We call E_i an* **encoding** *of the interaction history of agent i, if a joint policy $\tilde{\pi} = \langle \tilde{\pi}_1, \tilde{\pi}_2 \rangle$ is sufficient to maximize the global value, where $\tilde{\pi}_i$ is of the form $S \times E_i \rightarrow A_i \times \Sigma_i$ for a DEC-MDP-Com, or of the form $E_i \rightarrow A_i$ for a DEC-MDP.*

The encoding represents removing all elements from the interaction history that are unnecessary to generate an optimal policy. The important characteristic is the size of the smallest encoding. The interaction history is normally considered to be exponential in $|S|$ because the length of the observation sequence is $O(|S|)$. In some problems, however, the smallest encoding is only polynomial in $|S|$.

DEFINITION 4. *The interaction history of a DEC-MDP/DEC-MDP-Com is* **polynomially encodable** *if there exists an encoding E_i for each interaction history $\overline{\Omega}_i$ and a constant c_i, such that $|E_i| = O(|S|^{c_i})$.*

The criteria that determines the complexity of a multi-agent system is whether the interaction history can be polynomially encoded. If it can, then the problem is in NP. If it cannot be polynomially encoded, then it is provably harder than NP. The following two theorems prove this relationship between the encoding and the complexity.

THEOREM 2. *Deciding a polynomially encodable DEC-MDP/DEC-MDP-Com is NP-complete.*

PROOF. Here we prove the DEC-MDP case, the DEC-MDP-Com is essentially the same.

To prove NP-completeness, we (1) provide a polynomial time verifier and (2) show a reduction from an NP-Complete problem to this one.

(1) A joint policy in a DEC-MDP can be evaluated by representing it as a belief state graph. Each node in the graph is composed of the state, the sequence of observations for agent i and for agent j. Each node has a single joint action, which is defined by the joint policy. The transitions between the nodes depends only on the transition and observation functions, and each transition has a reward defined by the reward function. The belief state graph can be evaluated using the standard MDP recursive value function and policy evaluation, which runs in time polynomial in the size of the graph. For a DEC-MDP, this size is $|S| \times |\Omega_i|^T \times |\Omega_j|^T$, where $T = O(|S|)$. However, since there exists a polynomial encoding E_i for each observation sequence $\overline{\Omega}_i$, the size of the graph is only $|S| \times |E_i| \times |E_j|$ and the policy evaluation takes $O((|S|^{c_i+c_j+1})^c)$, which is polynomial in the size of the state space for constants c_i, c_j and c.

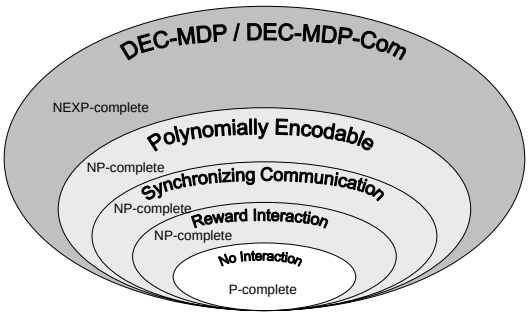


Figure 1: The relationships and complexity between various distributed MDP models.

(2) To prove the lower bound we will reduce the NP-complete problem DTEAM [9, 10] to this problem. DTEAM is a single-step discrete team decision problem with two agents. Agent i observes a random integer k_i, $1 \leq k_i \leq N$, and takes an action $\gamma_i(k_i) \in \{1, \ldots, M\}$. Their actions incur cost $c(k_i, k_j, \gamma_i(k_i), \gamma_j(k_j))$. The problem is to find policies γ_i and γ_j that minimize the expected cost:

$$\sum_{k_i=1}^{N} \sum_{k_j=1}^{N} c(k_i, k_j, \gamma_i(k_i), \gamma_j(k_j)).$$

The reduction is quite straightforward. In the initial state, the agents take a null action and transition to one of N^2 intermediate states that correspond to the random integers k_i and k_j. Agent i observes k_i and takes its action to reach the final state. The reward is negative the cost, $R(\cdot) = -c(\cdot)$. The size of the observation sequence $|\overline{\Omega}_i| = N = O(|S|)$ is polynomial in the size of the state space.

Given the polynomial time verifier and the reduction, a DEC-MDP with a polynomially encodable interaction history is NP-complete. □

THEOREM 3. *Deciding a non-polynomially encodable DEC-MDP/DEC-MDP-Com is harder than NP.*

PROOF. We prove this by contradiction. Assume that an arbitrary DEC-MDP without a polynomial encoding is in NP. This means that there exists a polynomial time verifier for any policy in the DEC-MDP. If a policy can be verified in polynomial time then it must have a polynomial sized representation. Since a policy is a mapping from $E_i \rightarrow A_i$, this polynomial sized representation is the encoding of the interaction history. Contradiction. Therefore, the DEC-MDP without a polynomial encoding is not in NP.

The proof for DEC-MDP-Com is similar. □

Figure 1 illustrates the relationship and complexity between the models discussed in this paper. Polynomially Encodable interaction histories are an NP-complete subset of DEC-MDPs. Other models, like the synchronizing communication are NP-complete subsets of polynomially encodable problems. No interaction between the agents is a P-complete class of problems.

4. EXAMPLES OF PROTOCOLS

In this section, we present two interaction protocols known to be NP-complete, and demonstrate how to prove the existence of a polynomial encodings.

4.1 Reward Dependence

The first example is a DEC-MDP in which the agents are mostly independent of each other. All of their actions are independent actions. The dependence between the agents comes from the reward function which depends on the world state and joint action. Becker et al. [3] formally defined this class of problems as a Transition Independent DEC-MDP (TI-DEC-MDP). It is a DEC-MDP with a factored state space, which means that there is a local state for each agent and the global state is the product of all of the local states, $S = S_i \times S_j$. The transition from one local state to the next depends only on the actions of that agent. Similarly, the observations of agent i depends only on i's local states and actions. We call these properties transition and observation independent.

DEFINITION 5 ([3]). *A factored DEC-MDP is said to be* **transition independent** *if the new local state of each agent depends only on its previous local state and the action taken by that agent:*

$$P(s_i'|(s_i, s_j), (a_i, a_j), s_j') = P_i(s_i'|s_i, a_i).$$

DEFINITION 6 ([3]). *A factored DEC-MDP is said to be* **observation independent** *if the observation an agent sees depends only on that agent's current and next local state and current action:* $\forall o_i \in \Omega_i$

$$P(o_i|(s_i, s_j), (a_i, a_j), (s_i', s_j'), o_j) = P(o_i|s_i, a_i, s_i').$$

At each step each agent fully observes its local state, and observes no information about the local state of the other agent.

DEFINITION 7 ([3]). *A factored DEC-MDP is said to be* **locally fully observable** *if each agent fully observes its own local state at each step., i.e.,* $\forall o_i \ \exists s_i \ : P(s_i|o_i) = 1$.

The interaction history of the DEC-MDP is the sequence of local observations, which in this problem translates into the sequence of local states. However, due to the limited interaction, the most recent local state and the time step is sufficient to maximize the global value. Proving that this encoding is sufficient is the primary component in proving that this class of problems is polynomially encodable.

THEOREM 4. *The interaction history of a DEC-MDP with independent transitions and independent observations is polynomially encodable.*

PROOF. The interaction history of a transition independent and observation independent DEC-MDP is a sequence of local states $\overline{s}_i$ with an upper bound on the length being $T = O(|S|)$. We will prove that there exists an encoding of the interaction history composed of the last state in the sequence and the length of the sequence. This encoding $E_i = S_i \times T$ is polynomial: $|S_i \times T| = O(|S|^2)$.

The Q value of the belief state graph (see Theorem 2) is the Q value of taking the given action in the current state sequence and then following the given optimal policy $\overline{\pi}^* = \langle \overline{\pi}_1^*, \overline{\pi}_2^* \rangle$, where $\overline{\pi}_i^* : \overline{S}_i \rightarrow A_i$.

$$Q_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2) = \sum_{s_1' s_2'} P(s_1' s_2' | \overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2) \times$$
$$[R(\overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2, s_1' s_2') + \gamma V_{\overline{\pi}^*}(\overline{s}_1 s_1 s_1', \overline{s}_2 s_2 s_2')].$$

If for all possible histories $\overline{s}_1', \overline{s}_2'$ of length T that led to the current state, the Q value is the same, then the history of states is irrelevant and can be replaced in the policy by just the length of the sequence T.

To prove $Q_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2) = Q_{\overline{\pi}^*}(\overline{s}_1' s_1, \overline{s}_2' s_2, a_1 a_2)$, we have to show three things:

1. $P(s_1' s_2' | \overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2) = P(s_1' s_2' | \overline{s}_1' s_1, \overline{s}_2' s_2, a_1 a_2)$.

From the definition of Markov,

$$P(s_1' s_2' | \overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2) = P(s_1' s_2' | s_1 s_2, a_1 a_2). \quad (1)$$

2. $R(\overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2, s_1' s_2') = R(\overline{s}_1' s_1, \overline{s}_2' s_2, a_1 a_2, s_1' s_2')$.

From the definition of reward,

$$R(\overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2, s_1' s_2') = R(s_1 s_2, a_1 a_2, s_1' s_2'). \quad (2)$$

3. $V_{\overline{\pi}^*}(\overline{s}_1 s_1 s_1', \overline{s}_2 s_2 s_2') = V_{\overline{\pi}^*}(\overline{s}_1' s_1 s_1', \overline{s}_2' s_2 s_2')$

We show $V_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2) = V_{\overline{\pi}^*}(\overline{s}_1' s_1, \overline{s}_2' s_2)$ by induction.

Base case: s_1 and s_2 are final states and their values are always zero. $V_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2) = 0$, $V_{\overline{\pi}^*}(\overline{s}_1' s_1, \overline{s}_2' s_2) = 0$.

Inductive case: We assume it is true for

$$V_{\overline{\pi}^*}(\overline{s}_1 s_1 s_1', \overline{s}_2 s_2 s_2') = V_{\overline{\pi}^*}(\overline{s}_1' s_1 s_1', \overline{s}_2' s_2 s_2'), \quad (3)$$

we need to show that it is true for

$$V_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2) = V_{\overline{\pi}^*}(\overline{s}_1' s_1, \overline{s}_2' s_2).$$

The value function is very similar to the Q function, except the current action is chosen from the policy.

$$\begin{aligned}
& V_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2) \\
= & \sum_{s_1' s_2'} P(s_1' s_2' | \overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2) \times \\
& [R(\overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2, s_1' s_2') + \gamma V_{\overline{\pi}^*}(\overline{s}_1 s_1 s_1', \overline{s}_2 s_2 s_2')] \\
\overset{(1),(2)}{=} & \sum_{s_1' s_2'} P(s_1' s_2' | s_1 s_2, a_1 a_2) \times \\
& [R(s_1 s_2, a_1 a_2, s_1' s_2') + \gamma V_{\overline{\pi}^*}(\overline{s}_1 s_1 s_1', \overline{s}_2 s_2 s_2')] \\
\overset{(1),(2),(3)}{=} & \sum_{s_1' s_2'} P(s_1' s_2' | \overline{s}_1' s_1, \overline{s}_2' s_2, a_1 a_2) \times \\
& [R(\overline{s}_1' s_1, \overline{s}_2' s_2, a_1 a_2, s_1' s_2') + \gamma V_{\overline{\pi}^*}(\overline{s}_1' s_1 s_1', \overline{s}_2' s_2 s_2')]
\end{aligned}$$

Since a_1' and a_2' are optimal actions, we have:

$$\begin{aligned}
& \sum_{s_1' s_2'} P(s_1' s_2' | \overline{s}_1' s_1, \overline{s}_2' s_2, a_1 a_2) [R(\overline{s}_1' s_1, \overline{s}_2' s_2, a_1 a_2, s_1' s_2') \\
& + \gamma V_{\overline{\pi}^*}(\overline{s}_1' s_1 s_1', \overline{s}_2' s_2 s_2')] \\
\leq & \sum_{s_1' s_2'} P(s_1' s_2' | \overline{s}_1' s_1, \overline{s}_2' s_2, a_1' a_2') [R(\overline{s}_1' s_1, \overline{s}_2' s_2, a_1' a_2', s_1' s_2') \\
& + \gamma V_{\overline{\pi}^*}(\overline{s}_1' s_1 s_1', \overline{s}_2' s_2 s_2')] \\
= & V_{\overline{\pi}^*}(\overline{s}_1' s_1, \overline{s}_2' s_2)
\end{aligned}$$

As a result, we have $V_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2) \leq V_{\overline{\pi}^*}(\overline{s}_1' s_1, \overline{s}_2' s_2)$. Due to symmetry, we can also show that $V_{\overline{\pi}^*}(\overline{s}_1' s_1, \overline{s}_2' s_2) \leq V_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2)$. Therefore $V_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2) = V_{\overline{\pi}^*}(\overline{s}_1' s_1, \overline{s}_2' s_2)$.

Since the value of taking a joint action while following the optimal policy does not depend on the history, the same joint action is optimal for all histories and the policy need not include it. The interaction history can be summarized by the current state and time. □

Theorem 4 implies that a DEC-MDP with independent transitions and observations can be decomposed into two independent MDPs, with local states only affected by the local actions. The policies are standard policies for MDPs with the addition of time, and the goal is to maximize the expected reward received from a global value function.

4.2 Synchronizing Communication

In the DEC-MDP with independent transitions and observations presented above, the agents interact only through reward function. This highly restricted form of interaction does not reveal any information about other agent's local state or observations. In this section, we look at a less restricted form of interaction, which we call **synchronizing communication**. We define it in the context of DEC-MDP-Com, since the explicit modeling of communication allows it to remain distinct from the domain actions.

Definition 8. *A communication protocol is said to be a* **synchronizing communication protocol** *if whenever any agent communicates, all agents send sufficient information to all other agents to unify their world views. Such an exchange is viewed as a single communication action with a single cost, even though there are potentially many messages sent.*

In a DEC-MDP-Com with a synchronizing communication protocol, whenever there is communication, each agent has the same view of the world. Since the world state is jointly fully observable, each agent has a complete view of the world state, and knows that the other agent has a complete view as well. The DEC-MDP-Com is essentially reset to an identical problem with a different start state, and the agents are safe to forget their past observation histories and communication histories. The communication actions essentially divide the DEC-MDP-Com into individual episodes, each of which is a DEC-MDP with no communication actions. The length of each episode varies depending on when it is optimal to communicate.

There are many applications in which synchronizing communication is an appropriate protocol. In certain problems, the communication setup cost is so high that it does not matter how much actual information is transferred. In other systems, the minimum packet size sent over the network may be larger than the messages the agents send, giving them a constant cost per message. For applications such as these, the amount of information contained in each message does not change its cost. A communication protocol is said to have **constant cost** if all the communication actions have the same cost. Specifically, a synchronizing communication action has the same cost as any other communication actions, no matter how many messages are actually exchanged to synchronize their partial views of the world state. Goldman and Zilberstein [7] proved that given a DEC-MDP-Com with constant communication cost, there is an optimal communication policy such that whenever there is communication, the agents exchange their last observations. Since a DEC-MDP-Com is jointly fully observable, when the agents exchange their last observations, they synchronize their views of the global state. As a result, if a DEC-MDP-Com has constant communication cost, there is an optimal communication policy such that whenever there is communication between the agents, it is synchronizing communication.

The second example of a polynomially encodable interaction protocol is the synchronizing communication in a DEC-MDP-Com with independent transitions and observations and a constant communication cost. This protocol has been studied in other work [7, 1].

Theorem 5. *The interaction history of a DEC-MDP-Com with independent transitions and observations and constant communication cost is polynomially encodable.*

Proof. From Definition 2, a local policy for a DEC-MDP-Com is of the form $\pi_i : S \times \overline{\Omega}_i \times \overline{\Sigma}_i \times \overline{\Sigma}_j \rightarrow A_i \times \Sigma_i$, where S is the last synchronized state, and $\overline{\Omega}_i$, $\overline{\Sigma}_i$ and $\overline{\Sigma}_j$ are the observation history and communication history since S. Since the DEC-MDP-Com has a constant communication cost, a synchronizing communication protocol is optimal. As a result, whenever there is communication the last synchronized global state S is updated to the newly synchronized state, and $\overline{\Omega}_i$ is set to $\emptyset$. When there is no communication, the observation is appended to $\overline{\Omega}_i$. Neither $\overline{\Sigma}_i$ nor $\overline{\Sigma}_j$ are used. Therefore, between communications the DEC-MDP-Com is equivalent to a DEC-MDP with independent transitions and observations, which is polynomially encodable (from Theorem 4). Since the interaction history between communications is polynomially encodable and communication resets the interaction history to $\emptyset$, the problem is polynomially encodable. □

Even though a DEC-MDP-Com with independent transitions and observations and constant communication cost has considerably more communication than a DEC-MDP with only reward dependence, Theorem 5 shows that its interaction protocol is still polynomially encodable, and therefore it remains in NP.

5. EXAMPLES OF APPROXIMATIONS

For applications where communication cost is constant, one can find an optimal policy that periodically synchronizes the world views of the agents. However, there are many other applications where the cost of message depends on its contents, and it may be beneficial to send less information than would synchronize the agents' world views. Unfortunately, most such interactions do not seem to be polynomially encodable because each piece of information changes an agent's belief about the local state of the other agent. A good guideline is that an agent needs to keep track of both the other agent's local state as well as the other agent's knowledge. However, we may still be able to design approximate encodings that are polynomial for such problems. The purpose of these encodings is to put extra restrictions on the interactions so that the complexity of approximating the optimal policy is reduced to at most NP. In this section, we show two examples of such approximations.

5.1 Constant Horizon

Consider the token collecting example. While there is no explicit communication between the agents, both of the agents observe the location of all available tokens. When agent i picks up a token at time t, agent j observes the fact that a token disappeared and can infer the location of agent

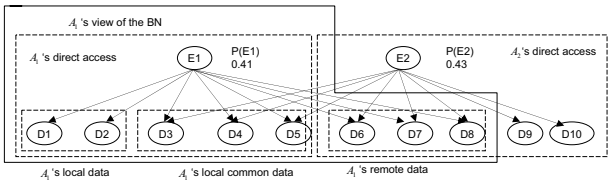


Figure 2: There are two events E_1 and E_2. Data $D_1, D_2, ...D_{10}$ are distributed between two agents. A_1 has access to $D_1, ...D_5$ and is responsible for solving E_1, while A_2 can see only $D_6, ...D_{10}$ and is responsible for solving E_2. The objective is for A_1 and A_2 to figure out what E_1 and E_2 are with required confidence and with minimum expected communication cost

i. With this observation, agent j knows the global state but agent i does not, so the interaction is not synchronizing but what we call asymmetric synchronizing. Asymmetric synchronization is where an agent either gains no information about the other agent's local state or complete information, giving it a belief of 1.0 about the current global state. The difference between synchronization and asymmetric synchronization is in the asymmetric case the agents do not necessarily both synchronize at the same time.

This difference precludes a polynomial encoding. In the token collecting example when agent i picks up a token both agents i and j know that i picked it up as well as both know the other knows, and so on. Agent i, then, must keep track of the information j has collected about i to best predict what j will do at least until j picks up a token itself and i learns where j is located. Remembering this sequence of information is exponential in the size of the state space in the worst case.

While the problem itself is harder than NP, it can be approximated by making assumptions that allow the interaction history to be polynomially encoded. For example, instead of keeping track of the entire history of interaction, one could assume that the last c interactions, for some constant c, was sufficient. The interaction history is now of size $|S|^c$, which is polynomial in the size of the state space. In the token collecting example this could correspond to agent i remembering the last 5 tokens it collected.

5.2 Flat Representation

Now let us look at a communication optimization problem in a Distributed Sensor Interpretation (DSI) system [13]. There are two agents in the system. Each agent directly observes data from a group of sensors and is responsible for solving a subset of the interpretation problem to best explain the sensor data collectively observed by the agents. Since each subproblem may relate to not only the sensor data collected by the agent itself, but also those collected by the other agent, the agents may need to communicate with each other in order to solve its local problem with required confidence. Figure 2 illustrates such a DSI system that is represented in a Bayesian Network. The top level nodes are the events that are the possible causes of the observed data, while the leaves are the raw data gathered by various agents. There are two classes of actions for each agent: SEND and REQUEST. An agent can only send one piece of data that the other agent has no knowledge of, and can only request one piece of the remote data that it has

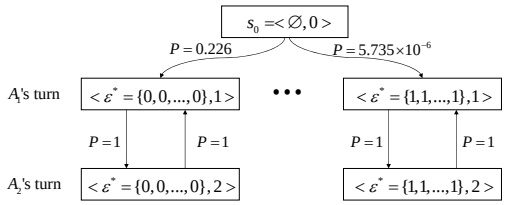


Figure 3: The decentralized MDP generated for the problem in Figure 2.

not yet acquired. The two agents alternate actions until the confidence threshold is reached by both. The communication optimization problem is, for the specified BN structure, to find the communication strategy whose expected cost is minimized in order to reach the required confidence for both agents.

Shen et al. [13] models this problem with a DEC-MDP as shown in Figure 3. Every global state is in the form of $s = \langle e^*, i \rangle$, where e^* is all of the sensor data values observed by the system, and i is an external feature that indicates it is agent i's turn to communicate. s_0 is a dummy start state $\langle \emptyset, 0 \rangle$. When all the sensors collect their data, the start state transitions to one of the possible real global states $\langle e^*, 1 \rangle$ before any communication takes place. A_1 and A_2 are action sets of the two agents, $A_i = \{$SEND x, REQUEST $y\}$ for local data x and remote data y. An observation o_i is the data value just sent or received by agent i. As a special case, after s_0 each agent observes the values of its local data.

In this DEC-MDP every action is explicit communication. Instead of explicitly modeling it as Σ in a DEC-MDP-Com, these actions are implicitly modeled in the DEC-MDP framework itself. The observation history $\overline{\Omega}$ records the interaction history of the system. Since every observation at every time step is needed to calculate the confidence level achieved, it is necessary for the agents to remember all the data values transferred between the agents in the past. Furthermore, remembering the order in which the data are exchanged is valuable because it carries useful information. An agent can infer why the other agent chose to transfer this piece of data before the other piece. Therefore, remembering the entire $\overline{\Omega}$ is essential to generating the optimal policy. We write the number of local sensor data as n, and assume that each sensor data has at most m possible values. As a result, $|S| = m^n$, and $|\overline{\Omega}| = O(n! \cdot m^n) = O(n^n) = O(|S|^{\log_2 n})$. Since n is not independent of $|S|$, $\overline{\Omega}$ is not polynomially encodable and therefore the DEC-MDP is harder than NP.

One way to approximate the optimal solution to this problem is that, instead of remembering the entire $\overline{\Omega}$, each agent only remembers the values of the data exchanged so far without remembering the order in which they were transferred. This is a reasonable approximation since to calculate the confidence level of the local interpretations, only the sensor data values are needed. In this approximation, the approximate encoding E_i of $\overline{\Omega}$ is of the size $O((m+1)^n) = O(|S|^{\log_m (m+1)})$. Since m is independent of $|S|$, E_i is a polynomial encoding, and therefore this approximation is no harder than NP.

6. CONCLUSIONS

Distributed POMDPs have been developed and employed to model various multi-agent coordination problems. Un-

derstanding the source of their high complexity is crucial to identifying new and more tractable models as well as developing appropriate approximations to otherwise intractable problems. This paper establishes that the interactions present among the agents is the cause of the high complexity of distributed POMDPs. We proved that deciding a distributed POMDP whose interaction history contains information of a size polynomial in the number of states is NP-complete, and that deciding a non-polynomially encodable distributed POMDP is harder than NP. We demonstrated how two subclasses of distributed POMDPs known to be NP-complete can be polynomially encoded. This is the first time that a well-defined condition has been identified that can distinguish between multi-agent problems in NP and those that are strictly harder than NP. It is an important step in mapping out the complexity hierarchy of multi-agent systems.

Our goal in this paper was not to introduce new models or algorithms, but to change the way people view interactions between agents in the context of distributed POMDPs. Multi-agent researchers have long intuitively understood that the interaction between the agents is the cause of their high complexity. The theoretical results of this paper are significant in that they both formally justify this intuition as well as explain how the interaction affects the complexity. This new understanding of interaction and its relationship to complexity will help us to identify new classes of multi-agent systems with a lower complexity.

The significance of this theoretical result also has a more practical side. Most multi-agent systems are provably harder than NP and solving them optimally is not possible. Much work has been put into developing good algorithms for approximating these problems. This work provides theoretical guidance in understanding how the approximations in a model limit the search space and reduce the complexity. We demonstrate this on two non-polynomially encodable problems by providing two assumptions that reduce the complexity to no harder than NP.

We are currently pushing this work in two directions. First, we are searching for new examples of polynomially encodable interaction protocols, as well as common protocols that we can prove are not polynomially encodable. Second, we are evaluating the performance of the approximations we presented here in addition to finding new approximations appropriate to the protocols that we can prove to be harder than NP.

7. ACKNOWLEDGMENTS

This work was supported in part by NASA under grant NCC 2-1311, by NSF under grants IIS-0219606 and IIS-0414711, and by AFOSR under grant F49620-03-1-0090. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of NASA, NSF or AFOSR.

8. REFERENCES

[1] R. Becker, V. Lesser, and S. Zilberstein. Analyzing myopic approaches for multi-agent communication. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 550–557, Compiegne, France, September 2005. IEEE Computer Society.

[2] R. Becker, S. Zilberstein, and V. Lesser. Decentralized Markov decision processes with structured transitions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, volume 1, pages 302–309, New York City, New York, July 2004. ACM Press.

[3] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Solving transition independent decentralized MDPs. *Journal of Artificial Intelligence Research*, 22:423–455, 2004.

[4] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, November 2002.

[5] M. Ghavamzadeh and S. Mahadevan. Learning to communicate and act in cooperative multiagent systems using hierarchical reinforcement learning. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 1114–1121, New York, July 2004. ACM Press.

[6] C. V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 137–144, Melbourne, Australia, July 2003. ACM Press.

[7] C. V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, 22:143–174, 2004.

[8] H. Li, E. H. Durfee, and K. G. Shin. Multiagent planning for agents with internal execution resource constraints. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 560–567, New York, NY, USA, 2003. ACM Press.

[9] C. H. Papadimitriou and J. Tsitsiklis. On the complexity of designing distributed protocols. *Information and Control*, 53:211–218, 1982.

[10] C. H. Papadimitriou and J. Tsitsiklis. Intractable problems in control theory. *SIAM Journal on Control and Optimization*, 24(4):639–654, 1986.

[11] L. Peshkin, K.-E. Kim, N. Meuleau, and L. P. Kaelbling. Learning to cooperate via policy search. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 489–496, San Francisco, CA, 2000. Morgan Kaufmann.

[12] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[13] J. Shen, V. Lesser, and N. Carver. Minimizing communication cost in a distributed Bayesian network using a decentralized MDP. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 678–685, Melbourne, Australia, July 2003. ACM Press.

[14] P. Xuan and V. Lesser. Multi-agent policies: From centralized ones to decentralized ones. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 1098–1105. ACM Press, 2002.

A Technique for Reducing Normal-Form Games to Compute a Nash Equilibrium*

Vincent Conitzer
Carnegie Mellon University
Computer Science Department
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
conitzer@cs.cmu.edu

Tuomas Sandholm
Carnegie Mellon University
Computer Science Department
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
sandholm@cs.cmu.edu

ABSTRACT

We present a technique for reducing a normal-form (aka. (bi)matrix) game, O, to a smaller normal-form game, R, for the purpose of computing a Nash equilibrium. This is done by computing a Nash equilibrium for a subcomponent, G, of O for which a certain condition holds. We also show that such a subcomponent G on which to apply the technique can be found in polynomial time (if it exists), by showing that the condition that G needs to satisfy can be modeled as a Horn satisfiability problem. We show that the technique does not extend to computing Pareto-optimal or welfare-maximizing equilibria. We present a class of games, which we call ALAGIU (Any Lower Action Gives Identical Utility) games, for which recursive application of (special cases of) the technique is sufficient for finding a Nash equilibrium in linear time. Finally, we discuss using the technique to compute *approximate* Nash equilibria.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*; F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Algorithms, Economics, Theory

Keywords

Game Theory, Computing Nash Equilibria, Preprocessing

*This material is based upon work supported by the National Science Foundation under ITR grants IIS-0121678 and IIS-0427858, a Sloan Fellowship, and an IBM Ph.D. Fellowship.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

Nash equilibrium is the most important solution concept for games. A vector of mixed strategies for the players (that is, for each player, a probability distribution over that player's actions in the game) is a Nash equilibrium if no individual player can benefit from changing her strategy. Every finite game has at least one Nash equilibrium [20]. However, for the concept to be operational, the mere existence result is not sufficient: it needs to be accompanied by an *algorithm* for finding an equilibrium. Unfortunately, even in 2-player normal-form games, it is still unknown whether a polynomial-time algorithm exists for computing a Nash equilibrium. The *Lemke-Howson* algorithm [15] is the best-known algorithm for computing a Nash equilibrium in a 2-player normal-form game, but it can require exponential time [25]. Some special cases can be solved in polynomial time (for example, zero-sum games [18]); other related questions, such as finding a welfare-maximizing equilibrium, are NP-hard [11, 5].

Recently, there has been a renewed surge of interest in the computation of Nash equilibria. Christos Papadmitriou has called the basic problem "a most fundamental computational problem whose complexity is wide open" and "together with factoring, [...] the most important concrete open question on the boundary of P today" [22],[1] and new (exponential-time) algorithms have been suggested that search over the supports of the mixed strategies [23, 24]. Also, there has been growing interest in computing equilibria of games with special structure that allows them to be represented concisely [13, 16, 2, 12, 1].

The idea we pursue in this paper is the following. For the basic problem of computing a Nash equilibrium of a normal-form game, it would be helpful to have a recursive technique that decomposes a Nash equilibrium computation problem into one or more smaller such problems, in such a way that a solution to the original problem can easily be computed from the solutions to the smaller problems. If there were such a technique that could be applied to *any* game, and that de-

[1] A recent sequence of papers [8, 3, 9, 4] shows that the problem of finding a Nash equilibrium (even in the two-player case) is complete for a class of problems called PPAD. This makes it seem less likely that a polynomial-time algorithm for finding a Nash equilibrium exists; on the other hand, unlike (say) NP, PPAD is not known to contain many problems of interest, so that it would be less of a shock if PPAD turned out to be contained in P.

composed it into small enough subproblems, then repeatedly applying this technique would constitute a polynomial-time algorithm for computing a Nash equilibrium. However, even if the technique could not be applied to all games, it would still be of interest. It could be used as a preprocessing step in computing a Nash equilibrium, thereby reducing the load on the algorithm that is eventually called to solve the irreducible subproblems. The technique could also be applied at intermediate stages of any other algorithm that works by reducing the size of the game. Moreover, there may be special subclasses of games such that the technique can be applied on any of those games, as well as on any subproblem resulting from applying the technique on those games. In that case, the technique would constitute a polynomial-time Nash equilibrium finding algorithm for the particular subclass of games.

In this paper, we introduce such a technique for 2-player normal-form games. When possible, this technique finds a subcomponent, G, of the original game, O, for which a certain condition holds. G is then solved recursively. Based on the recursively computed equilibrium of G, the original game O is then reduced to a smaller game, R, which is also solved recursively. From the computed equilibria of G and R, an equilibrium of the original game O can then easily be constructed. To our knowledge, this is the first recursive technique for computing a Nash equilibrium (other than the iterated elimination of dominated strategies[2]).

2. MAIN TECHNIQUE

In this section, we introduce our main technique. We are given a 2-player normal-form game O, in which the row player chooses a pure strategy from Σ_1 and the column player chooses a pure strategy from Σ_2. Suppose that the strategies in Σ_1 can be labeled as u_i, s_i, and those in Σ_2 as v_j, t_j, so that the game can be written as follows:

	v_1	v_2	$\cdots$	v_l	t_1	t_2	$\cdots$	t_n
u_1					c_{11}, b_1	c_{12}, b_1	$\cdots$	c_{1n}, b_1
u_2					c_{21}, b_2	c_{22}, b_2	$\cdots$	c_{2n}, b_2
$\vdots$		H			$\vdots$	$\vdots$		$\vdots$
u_k					c_{k1}, b_k	c_{k2}, b_k	$\cdots$	c_{kn}, b_k
s_1	a_1, d_{11}	a_2, d_{12}	$\cdots$	a_l, d_{1l}				
s_2	a_1, d_{21}	a_2, d_{22}	$\cdots$	a_l, d_{2l}				
$\vdots$	$\vdots$	$\vdots$		$\vdots$		G		
s_m	a_1, d_{m1}	a_2, d_{m2}	$\cdots$	a_l, d_{ml}				

The condition that allows for writing the game like this is the following:

CONDITION 1. *Against any fixed v_j, all the s_i give the row player the same utility (a_j); and against any fixed u_i, all the t_j give the column player the same utility (b_i).*

The intuition for why this condition is useful is the following. Suppose that the row player has decided to play one of the s_i, but has not yet decided which s_i. If the column player plays one of the v_j, then it does not matter which of the s_i the row player plays. Thus, in deciding which s_i to play, the row player only needs to consider the probabilities that the column player places on the t_j. Symmetrically, if the column player has decided to play one of the t_j (but has not yet decided which t_j), then the column player only needs to consider the probabilities that the row player places on the s_i. Hence, intuitively, the parts of the players' mixed strategies that concern the s_i and t_j should themselves be in equilibrium.

So, let us compute a Nash equilibrium of the game G (using any technique):

	t_1	t_2	t_3	$\cdots$	t_n
s_1					
s_2					
s_3		G			
$\vdots$					
s_m					

Let $p^G_{s_i}$ be the row player's probabilities in that Nash equilibrium, and let $p^G_{t_j}$ be the column player's probabilities. Also, let π^G_s be the expected utility for the row player, and let π^G_t be the expected utility for the column player. We can use these to reduce the original game O, as follows. We replace the lower left quadrant of O with a single row. The row player's payoff in the jth entry is a_j (which is the row player's payoff in the jth entry for any of the original rows). The column player's payoff in the jth entry is $\sum_{i=1}^{m} p^G_{s_i} d_{ij}$—that is, the average of the column player's payoffs in the jth entry, taken across the replaced rows, weighted by the probability with which the corresponding s_i is played in the computed equilibrium of G. We replace the upper right quadrant of O with a single column in a similar fashion. Finally, we replace the lower right quadrant (G) with a single entry π^G_s, π^G_t (the expected utilities of the players in the computed equilibrium of G). Thus, the reduced game R is the following:

	v_1	v_2	$\cdots$	v_l	t
u_1					$\sum_{j=1}^{n} p^G_{t_j} c_{1j}, b_1$
u_2					$\sum_{j=1}^{n} p^G_{t_j} c_{2j}, b_2$
$\vdots$		H			$\vdots$
u_k					$\sum_{j=1}^{n} p^G_{t_j} c_{kj}, b_k$
s	$a_1, \sum_{i=1}^{m} p^G_{s_i} d_{i1}$	$a_2, \sum_{i=1}^{m} p^G_{s_i} d_{i2}$	$\cdots$	$a_l, \sum_{i=1}^{m} p^G_{s_i} d_{il}$	π^G_s, π^G_t

We then compute a Nash equilibrium for R (using any technique), obtaining equilibrium probabilities $p^R_{u_i}, p^R_s$ for the row player and equilibrium probabilities $p^R_{v_j}, p^R_t$ for the column player. Then (as we will show later in the section), setting $p^O_{u_i} = p^R_{u_i}, p^O_{s_i} = p^R_s p^G_{s_i}, p^O_{v_j} = p^R_{v_j}$, and $p^O_{t_j} = p^R_t p^G_{t_j}$ constitutes a Nash equilibrium of O.

Before proving formally that the technique is correct, we first illustrate it with a small example. Consider the following game O:

[2]Most variants of iterated dominance can be executed in polynomial time [14, 19, 10, 6]. We have recently proposed a more general strategy eliminability criterion that can help in computing a Nash equilibrium [7], but it is not a recursive technique.

	v_1	t_1	t_2
u_1	$2,2$	$0,3$	$2,3$
s_1	$1,2$	$4,0$	$0,4$
s_2	$1,4$	$0,4$	$4,0$

Condition 1 holds for this game. Thus, we first solve the subcomponent G:

	t_1	t_2
s_1	$4,0$	$0,4$
s_2	$0,4$	$4,0$

This is a matching-pennies game where in equilibrium, each player places probability $1/2$ on each pure strategy and receives expected utility 2.[3] Thus, the reduced game R becomes:

	v_1	t
u_1	$2,2$	$1,3$
s	$1,3$	$2,2$

Again, this is a matching-pennies game where in equilibrium, each player places probability $1/2$ on each action. Thus, we have discovered an equilibrium for the original game O where u_1 and v_1 are played with probability $1/2$ each, and s_1, s_2, t_1, t_2 are played with probability $(1/2) \cdot (1/2) = 1/4$ each. We now prove the technique is correct in general.

THEOREM 1. *Suppose O is reduced to R using the equilibrium $p^G_{s_i}, p^G_{t_j}$ of G. If $p^R_{u_i}, p^R_s, p^R_{v_j}, p^R_t$ constitute a Nash equilibrium of R, then setting $p^O_{u_i} = p^R_{u_i}, p^O_{s_i} = p^R_s p^G_{s_i}, p^O_{v_j} = p^R_{v_j}$, and $p^O_{t_j} = p^R_t p^G_{t_j}$ constitutes a Nash equilibrium of O.*

Proof. To show that these mixed strategies constitute a Nash equilibrium for O, we first observe that the row player has no incentive to redistribute the probability placed on strategies s_i to another strategy $s_{i'}$. That is, *given* that the row player plays one of the strategies s_i (note that given this, the probability that the row player plays a given strategy s_i is $p^R_s p^G_{s_i} / p^R_s = p^G_{s_i}$), there is no incentive to switch to another strategy $s_{i'}$. The reason is as follows. Given that the column player plays a given strategy v_j, both $p^G_{s_i}$ and $s_{i'}$ will give the same utility. Given that the column player plays one of the strategies t_j (note that given this, the probability that the column player plays a given strategy t_j is $p^R_t p^G_{t_j} / p^R_t = p^G_{t_j}$), $p^G_{s_i}$ is a best response because $p^G_{s_i}, p^G_{t_j}$ constitute a Nash equilibrium for G. As a result, to show that we have an equilibrium for O, we do not need to consider deviations to an arbitrary s_i; instead, we only need to consider deviations to the mixed strategy $p^G_{s_i}$.

Next, we observe that the expected utility for the row player of playing a given u_i is the same as in the equilibrium computed for R. The reason is as follows. The probability that the column player plays a given v_j is the same in both R and O, and so is the row player's utility for the outcome (u_i, v_j). Moreover, the probability that the column player plays one of the strategies t_j in O is p^R_t, and the expected utility for the row player of playing u_i in that case is $\sum_{j=1}^{n} p^G_{t_j} c_{ij}$, which is the same as in R.

We also observe that the expected utility for the row player of playing $p^G_{s_i}$ (that is, the row player's mixed strategy given that the row player plays one of the strategies s_i) is the same as the expected utility of playing s in the equilibrium computed for R. The reason is as follows. The probability that the column player plays a given v_j is the same in both R and O, and the row player's utility in either case is a_j. Moreover, the probability that the column player plays one of the strategies t_j in O is p^R_t, and the expected utility for the row player of playing $p^G_{s_i}$ in that case is the expected utility the row player gets in the equilibrium of G, namely, π^G_s, which is the same as the row player's utility for the outcome (s, t) in R.

But from these last two observations, and the fact that we used an equilibrium for R, it follows that the row player has no incentive to deviate to any u_i or to $p^G_{s_i}$; and from the first observation, the fact that the row player has no incentive to deviate to $p^G_{s_i}$ implies that the row player has no incentive to deviate to any s_i. Hence, the row player has no incentive to deviate; by symmetry, neither does the column player. ■

3. DETECTING WHETHER THE TECHNIQUE CAN BE APPLIED

It is easy to verify whether Condition 1 holds when the pure strategies of the game are labeled u_i, s_i, v_j, t_j. However, in general, this labeling will not be given to us. As an example, suppose we are given the following game:

	β_1	β_2	β_3
α_1	$4,0$	$1,2$	$0,4$
α_2	$0,3$	$2,2$	$2,3$
α_3	$0,4$	$1,4$	$4,0$

This game is, in fact, the same game that was used as an example in Section 2, with the rows and columns permuted $(\alpha_1 = s_1, \alpha_2 = u_1, \alpha_3 = s_2, \beta_1 = t_1, \beta_2 = v_1, \beta_3 = t_2)$. But we cannot apply the technique without knowing the labeling of the strategies as u_i, s_i, v_j, and t_j.

Hence, we need an algorithm that finds a labeling of the strategies as u_i, s_i, v_j, and t_j such that Condition 1 holds. Note that it does not matter whether we label a given strategy (*e.g.*) s_1 or s_2. That is, all we need to do is partition the row player's strategies Σ_1 into two subsets, the ones labeled u_i and the ones labeled s_i; and partition the column player's strategies Σ_2 into two subsets, the ones labeled v_j and the ones labeled t_j.

There are two trivial ways of labeling the strategies so that the condition holds: 1. Label all strategies s_i or t_j. 2. Label at most one strategy s_i, and at most one strategy t_j. These trivial labelings are not useful: the first would give us $G = O$ to recurse on, and the second would not reduce the game at all $(R = O)$. Any other labeling for which Condition 1 is satisfied, though, would give us a game G that is smaller than O to recurse on, and reduce the game to a game R that is smaller than O. Moreover, the number of rows (columns) by which R is smaller than O is equal to

[3]We emphasize that there is no restriction that the game G or R should be a zero-sum game (or any other special type of game).

the number of rows (resp. columns) in G, minus 1. Hence, our objective is to find a labeling for which the condition is satisfied, other than the two trivial ones described above.

We now show how to reformulate this problem as a Horn satisfiability problem. (Recall that a satisfiability clause is a Horn clause if it can be written as $(\neg x_1 \vee \neg x_2 \vee \ldots \vee \neg x_n \vee y)$, or equivalently, $x_1 \wedge x_2 \wedge \ldots \wedge x_n \Rightarrow y$.) Let the variable $v(\sigma)$ be *true* if σ is labeled as one of the s_i or t_j, and *false* if σ is labeled as one of the u_i or v_j. The key observation is that to satisfy Condition 1, if row player strategies α_1 and α_2 obtain different payoffs for the row player against column player strategy β, then it cannot be the case that α_1 and α_2 are both labeled as one of the s_i and β is labeled as one of the v_j—in other words, $v(\alpha_1) \wedge v(\alpha_2) \Rightarrow v(\beta)$. Similarly, if column player strategies β_1 and β_2 obtain different payoffs for the column player against row player strategy α, then $v(\beta_1) \wedge v(\beta_2) \Rightarrow v(\alpha)$.

Using this, the game described at the beginning of this section has the following Horn clauses:[4]

1. $v(\alpha_1) \wedge v(\alpha_2) \Rightarrow v(\beta_1) \wedge v(\beta_2) \wedge v(\beta_3)$,
2. $v(\alpha_1) \wedge v(\alpha_3) \Rightarrow v(\beta_1) \wedge v(\beta_3)$,
3. $v(\alpha_2) \wedge v(\alpha_3) \Rightarrow v(\beta_2) \wedge v(\beta_3)$,
4. $v(\beta_1) \wedge v(\beta_2) \Rightarrow v(\alpha_1) \wedge v(\alpha_2)$,
5. $v(\beta_1) \wedge v(\beta_3) \Rightarrow v(\alpha_1) \wedge v(\alpha_3)$,
6. $v(\beta_2) \wedge v(\beta_3) \Rightarrow v(\alpha_1) \wedge v(\alpha_2) \wedge v(\alpha_3)$.

It is straightforward to check that the only nontrivial satisfying assignment, that is, the only assignment that

- satisfies all these clauses,
- does not set all variables to *true*, and
- sets either at least two $v(\alpha_i)$ to *true* or at least two $v(\beta_j)$ to *true*;

sets $v(\alpha_1), v(\alpha_3), v(\beta_1)$, and $v(\beta_3)$ to *true* and everything else to *false*. We note that this corresponds to the labeling that we presented. The Horn clauses can be computed efficiently:

LEMMA 1. *The set of Horn clauses for a normal-form game can be computed in time* $O(|\Sigma_1|^2 \cdot |\Sigma_2| + |\Sigma_1| \cdot |\Sigma_2|^2)$.

We now show that the Horn clauses fully capture the problem, that is, the Horn clauses being satisfied is a necessary and sufficient condition for Condition 1 to be satisfied.

THEOREM 2. *A labeling of the strategies as* u_i, s_i, v_j, *and* t_j *satisfies Condition 1 if and only if it satisfies all the Horn clauses.*

Proof. First, suppose that not all of the Horn clauses are satisfied. Then (without loss of generality) there exist $\alpha_{i_1}, \alpha_{i_2}$, and β_j such that $v(\alpha_{i_1}) \wedge v(\alpha_{i_2}) \Rightarrow v(\beta_j)$, $v(\alpha_{i_1})$ and $v(\alpha_{i_2})$ are set to *true*, and $v(\beta_j)$ is set to *false*. (Alternatively the roles of the α_i and β_j could be reversed, but by symmetry we can without loss of generality restrict our attention to the first case.) That is, α_{i_1} and α_{i_2} are among the strategies s_i, β_j is among the strategies v_j, and α_{i_1} and α_{i_2} give the row player different payoffs against β_j. But then Condition 1 is not satisfied.

Now, suppose that all of the Horn clauses are satisfied. We must show, for any v_j, that any s_{i_1} and s_{i_2} give the row player the same payoff against v_j. (We must show the same for the u_i and the t_j, but this will follow by symmetry.) Suppose they do not; then one of the Horn clauses must be $v(s_{i_1}) \wedge v(s_{i_2}) \Rightarrow v(v_j)$. But this clause would then be *false*, which is contrary to our assumption. ■

A general systematic approach to finding a nontrivial satisfying assignment is to start by setting two $v(\alpha_i)$ (or two $v(\beta_j)$) to *true*, and subsequently iteratively apply the implication clauses; if this process ends without all the variables being set to *true*, we have found an assignment with the desired properties. If we do this once for every initial pair of $v(\alpha_i)$ and every initial pair of $v(\beta_j)$, then we will find an assignment with the desired properties if it exists.

For instance, in the above example, we start by setting $v(\alpha_1)$ and $v(\alpha_2)$ to *true*; then, applying the first implication, we find that all the $v(\beta_j)$ must be set to *true* as well, and hence by the last implication so must $v(\alpha_3)$, so that all variables are set to *true*. Then, we try again, starting by setting $v(\alpha_1)$ and $v(\alpha_3)$ to *true*; by the second implication, we find that $v(\beta_1)$ and $v(\beta_3)$ must also be set to *true*. Then we apply the fifth implication, but this does not set any new variables to *true*, so the process ends here, and we have found an assignment with the desired properties.

THEOREM 3. *The algorithm described above requires* $O((|\Sigma_1| + |\Sigma_2|)^4)$ *applications of a Horn clause.*

Proof. There are $O((|\Sigma_1| + |\Sigma_2|)^2)$ iterations of the outer loop (choosing which pair of variables to start with). Moreover, within each iteration, we never need to apply the same clause twice, and there are only $O((|\Sigma_1| + |\Sigma_2|)^2)$ clauses. ■

4. LIMITATIONS OF THE TECHNIQUE

One may wonder whether the technique can be extended to find Nash equilibria with special properties. For example, if we compute a Pareto optimal or social-welfare maximizing Nash equilibrium in each recursive call, will this give us a Pareto optimal or social-welfare maximizing Nash equilibrium for the original game? The following game O shows that this is not the case.

	v_1	t_1	t_2
u_1	$1,1$	$4,0$	$0,0$
s_1	$0,4$	$3,3$	$0,0$
s_2	$0,0$	$0,0$	$2,2$

The subcomponent G is as follows:

	t_1	t_2
s_1	$3,3$	$0,0$
s_2	$0,0$	$2,2$

It has multiple equilibria, but the only Pareto optimal Nash equilibrium is (s_1, t_1). Thus the reduced game R becomes:

[4]When we have Horn clauses $v(\alpha_1) \wedge v(\alpha_2) \Rightarrow v(\beta_1)$ and $v(\alpha_1) \wedge v(\alpha_2) \Rightarrow v(\beta_2)$, we aggregate them into $v(\alpha_1) \wedge v(\alpha_2) \Rightarrow v(\beta_1) \wedge v(\beta_2)$ to improve readability.

	v_1	t
u_1	$1,1$	$4,0$
s	$0,4$	$3,3$

This is a Prisoner's Dilemma game where the only equilibrium is (u_1, v_1), and thus we will compute the equilibrium (u_1, v_1) for the original game O. Unfortunately, this is not a Pareto optimal equilibrium for O, because (s_2, t_2) is also a Nash equilibrium for O. Of course, in practice, choosing a Nash equilibrium with high social welfare in each recursive call may still be a useful heuristic for finding good equilibria.

5. ALAGIU GAMES: A CLASS OF GAMES THAT CAN BE SOLVED BY APPLYING THE TECHNIQUE RECURSIVELY

In general, we may only be able to apply the technique a limited number of times, after which we need to resort to other algorithms to obtain Nash equilibria for games in the recursive calls on which the technique cannot be applied. However, in this section, we present a subclass of normal-form games such that the technique works on any of these games, and any game resulting from a recursive call is itself in the subclass as well. Thus, the technique is sufficient for finding a Nash equilibrium.

DEFINITION 1. *We say that a 2-player normal-form game is an* ALAGIU (Any Lower Action Gives Identical Utility) *game if each player's strategy set Σ_i is a subset of $\mathbb{R}$, and the game has the property that for any pure opponent strategy $\sigma_{-i} \in \Sigma_{-i}$, for any $\sigma_i^1, \sigma_i^2 \in \{\sigma_i \in \Sigma_i : \sigma_i < \sigma_{-i}\}$, we have $\pi_i(\sigma_i^1, \sigma_{-i}) = \pi_i(\sigma_i^2, \sigma_{-i})$. That is, given that a player chooses a lower strategy than the opponent, it does not matter to that player which of the lower strategies she chooses.*

Before showing how the technique can be applied to (finite) ALAGIU games, we first give some examples. The first one is very simple:

EXAMPLE 1 (MATCHING n-SIDED PENNIES). *In matching n-sided pennies, both players choose an integer in $\{1, 2, \ldots, n\}$; player 1 wins if the players' integers are the same, and player 2 wins if they are different.*

Incidentally, this example shows that ALAGIU games can have nontrivial equilibria:

PROPOSITION 1. *There exist ALAGIU games (of arbitrary size) that have only one equilibrium in which each player randomizes uniformly over all strategies.*

PROOF. We claim that matching n-sided pennies has the property. This is a zero-sum game, and thus any equilibrium strategy is a minimax strategy. For player 1, the only minimax strategy is to place probability $1/n$ on each integer, because if player 1 plays some integer with lower probability, the other player will be better off playing that integer. Similarly, for player 2, the only minimax strategy is to place probability $1/n$ on each integer, because if player 2 plays some integer with higher probability, the other player will be better off playing that integer. ■

We note that state-of-the-art equilibrium finding algorithms tend to have difficulty computing equilibria with large supports (*i.e.* in which the players randomize over many strategies). For example, the PNS algorithm [23] for finding a Nash equilibrium explicitly searches over all (exponentially many) possible supports, searching the smaller supports first. Other algorithms similarly tend to require more time on games that only have equilibria with large supports.

Matching n-sided pennies does not illustrate the full generality of ALAGIU games. The next (somewhat playful) example allows for richer structure:

EXAMPLE 2 (VENDORS ON A ONE-WAY STREET). *Say we have a hot dog vendor and an ice cream vendor that must each choose a location on a one-way street. Assume that, given that a vendor chooses a location earlier on the street than the other vendor, it does not matter to the earlier vendor where exactly she locates herself (all traffic must pass her anyway). However, it may matter to the later vendor where he locates himself. (For example, it may be bad for the ice cream vendor to immediately follow the hot dog vendor, because potential customers will have their hands full with the hot dog. In contrast, it may be good for the ice cream vendor to locate himself significantly later on the street than the hot dog vendor, to provide some dessert.) Then, this is an ALAGIU game.*

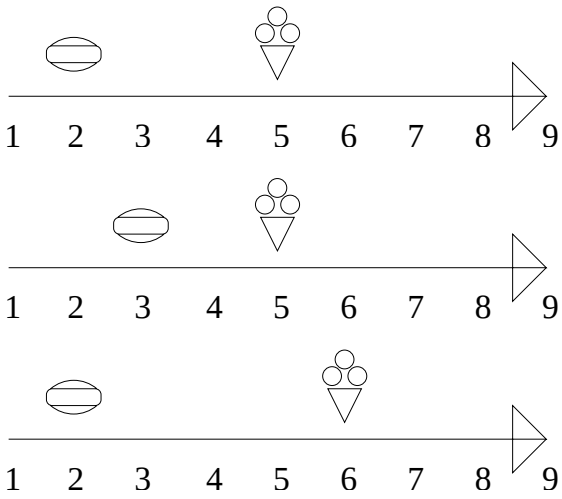


Figure 1: Three outcomes for the vendor game. Under the ALAGIU restriction, the hot dog vendor's utility must be the same for the top two outcomes. However, the ice cream vendor's utility need not be the same in any of the three outcomes.

More practically, our final example shows that ALAGIU games also encompass typical auctions.

EXAMPLE 3 (HIGHER-BIDDER-WINS AUCTIONS). *Any two-player single-item auction in which a lower bidder never wins and never pays anything is an ALAGIU game. Note that arbitrary allocation rules are allowed in case of ties. Also, arbitrary payment rules for the winning bidder are allowed, including first price and second price (but also bizarre,* e.g. *nonmonotonic, payment rules).*

Perhaps surprisingly, in contrast to auctions, *bargaining* games typically do not have the structure of ALAGIU games.[5]

[5]We thank Piotr Gmytrasiewicz for asking the question whether bargaining games are ALAGIU games.

We discuss this in more detail in the appendix, but we now move on to showing how ALAGIU games can be solved using the technique described in this paper. It turns out that any finite ALAGIU game takes one of three special forms described below, each of which satisfies Condition 1. Consider the highest strategy in the game, that is, $\sigma_M = \max\{\sigma : \sigma \in \Sigma_1 \cup \Sigma_2\}$. If only the row player has σ_M in her strategy set ($\sigma_M \in \Sigma_1 - \Sigma_2$), then the game must have the following form:

	t_1	t_2	t_3	$\cdots$	t_n
$u_1 = \sigma_M$	c_{11}, b_1	c_{12}, b_1	c_{13}, b_1	$\cdots$	c_{1n}, b_1
s_1					
s_2					
s_3			G		
$\vdots$					
s_m					

On the other hand, if only the column player has σ_M in his strategy set ($\sigma_M \in \Sigma_2 - \Sigma_1$), then the game must have the following form:

	$v_1 = \sigma_M$	t_1	t_2	t_3	$\cdots$	t_n
s_1	a_1, d_{11}					
s_2	a_1, d_{21}					
s_3	a_1, d_{31}			G		
$\vdots$	$\vdots$					
s_m	a_1, d_{m1}					

Finally, if both the row and the column player have σ_M in their strategy sets ($\sigma_M \in \Sigma_1 \cap \Sigma_2$), then the game must have the following form:

	$v_1 = \sigma_M$	t_1	t_2	t_3	$\cdots$	t_n
$u_1 = \sigma_M$	h_1, h_2	c_{11}, b_1	c_{12}, b_1	c_{13}, b_1	$\cdots$	c_{1n}, b_1
s_1	a_1, d_{11}					
s_2	a_1, d_{21}					
s_3	a_1, d_{31}			G		
$\vdots$	$\vdots$					
s_m	a_1, d_{m1}					

Our technique can be applied to every one of these three forms. Moreover, in each case the subcomponent G is still an ALAGIU game, and thus we can apply the technique recursively. The reduced game R is a 2×1, 1×2, or 2×2 game (for the three cases, respectively), so it can easily be solved.

THEOREM 4. *Using the technique described in this paper, a Nash equilibrium of a finite ALAGIU game can be computed in time $O(|\Sigma_1| \cdot |\Sigma_2|)$ (that is, in time linear in the size of the game).*

Proof. The subcomponent G that the algorithm recurses on is one row and/or column smaller than the current game. Whenever G is one row smaller (which happens $O(|\Sigma_1|)$ times), after the recursion we need to compute a weighted average of the c_{ij}, which requires $O(|\Sigma_2|)$ time. Similarly, whenever G is one column smaller (which happens $O(|\Sigma_2|)$ times), after the recursion we need to compute a weighted average of the d_{ij}, which requires $O(|\Sigma_1|)$ time. ■

6. USING THE TECHNIQUE TO COMPUTE APPROXIMATE EQUILIBRIA

It is known that *approximate* Nash equilibria can be computed faster than (we know how to compute) exact Nash equilibria [17]. In this section, we study how the technique can be used to compute approximate equilibria. Specifically, if we have found that there is no subcomponent G that satisfies the condition, we can *modify* the game so that in the modified game O', there does exist a subcomponent G' that satisfies the condition. Of course, the equilibria of O and O' are, in general, not the same, so that the equilibrium that we compute for O' using the technique is not actually an equilibrium for O. However, as we will show, if O' is close to O, then the equilibrium that we compute for O' is an approximate equilibrium for O. As our notion of distance between two games (of equal dimensions), we will use the maximum difference between two payoffs that occur in the same position in the matrices: $d(O, O') = \max_{i \in \{r,c\}} \max_{\alpha,\beta} |\pi_i^O(\alpha, \beta) - \pi_i^{O'}(\alpha, \beta)|$ (where π_r is the row player's utility function and π_c is the column player's utility function). We will use *ϵ-equilibrium* as our notion of approximate equilibrium. An ϵ-equilibrium is a pair of strategies for the agents such that neither can improve its expected payoff by more than ϵ by deviating.

LEMMA 2. *If O and O' have equal dimensions, and $d(O, O') \leq \epsilon/2$, then any equilibrium for O' is an ϵ-equilibrium for O.*

PROOF. Let p_r be the row player's mixed strategy in the computed equilibrium for O', and let p_c be the column player's mixed strategy in the computed equilibrium for O'. Let p'_r be a best response to p_c *in the original game* O. Because we computed an equilibrium, we have $\pi_r^{O'}(p_r, p_c) \geq \pi_r^{O'}(p'_r, p_c)$. Moreover, because $d(O, O') \leq \epsilon/2$, we have $\pi_r^O(p_r, p_c) \geq \pi_r^{O'}(p_r, p_c) - \epsilon/2$ and $\pi_r^{O'}(p'_r, p_c) \geq \pi_r^O(p'_r, p_c) - \epsilon/2$. Hence, $\pi_r^O(p_r, p_c) \geq \pi_r^{O'}(p_r, p_c) - \epsilon/2 \geq \pi_r^{O'}(p'_r, p_c) - \epsilon/2 \geq \pi_r^O(p'_r, p_c) - \epsilon$, as desired. By symmetry, the same holds for the column player. ■

For sufficiently large ϵ, we can always find a game O' with $d(O, O') \leq \epsilon$ for which there is a subcomponent G satisfying the condition. For example, if ϵ is larger than all of the payoffs in the game, then O' can simply be the game in which all payoffs are 0. Of course, this weakens the guarantee on how much the agents can gain by deviating. Thus, the challenge is to find a game O' that is as close as possible to O while still having a subcomponent G satisfying the condition. One greedy way of finding such a game is the following: for a given column (row), take two utilities that the row (column) player can obtain in this column (row) that are very close, and change both of them (wherever they occur in the column (row)) to the average of the two. Then, check if the modified game has a subcomponent G satisfying the condition using the technique in Section 3; if not, repeat with the modified game.

7. CONCLUSIONS

In this paper, we presented a technique for reducing a game, O, to a smaller game, R, for the purpose of computing a Nash equilibrium. This is done by computing a Nash equilibrium for a subcomponent, G, of O for which a certain

condition holds. We also showed that such a subcomponent G on which to apply the technique can be found in polynomial time (if it exists), by showing that the condition that G needs to satisfy can be modeled as a Horn satisfiability problem. We showed that the technique does not extend to computing Pareto-optimal or welfare-maximizing equilibria. We presented a class of games, which we call ALAGIU (Any Lower Action Gives Identical Utility) games, for which recursive application of (special cases of) the technique is sufficient for finding a Nash equilibrium in linear time. Finally, we discussed using the technique to compute *approximate* Nash equilibria.

Future research includes extending the techniques presented here to games with more than two players (we note, however, that from the viewpoint of complexity theory, the two-player case is as hard as the n-player case [4]). It also includes testing experimentally whether the technique is helpful in finding equilibria in games for which we do not have a theoretical result (as we do for ALAGIU games), for instance on test suites of game generators (such as GAMUT [21]). Of course, the games generated will need to have some structure in order for the technique to have a chance—for example, if the payoffs are drawn independently from an interval of real numbers, then the probability that the technique can be applied is 0, because the probability of any two payoffs being the same is 0. (However, this does not prevent us from searching for approximate equilibria using the technique, as described in Section 6.) Yet another avenue is to find other, complementary game reduction techniques that can be applied when the techniques presented here cannot. Ideally, this would lead to a portfolio of efficient techniques of which at least one can always be applied, giving us a polynomial-time algorithm for computing a Nash equilibrium.

8. REFERENCES

[1] N. A. R. Bhat and K. Leyton-Brown. Computing Nash equilibria of action-graph games. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, Banff, Canada, 2004.

[2] B. Blum, C. R. Shelton, and D. Koller. A continuation method for Nash equilibria in structured games. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

[3] X. Chen and X. Deng. 3-Nash is PPAD-complete. *Electronic Colloquium on Computational Complexity*, Report No. 134, 2005.

[4] X. Chen and X. Deng. Settling the complexity of 2-player Nash equilibrium. *Electronic Colloquium on Computational Complexity*, Report No. 150, 2005.

[5] V. Conitzer and T. Sandholm. Complexity results about Nash equilibria. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 765–771, Acapulco, Mexico, 2003.

[6] V. Conitzer and T. Sandholm. Complexity of (iterated) dominance. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 88–97, Vancouver, Canada, 2005.

[7] V. Conitzer and T. Sandholm. A generalized strategy eliminability criterion and computational methods for applying it. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 483–488, Pittsburgh, PA, USA, 2005.

[8] C. Daskalakis, P. Goldberg, and C. Papadimitriou. The complexity of computing a Nash equilibrium. *Electronic Colloquium on Computational Complexity*, Report No. 115, 2005.

[9] C. Daskalakis and C. Papadimitriou. Three-player games are hard. *Electronic Colloquium on Computational Complexity*, Report No. 139, 2005.

[10] I. Gilboa, E. Kalai, and E. Zemel. The complexity of eliminating dominated strategies. *Mathematics of Operation Research*, 18:553–565, 1993.

[11] I. Gilboa and E. Zemel. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, 1989.

[12] G. Gottlob, G. Greco, and F. Scarcello. Pure Nash equilibria: hard and easy games. In *Theoretical Aspects of Rationality and Knowledge (TARK)*, pages 215–230, Bloomington, Indiana, USA, 2003.

[13] M. Kearns, M. Littman, and S. Singh. Graphical models for game theory. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2001.

[14] D. E. Knuth, C. H. Papadimitriou, and J. N. Tsitsiklis. A note on strategy elimination in bimatrix games. *Operations Research Letters*, 7(3):103–107, 1988.

[15] C. Lemke and J. Howson. Equilibrium points of bimatrix games. *Journal of the Society of Industrial and Applied Mathematics*, 12:413–423, 1964.

[16] K. Leyton-Brown and M. Tennenholtz. Local-effect games. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

[17] R. Lipton, E. Markakis, and A. Mehta. Playing large games using simple strategies. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 36–41, San Diego, CA, 2003.

[18] R. D. Luce and H. Raiffa. *Games and Decisions*. John Wiley and Sons, New York, 1957. Dover republication 1989.

[19] R. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, Cambridge, 1991.

[20] J. Nash. Equilibrium points in n-person games. *Proc. of the National Academy of Sciences*, 36:48–49, 1950.

[21] E. Nudelman, J. Wortman, K. Leyton-Brown, and Y. Shoham. Run the GAMUT: A comprehensive approach to evaluating game-theoretic algorithms. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, New York, NY, USA, 2004.

[22] C. Papadimitriou. Algorithms, games and the Internet. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pages 749–753, 2001.

[23] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a Nash equilibrium. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 664–669, San Jose, CA, USA, 2004.

[24] T. Sandholm, A. Gilpin, and V. Conitzer. Mixed-integer programming methods for finding Nash

equilibria. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 495–501, Pittsburgh, PA, USA, 2005.

[25] R. Savani and B. von Stengel. Exponentially many steps for finding a Nash equilibrium in a bimatrix game. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, 2004.

APPENDIX

In this appendix, we show that, in contrast to auctions, bargaining games typically do not have the structure of ALAGIU games, or even the structure required for the technique to be useful. Consider a dollar-splitting game in which each of two agents must simultaneously ask for a fraction of the dollar. If the fractions that the agents ask for sum to at most a dollar, they each receive their fraction; otherwise, neither agent receives anything. We note that this game *appears* to have somewhat similar structure to ALAGIU games, in the sense that if an agent asks for an amount that is too high, then it does not matter to the agent by how much the amount is too high. However, there is a subtle difference: informally, in (for example) auctions, bids are too low when the other bidder's bid is high, whereas in bargaining games such as this one, demands are too high if the other bargainer's demand is *also* high. We now show formally that the technique is in general not useful on bargaining games.

For simplicity, suppose that the agents can only ask for \$0.25, \$0.50, or \$0.75. Then the game is the following.

	\$0.25	\$0.50	\$0.75
\$0.25	$0.25, 0.25$	$0.25, 0.50$	$0.25, 0.75$
\$0.50	$0.50, 0.25$	$0.50, 0.50$	$0, 0$
\$0.75	$0.75, 0.25$	$0, 0$	$0, 0$

We will use the technique from Section 3 to show that there is no subcomponent G on which to apply the technique. Denoting by α_x agent 1's strategy of demanding x, and by β_x agent 2's strategy of demanding x, we obtain the following clauses:

1. $v(\alpha_{\$0.25}) \wedge v(\alpha_{\$0.50}) \Rightarrow v(\beta_{\$0.25}) \wedge v(\beta_{\$0.50}) \wedge v(\beta_{\$0.75})$,
2. $v(\alpha_{\$0.25}) \wedge v(\alpha_{\$0.75}) \Rightarrow v(\beta_{\$0.25}) \wedge v(\beta_{\$0.50}) \wedge v(\beta_{\$0.75})$,
3. $v(\alpha_{\$0.50}) \wedge v(\alpha_{\$0.75}) \Rightarrow v(\beta_{\$0.25}) \wedge v(\beta_{\$0.50})$,
4. $v(\beta_{\$0.25}) \wedge v(\beta_{\$0.50}) \Rightarrow v(\alpha_{\$0.25}) \wedge v(\alpha_{\$0.50}) \wedge v(\alpha_{\$0.75})$,
5. $v(\beta_{\$0.25}) \wedge v(\beta_{\$0.75}) \Rightarrow v(\alpha_{\$0.25}) \wedge v(\alpha_{\$0.50}) \wedge v(\alpha_{\$0.75})$,
6. $v(\beta_{\$0.50}) \wedge v(\beta_{\$0.75}) \Rightarrow v(\alpha_{\$0.25}) \wedge v(\alpha_{\$0.50})$.

It is straightforward to see that this set of clauses only has trivial solutions (all variables set to *true*, or at most one per player set to *true*). Hence the technique cannot be applied on this game (which also implies that the game is not an ALAGIU game).

Information-theoretic approaches to branching in search*

Andrew Gilpin
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
gilpin@cs.cmu.edu

Tuomas Sandholm
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
sandholm@cs.cmu.edu

ABSTRACT

Deciding what to branch on at each node is a key element of search algorithms. We present four families of methods for selecting what question to branch on. They are all information-theoretically motivated to *reduce uncertainty in remaining subproblems*. In the first family, a good variable to branch on is selected based on lookahead. In real-world procurement optimization, this entropic branching method outperforms default CPLEX and strong branching. The second family combines this idea with strong branching. The third family does not use lookahead, but instead exploits features of the underlying structure of the problem. Experiments show that this family significantly outperforms the state-of-the-art branching strategy when the problem includes indicator variables as the key driver of complexity. The fourth family is about branching using carefully constructed linear inequality constraints over *sets* of variables.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms

Keywords

search, branching heuristics, entropic branching, mixed-integer programming

1. INTRODUCTION

Search is a fundamental technique for problem solving in AI and operations research (OR). At a node of the search tree, the search algorithm poses a question, and then tries out the different answers (which correspond to the branches emanating from that node). Many different ways of deciding which question to pose (branching strategies) have been studied. In this paper we introduce a new paradigm for developing branching strategies, employing information theory as the principle that guides the development. In the context of solving integer programs, we develop four high-level families of strategies, and show that some of those families significantly improve performance over existing strategies. All four strategies aim to reduce the uncertainty in the current subtree.

The rest of this section introduces the necessary background material. Each of the following four sections introduces one high-level family of new strategies.

1.1 Integer programming

One of the most important computational problems in CS and OR is integer programming. Applications of integer programming include scheduling, routing, VLSI circuit design, and facility location [4]. Integer programming is the problem of optimizing a linear function subject to linear constraints and integrality constraints on some of the variables. While (the decision version of) MIP is $\mathcal{NP}$-complete, there are many sophisticated techniques that can solve very large instances in practice.

1.2 Strong branching

At every node of a tree search, the search algorithm has to decide what question to branch on (and thus what the children of the node will be). One commonly used method is to branch on the variable whose value is furthest from being integral [6]. In other words, the question is: "What should the value of this variable be?", and the children correspond to different answers to this question. For many problem instances, this approach is effective.

A more sophisticated approach, which is better suited for certain hard problem instances, is *strong branching* [1]. The algorithm performs a one-step lookahead for each variable that is non-integral in the LP at the node. The one-step lookahead computations solve the LP relaxation for each of the children.

2. ENTROPIC LOOKAHEAD FOR VARIABLE SELECTION

The main idea behind our approaches is to treat the fractional portion of integer-constrained variables as probabilities, indicating the probability with which we expect the variable to be greater than its current value in the optimal solution. Using this reasoning, our branching heuristic is designed to guide the search so as to remove uncertainty from the partial solution in the current search node. For any search algorithm, the beginning of the search is where there is the most uncertainty about the optimal solution to

*This was work was funded by, and conducted at, CombineNet, Inc., Fifteen 27th St., Pittsburgh, PA 15222.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

the problem. When the search is complete, there is zero uncertainty as the optimal solution has been found. Our branching heuristic is designed to direct the search from the root node (with high uncertainty) to the optimal node (with zero uncertainty) in as direct a way as possible. We determine the variable to branch on using one-step lookahead as in strong branching. The difference is that instead of examining the objective values of potential child nodes, we examine the remaining *uncertainty* in potential child nodes, choosing a variable to branch on that yields children with the least uncertainty.

While there has been much work on developing branching heuristics, to our knowledge this is the first work that takes an information-theoretic approach to guiding the search process.

Before we describe our approach to branching in search, it will be useful to define how we can quantify the "uncertainty" of a partial solution. For this, we borrow some definitions from information theory, from which the primary contribution is the notion of *entropy*, which measures the amount of uncertainty in a random event. Given an event with two outcomes (say 0 or 1) where outcome 0 occurs with probability x, we can compute the entropy of the event from the probability of each outcome occurring:

$$e(x) = \begin{cases} -x\log_2 x - (1-x)\log_2(1-x) : 0 < x < 1 \\ 0 : x \in \{0,1\}. \end{cases}$$

Now we are ready to present our variable selection method:

ALGORITHM 1. Entropic Branching (EB)

1. *candidates* $\leftarrow \{i \mid x_i$ fractional$\}$
2. *For each* $i \in$ *candidates:*
   (a) $x_f \leftarrow x_i - \lfloor x_i \rfloor$
   (b) *Let* $\hat{x}^l$ *be solution vector of LP with* $\hat{x}^l_i \leq \lfloor x_i \rfloor$.
   (c) *Let* $\hat{x}^u$ *be solution vector of LP with* $\hat{x}^u_i \geq \lceil x_i \rceil$.
   (d) $entropy(x_i) \leftarrow \sum_{j=1}^{n}(1-x_f)e\left(\hat{x}^l_j\right) + x_f e\left(\hat{x}^u_j\right)$
3. $i^* \leftarrow \text{argmin}_{i \in candidates}\ entropy(x_i)$
4. *Return* i^*

EB is a *general* method, usable on any MIP as it does not make any assumptions about the underlying model for the problem on which it is used. EB (and SB) could be modified to perform more than one-step lookahead in a straightforward way. However, we suspect that this increased computational effort would not reduce tree size enough to be beneficial.

For illustrative purposes, there is an interesting (though not exact) analogy between our entropic lookahead method for question selection at search nodes and algorithms for decision tree induction. In most recursive decision tree induction algorithms, a question is inserted at a leaf that results in the greatest information gain. Similarly, in search, by choosing questions whose children have the least entropy, we are creating children that in a sense also result in the greatest information gain.

3. COMBINING SB AND EB

From the descriptions, it is clear that SB and EB are computed quite similarly. A natural question arises: can we develop a hybrid approach combining the strengths of both without using significantly more computational resources? In this section we answer this question in the affirmative by introducing a second family of variable selection strategies. In this family, SB and EB are hybridized in different ways. Each of these methods requires only a small amount of additional computation compared to performing *only* SB or EB (because the same lookahead with the same child LP solves is used). We classify our hybrid approaches into two categories: tie-breaking and combinational.

3.1 Tie-breaking methods

In this approach, we first perform the SB computations, but instead of simply branching on the variable with best score, we break ties using an entropy computation. Since we have already computed the LP relaxations for each branch, computing the entropy is a negligible computational cost (relative to computing the LP relaxations). In addition to breaking exact ties, we can also consider two variables having SB scores within x% of each other as tied. For a given percentage x, we refer to this method as TB(x%).

3.2 Combinational methods

We present two methods of combining information from SB and EB in order to compute a single score for a variable. The variable with the best score will then be branched on.

The first method, RANK, performs the computation for SB and EB first. (Again, these computations are performed simultaneously at little additional cost beyond doing either SB or EB alone.) Define $rank_{SB}(x_i)$ to be the rank of variable x_i in terms of its SB score (*i.e.*, the variable with the largest score would have rank 1, the variable with the second-largest score would have rank 2, and so on). Similarly, define $rank_{EB}(x_i)$ to be the rank of variable x_i in terms of its EB entropy. Then, for each variable, we let $rank(x_i) = rank_{SB}(x_i) + rank_{EB}(x_i)$ and choose the variable x_i with the smallest rank.

The second method, COMB(α,$1-\alpha$), computes a convex combination of the SB score (with weight α) and the current entropy minus the EB score (with weight $1-\alpha$). It then selects the variable with the highest final score.

3.3 Experiments on these lookahead-based branching strategies

We conducted a host of experiments with the search methods described above, both on MIPLIB and real-world procurement optimization instances. To briefly summarize, EB and SB performed comparably on the MIPLIB data, while EB was 29.5% faster on the real-world procurement optimization instances. All of our experimental results are available in a separate technical report [3].

4. ENTROPIC LOOKAHEAD-FREE VARIABLE SELECTION

In this section we introduce a third family of branching strategies, again within the entropy-based branching paradigm. This method is computationally less expensive than the methods we have presented thus far because it does not use lookahead. However, it does require an advanced knowledge of the structure of the problem.

The problem with which we experiment is motivated by a real-world electronic commerce application: a combinatorial procurement auction (aka. reverse auction) where the buyer specifies the maximum number of winning suppliers [2, 5].

DEFINITION 1. *(Combinatorial procurement auction with*

maximum winners constraint)
Let $\mathcal{M} = \{1, \ldots, m\}$ be the m goods that the buyer wishes to procure (the buyer wants at least one unit of each good). Let $\mathcal{S} = \{1, \ldots, s\}$ be the s suppliers participating in the procurement and let $\mathcal{B} = \{B_1, \ldots, B_n\}$ be the bids, where $B_i = \langle G_i, s_i, p_i \rangle$ indicates that supplier s_i can supply the bundle of goods $G_i \subseteq \mathcal{M}$ at price p_i. Finally, the buyer indicates the maximum number of winners, k. The winner determination problem is to identify the winning bids so as to minimize the buyer's cost subject to the constraints that the buyer's demand is satisfied and that the maximum number of winners constraint is satisfied.

We can very naturally formulate the above generalized winner determination problem as a MIP:

$$\begin{array}{lll} \text{minimize} & \sum_{i=1}^{n} p_i x_i & \\ \text{such that} & \sum_{i \mid j \in G_i} x_i \geq 1 & j \in \{1, \ldots m\} \\ & \sum_{i \mid s_i = j} x_i - m y_j \leq 0 & j \in \{1, \ldots, s\} \\ & \sum_{j=1}^{s} y_j - k \leq 0 & \\ & x_i \in \{0, 1\} & i \in \{1, \ldots, n\} \\ & y_j \in \{0, 1\} & j \in \{1, \ldots, s\} \end{array}$$

This problem is $\mathcal{NP}$-complete, even if the bids are on single items only [5]. The formulation is typical of a common class of problems in which binary "indicator" variables—the y_j variables in the formulation above—are used to model logical connectives [4] (for example an implication that if an indicator is zero, then each of a set of other variables has to be zero). Constraints that state that at most (or exactly) k variables from a set of variables can be nonzero are an important special case. Typically, the LP relaxation gives poor approximate values for the indicator variables, and as we show, the branching method that we propose helps significantly to address this problem.

4.1 Branching strategy

The hardness in this problem comes primarily from determining the winning set of suppliers. In terms of the above MIP, we need to determine the y_j values. The main idea in our branching strategy for this problem is to branch on y_j values that correspond to suppliers about which we are most uncertain. But rather than deriving this uncertainty from the variable y_j (for which the LP gives very inaccurate values), we derive it from the variables corresponding to supplier $j's$ bids. The branching strategy works as follows. For each supplier j where $y_j \notin \{0, 1\}$, compute

$$entropy(j) = \sum_{i \mid s_i = j} e(x_i)$$

and branch on the variable $y_{j'}$ where

$$j' = \text{argmin}_j \; entropy(j).$$

This strategy does not use lookahead: it only uses the LP values of the current search node. We call this branching strategy *Indicator Entropic Branching (IEB)*. Our experimental results (available in a longer version of this paper [3]) for this problem indicate that IEB significantly outperforms the state-of-the-art branching strategy.

5. ENTROPIC LOOKAHEAD FOR MULTI-VARIABLE BRANCHES

In this section we introduce a fourth family of methods for determining good questions to branch on. In EB, we performed a one-step lookahead for each non-integral variable. Here, we generalize entropic lookahead beyond branching on variables. It is known that in integer programming one can branch on the sum of the values of a *set* of variables. For example, if the LP relaxation at the current search node has $x_i = 0.2$ and $x_j = 0.6$, we could set one branch to be $x_i + x_j \leq 0$ and the other branch to be $x_i + x_j \geq 1$. In general, given a set $\mathcal{X}$ of variables and the current LP relaxation solution $\hat{x}$, we can let $k = \lfloor \sum_{i \in \mathcal{X}} \hat{x}_i \rfloor$ and we can generate the following two branches:

$$\sum_{i \in \mathcal{X}} x_i \leq k \quad \text{and} \quad \sum_{i \in \mathcal{X}} x_i \geq k + 1.$$

No other value of k is worth considering; any other integer value would cause one child to be exactly the same as the node. Then, instead of branching on the variable with the smallest amount of entropy in its child nodes, we select the *set* of variables for branching that results in the smallest amount of entropy in the two child nodes. In step 2.d of EB, we weighted the entropy of each child by the probability that we expect the optimal solution to occur in each child. In the multi-variable case, we still perform this weighting, but it is more complicated since the probability of each branch depends on several variables.

Preliminary experimental results indicate that this branching scheme is successful at reducing the size of the search tree, but the extra time at each node makes it not worth it.

6. CONCLUSIONS

We introduced a new paradigm for branch selection in search based on an information-theoretic approach. In the beginning of a search, there is the most uncertainty about the optimal solution. When the search is complete, there is zero uncertainty. Using this observation, we developed four families of methods for selecting what question to branch on at a search node so as to reduce the amount of uncertainty in the search process. All four of our families of methods are information-theoretically motivated to reduce remaining entropy.

7. REFERENCES

[1] David Applegate, Robert Bixby, Vasek Chvátal, and William Cook. The traveling salesman problem. Technical report, Rutgers University, DIMACS, 1994.

[2] Andrew J Davenport and Jayant Kalagnanam. Price negotiations for procurement of direct inputs. Technical Report RC 22078, IBM, May 2001.

[3] Andrew Gilpin and Tuomas Sandholm. Information-theoretic approaches to branching in search. Technical Report CMU-CS-06-102, 2006.

[4] George Nemhauser and Laurence Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1999.

[5] Tuomas Sandholm and Subhash Suri. Side constraints and non-price attributes in markets. In *IJCAI-2001 Workshop on Distributed Constraint Reasoning*, pages 55–61, Seattle, WA, 2001. To appear in *GEB*.

[6] Laurence Wolsey. *Integer Programming*. John Wiley & Sons, 1998.

The Complexity of Model Checking Concurrent Programs against CTLK Specifications

Alessio Lomuscio, Franco Raimondi
Department of Computer Science
University College London – London, UK
{a.lomuscio,f.raimondi}@cs.ucl.ac.uk

ABSTRACT

This paper presents complexity results for model checking concurrent programs against temporal-epistemic formulae. We apply these results to evaluate the complexity of verifying programs by means of two model checkers for multi-agent systems: MCMAS and Verics.

General Terms

Algorithms; Theory; Verification

Keywords

Model checking multi-agent systems; Complexity

1. PRELIMINARIES

1.1 Temporal logics and model checking

The language of Computational Tree Logic (CTL, [7, 4]) is defined over a set of atomic formulae $AP = \{p, q, \dots\}$ as follows:

$$\varphi ::= p | \neg\varphi | \varphi \vee \varphi | EX\varphi | E[\varphi U \psi] | EG\varphi.$$

Other temporal operators to express eventuality and universality can be derived in standard way. We refer to [4] for more details.

CTL formulae are interpreted in *Kripke models*. A Kripke model M for CTL is a tuple $M = (S, R, V, I)$ where S is a set of states, $R \subseteq S \times S$ is a serial transition relation (the *temporal* relation), $V : S \rightarrow 2^{AP}$ is an evaluation function, and $I \subseteq S$ is a set of initial states. *Satisfiability* of a CTL formula φ at a state $s \in S$ in a given CTL model M is defined in a standard way, see [4] for more details.

Model checking is the problem of establishing whether or not a formula φ is satisfied on a given model M. While this check may be defined for a model M of any logic, traditionally the problem of model checking has been investigated mainly for temporal logics. In practical instances, when using model checkers, states and relations in temporal models are not listed *explicitly*. Instead, a *compact description* is usually given for a model M. Various techniques are available to give these specifications, a popular being *concurrent programs* [6]. Concurrent programs offer a suitable framework to investigate the computational complexity of model checking when compact representations are used because, as exemplified in Section 3, various techniques can be reduced as accepting concurrent programs as input.

Formally, a program in this setting is a tuple $D =< AP, AC, S, \ , s^0, L >$, where AP is a set of atomic propositions, AC is a set of actions, S is a set of states, $\ : S \times AC \rightarrow S$ is a transition function, s^0 is an initial state, and $L : S \rightarrow 2^{AP}$ is a valuation function. Given n programs $D_i =< AP_i, AC_i, S_i, \ _i, s_i^0, L_i >$ ($i \in \{1, \dots, n\}$), a concurrent program $D_C = < AP_C, AC_C, S_C, \ _C, s_C^0, L_C >$ is defined as the parallel composition of the n programs D_i, as follows: $AP_C = \cup_{1 \leq i \leq n} AP_i$; $AC_C = \cup_{1 \leq i \leq n} AC_i$; $S_C = \prod_{1 \leq i \leq n} S_i$; $(s, a, s') \in \ _C$ iff (i) $\forall 1 \leq i \leq n$, if $a \in AC_i$, then $(s[i], a, s'[i]) \in \ _i$, where $s[i]$ is the i-th component of a state $s \in S$; (ii) if $a \notin AC_i$, then $s[i] = s'[i]$; $L_C(s) = \cup_i L_i(s[i])$ (in the remainder, we will drop the subscript C when this is clear from the context). CTL formulae can be interpreted on a (concurrent) program D by using the standard Kripke semantics for CTL formulae. By slight abuse of notation, we will sometimes refer to the programs D_i and to D with the term "Kripke models".

Traditionally, the complexity of temporal logics model checking has been investigated assuming that models are given explicitly. Following this approach, the complexity is given as a function of the size of the model and of the formula being checked. In the case of CTL, the problem of model checking is P-complete [3]. Instead, the complexity of model checking concurrent programs against CTL specifications is investigated in [6] where it is shown that model checking is a PSPACE-complete problem.

1.2 CTLK

CTLK is an extension of CTL with epistemic operators $K_i, i \in \{i, \dots, n\}$ [5]. The formula $K_i\varphi$ expresses the fact that agent i *knows* φ. **CTLK** formulae may be interpreted in a Kripke model $M = (W, R_t, \sim_1, \dots, \sim_n, V)$ where W is a set of states, $R_t \subseteq S \times S$ is a serial transition relation (the *temporal* relation), $\sim_i \subseteq S \times S$ are equivalence relations (the *epistemic* relations), and $V : S \rightarrow 2^{AP}$ is an evaluation function for a given set AP of atomic propositions. Formulae are interpreted in a standard way, by extending the interpretation of CTL formulae with the following:

$$M, w \models K_i\varphi \quad \text{iff} \quad \text{for all } w' \in W, w \sim_i w' \text{ implies } M, w' \models \varphi,$$

CTLK formulae can be interpreted in concurrent programs as well: the temporal operators of **CTLK** are interpreted as in [6], while epistemic operators are evaluated by defining epistemic accessibility relations based on the equality of the components of the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

states of a concurrent program (a similar approach can be found in [5]).

2. COMPLEXITY OF MODEL CHECKING CONCURRENT PROGRAMS

In this section we present a proof for the PSPACE-completeness of the problem of model checking concurrent programs against **CTLK** specifications. The following two lemmas will be used:

LEMMA 1. *Given a Kripke model $M = (S, R, V, I)$ for CTL, a state $s \in S$, and a formula φ, $M, s \models EG\varphi$ iff there exists a sequence of states π starting from s of length $|\pi| \geq |M|$ s.t. $M, \pi_i \models \varphi$ for all $0 \leq i \leq |M|$.*

LEMMA 2. *Given a Kripke model $M = (S, R, V, I)$ for CTL, a state $s \in S$, and two formulae φ and ψ, $M, s \models E[\varphi U \psi]$ iff there exists a sequence of states π starting from s s.t. $M, \pi_i \models \psi$ for some $i \leq |M|$, and $M, \pi_j \models \varphi$ for all $0 \leq j < i$.*

THEOREM 1. *Model checking concurrent programs against* **CTLK** *specifications is a PSPACE-complete problem.*

PROOF. Given a **CTLK** formula and n programs D_i defining a concurrent program D , we define a deterministic, polynomially-space bounded Turing machine T that halts in an accepting state iff $\neg\varphi$ is satisfiable in D (i.e., iff there exists a state $s \in S$ such that $D, s \models \neg\varphi$). Based on this, we conclude that the problem of model checking is in co-PSPACE. As deterministic complexity classes are closed under complement, we conclude that the problem is PSPACE-complete (the lower bound being given by the complexity of model checking CTL in concurrent programs). T is a multi-string Turing machine whose inputs are the n programs D_i and the formula φ. T operates "inductively" on the structure of the formula φ (see also [2] for similar approaches), by calling other machines ("sub-machines") dealing with a particular logical operator only. The input of T includes the states of the program S_i ($1 \leq i \leq n$), the transition relations, the evaluation functions and all the other input parameters of each $_i$. This information can be stored on a single input tape, separated by appropriate delimiters, together with the formula φ. T returns "yes" iff there exists a state $s \in S$ such that $D, s \models \neg\varphi$. The machine T iterates over the set of states $s = (s_1, \ldots, s_n)$ and checks whether or not $\neg\varphi$ holds in one of these. If a state is found such that $D, s \models \neg\varphi$, then the machine halts in a "yes" state; if the machine loops over all the states without finding a state satisfying $\neg\varphi$, then T halts in a "no" state.

The REACHABILITY algorithm [9] can be used here to check reachability from initial states; notice that only a polynomial amount of space is needed to store states, as they are the product of states of D_i.

A "main" procedure SATISFIABLE can be defined, which operates recursively on the structure of the formula by calling one of the machines described below. Each machine accepts a state s and a formula, and returns either "NO" (the formula is false at s) or "YES" (the formula is true at s). Notice that each machine can call any of the other machines. The following is a description of the formula-specific machines that may be called by SATISFIABLE:

- The machine T_p for atomic formulae simply checks whether or not a `state` is in L(`state`), where the evaluation L is obtained from the evaluations for each program in the input string; if the proposition is true at `state`, then T_p returns YES, otherwise it returns NO.
- The machine $T_\neg$ for formulae of the form $\psi = \neg\varphi$ calls the appropriate machine for φ and returns the opposite value.
- The machine $T_\vee$ for the disjunction $\psi = \psi' \vee \psi''$ first calls the machine for ψ'. If the result is yes, it outputs yes, otherwise it outputs the result of the machine for ψ''.
- The machine T_{EX} accepts a formula φ and a state as input; the machine iterates over the set of states and for each state it checks whether the state is reachable from the input state; if this is the case, then T_{EX} checks whether or not φ is satisfied. If T_{EX} finds such a state, then it halts in a YES state; otherwise, if no reachable state satisfying φ can be found T_{EX} terminates in a NO state. Notice that this machine uses a polynomial amount of space: the space required to store the value of the state.
- The machine T_{EU} for formulae of the form $\psi = E[\varphi' U \psi'']$ is as follows:

```
T_EU(φ,  ,state) {
 state2 = (1, 1, ..., 1);
 repeat
  if SATISFIABLE(  ,state2) then
   if ( state == state2) return YES;
   else
    if ( PATH(state,state2,φ,N) ) then
     return YES;
    else
     move to next state2;
    end if;
   end if;
  else
   move to next state2;
  end if
 until last state2;
 return NO;
}
```

  The machine T_{EU} accepts two formulae and a state. The machine checks whether ψ holds in `state2` and whether `state` and `state2` are the same state. If this is the case, then the machine halts in a YES state. Otherwise, the machine checks whether or not there is a sequence of states from `state` to `state2` such that φ holds along the sequence. This check is performed by the procedure PATH(`state`,`state2`,φ,N), which returns YES if there is such a path, of length at most 2^N. By Lemma 2, we take N to be the logarithm of the size of the model. A recursive algorithm to solve PATH is presented in [9]; this algorithm employs at most space proportional to N, and it can be extended by adding a simple check for the satisfiability of φ. As there can be at most $|\varphi|$ checks, PATH uses at most $O(n \cdot |\{D_i\}_{i\in\{1,\ldots,n\}}| \cdot |\varphi|)$ space (i.e. it operates in PSPACE).
- Based on Lemma 1, a non-deterministic machine NT_{EG} can be defined to guess a sequence of states of length greater than $|\{D_i\}_{i\in\{1,\ldots,n\}}|$ in which φ holds. When (and if) such a sequence is found, the machine returns yes (notice that this machine uses a polynomial amount of space and always halts). As NPSPACE=PSPACE [9], it is possible to build a deterministic machine T_{EG} in PSPACE that returns yes iff there exists a sequence of states of length greater than $|\{D_i\}_{i\in\{1,\ldots,n\}}|$ in which φ holds.
- The machine T_K accepts a formula φ, an index i, and and a state as input; this machine operates similarly to T_{EX}, but uses epistemic relations instead of temporal relations.

Each of the machines above uses at most a polynomial amount of space, and there are at most $|\varphi|$ calls to these machines in each run of T. Thus, T uses a polynomial amount of space. □

This proof differs from the proof of PSPACE-completeness for model checking concurrent programs against CTL specifications presented in [6]. The authors of [6] investigate the complexity of various automata and apply their results to the verification of branching time logics. This technique cannot be easily extended to the verification of temporal *and* epistemic modalities. Thus, the proof above provides an alternative proof of the upper bounds for model checking CTL in concurrent programs, which can be easily extended to CTLK.

3. APPLICATIONS

3.1 The complexity of model checking MCMAS programs

MCMAS [11] is a symbolic model checker for interpreted systems. Interpreted systems [5] provide a semantics for temporal and epistemic operators, based on a system of *agents*. Each agent is characterised by a set of local states, by a set of actions, by a protocol specifying the actions allowed in each local state, and by an evolution function for the local states. MCMAS accepts as input a description of an interpreted system and builds a *symbolic* representation of the model by using Ordered Binary Decision Diagrams. We refer to [5, 10, 11] for more details. An interpreted systems described in MCMAS can be reduced to a concurrent program: each agent can be associated with a program $D_i =< AP_i, AC_i, S_i, \ _i, s_i^0, L_i >$, where AC_i is the set of actions for agent i, S_i is the set of local states for agent i, and the evolution function $\ _i$ is the one provided for the agent.

Conversely, the problem of model checking a formula φ in the parallel composition of n programs $D_i =< AP_i, AC_i, S_i, \ _i, s_i^0, L_i >$ can be reduced to an MCMAS program. Indeed, it suffices to introduce an agent for each program, whose local states are S_i and whose actions are AC_i. The transition conditions for the agent can be taken to be $\ _i$, augmented with the condition that a transition between two local states is enabled if all the agents including the same action in AC_i perform the transition labelled with the particular action.

Therefore, we conclude that the problem of model checking MCMAS programs is a P-complete problem.

It is worth noticing that the actual implementation of MCMAS requires, in the worst case, an exponential space to perform verification. Indeed, MCMAS uses OBDDs, and it is known [1] that for certain problems OBDDs may have a size which is exponential in the number of variables used, irrespective of the ordering of variables chosen.

3.2 The complexity of model checking Verics programs

Verics [8] is a tool for the verification of various types of timed automata and for the verification of **CTLK** properties in multi-agent systems. In this section we consider only the complexity of verification of **CTLK** properties in Verics using un-timed automata.

A multi-agent system is described in Verics by means of a network of (un-timed) automata: each agent is represented as an automaton, whose states correspond to local states of the agent. In this formalism a single set of action is present, and automata synchronise over common actions.

The reduction from Verics code to concurrent programs is straightforward: each automaton is a program D_i and no changes are required for the parallel composition, and similarly a concurrent program can be seen as a network of automata. Thus, we conclude that the problem of model checking un-timed Verics programs is PSPACE-complete.

Notice that the actual implementation of Verics performs verification by reducing the problem to a satisfiability problem for propositional formulae. Similarly to MCMAS, this reduction may lead to exponential time and space requirements in the worst case.

Acknowledgements

We gratefully acknowledge Mike Wooldridge for the valuable comments on an earlier draft of this paper.

Note

A preliminary version of this paper appears in the proceedings of the workshop *Concurrency Specification and Programming* (CS&P 2005), Ruciane Nida, Poland.

4. REFERENCES

[1] R. E. Bryant. On the complexity of VLSI implementations and graph representations of boolean functions with application to integer multiplication. *IEEE Trans. Comput.*, 40(2):205–213, 1991.

[2] Allan Cheng. Complexity results for model checking. Technical Report RS-95-18, BRICS - Basic Research in Computer Science, Department of Computer Science, University of Aarhus, February 1995.

[3] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications: A practical approach. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[4] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

[5] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, 1995.

[6] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.

[7] K. McMillan. *Symbolic model checking: An approach to the state explosion problem*. Kluwer Academic Publishers, 1993.

[8] W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, and M. Szreter. VerICS 2004: A model checker for real time and multi-agent systems. In *Proceedings of the International Workshop on Concurrency, Specification and Programming (CS&P'04)*, volume 170 of *Informatik-Berichte*, pages 88–99. Humboldt University, 2004.

[9] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[10] F. Raimondi and A. Lomuscio. MCMAS - A tool for verification of multi-agent systems. http://www.cs.ucl.ac.uk/staff/f.raimondi/MCMAS/.

[11] F. Raimondi and A. Lomuscio. Automatic verification of multi-agent systems by model checking via OBDDs. *Journal of Applied Logic*, 2005. To appear in Special issue on Logic-based agent verification.

Cooperation and coordination

Distributed Navigation in an Unknown Physical Environment *

Arnon Gilboa, Amnon Meisels
Department of Computer Science
Ben-Gurion University of the Negev
Beer-Sheva, 84-105, Israel
{gilboaar,am}@cs.bgu.ac.il

Ariel Felner
Department of Information Systems Engineering
Ben-Gurion University of the Negev
Beer-Sheva, 84-105, Israel
felner@bgu.ac.il

ABSTRACT

We address the problem of navigating from an initial node to a goal node by a group of agents in an unknown physical environment. In such environments mobile agents must physically move around to discover the existence of nodes and edges. We assume that agents communicate by exchanging messages about their discoveries, their current locations and their intended plans. We also assume that an agent can only communicate with a predefined set of *neighboring* agents. A distributed algorithm, which is run independently by each agent, is presented. Given the current knowledge of the agent about the environment and the positions and intentions of other agents, the algorithm instructs the agent where to go next. An experimental evaluation of the algorithm is presented, with constrained and liberal neighborhood schemes. Results show that it is more beneficial to have a constrained neighborhood scheme because with this scheme the distributed intelligent behavior of agents generates a spread of knowledge throughout the environment more efficiently. Agents reach the goal node fast and the length of the path that they find is very close to that of the optimal path.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed AI—*Intelligent Agents*

General Terms

Algorithms, Experimentation

Keywords

Mobile agents, Distributed navigation

*Supported by the Lynn and William Frankel center for Computer Sciences and the Paul Ivanier Center for Robotics and Production Management.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

We address the problem of navigation from an initial node to a goal node in an unknown physical environment, by a group of agents. To abstract away from problems of real robots and obstacle avoidance [2, 3] and to concentrate on navigation, we use the notion of a partially known physical graph [4]. The environment is represented by a planer graph where nodes are physical points and a weight of an edge is the Euclidean distance between the two nodes it connects. In such environments, only when an agent physically visits a node, the weights of outgoing edges and the adjacent nodes are revealed. Therefore, agents must move around the graph in order to find information about other nodes and about weights of edges.

1.1 Navigation versus optimal path finding

Let us start by making a distinction between navigating to a node and finding the optimal path from an initial node to a goal node. When navigating, the task is to reach the goal node as soon as possible. At every step the task of the navigator is to find a path from its current node to the goal node. Once the navigator reaches the goal, the task is accomplished. Many times, the navigator did not travel via the shortest path between the initial and goal node and does not even know this shortest path upon arriving to the goal node. Much of the literature on navigation deals with physical mobile robots that move in a real environment. The published research usually focuses on the issue of assisting the robot to recognize physical objects in its environment. We refer the reader to [2, 3] which contain extensive surveys of various approaches and state of the art techniques of robot navigation . The present work is different, as it represents the physical world by a discrete abstract graph which is initially unknown.

Finding an optimal (shortest) path is different than navigation. Here, even if we have some path from the initial node to the goal node, we must perform a systematic search on the graph in order to guarantee the optimality of the path. A common method for finding the shortest path in graphs is to use the A* algorithm [5]. A* keeps an *Open-list* of generated nodes and expands them in a best-first order according to a cost function defined by $f(n) = g(n) + h(n)$. In standard A*, $g(n)$ is the distance from the initial node to node n, and $h(n)$ is an *admissible* heuristic estimate of the cost of the path from node n to the goal node. $h(n)$ is *admissible* if it is always a lower bound on actual cost from node n to the goal (e.g. Euclidean distance between the two nodes in a physical environment).

Usually, A* (or its variants, e.g., IDA*) are applied to exponential combinatoric problems. In such domains, an A* expansion cycle is carried out in constant time as it takes a constant amount of time to retrieve a node from the open-list and to generate all its neighbors by applying domain-specific operators to the expanded node. Therefore, the time complexity of A* is usually measured in terms of the number of generated nodes. This is not the case in physical graphs where an agent needs to physically travel to a node in order to explore it. The Physical A* algorithm (PHA*) and its multi-agent version MAPHA* [4] find the optimal path between two points in a partially known real physical environment with either one or many mobile agents. These algorithms modify the A* algorithm to work on graphs with physical characteristics where agents need to physically travel to a node in order to learn about its neighbors. The focus of these algorithms is to minimize the travel effort of the agents. In MAPHA*, a complete sharing of knowledge between agents is assumed. This could be achieved by a supervisor agent coordinating all agents or by a global database which holds the entire knowledge. A reminiscent approach is the D* algorithm family (e.g., D*-lite) [6] where a mobile robot needs to find the shortest path in an environment that was completely known but changes dynamically over time.

The present paper addresses the navigation problem in the partially known physical environments by a distributed group of agents. Agents are autonomous entities, processing information received from other agents and making their own navigational decisions. The task is to reach the goal node as soon as possible. When the first agent reaches the goal the task is achieved and the agents terminate. Agents explore the graph by moving on it, sharing their discovered nodes and edges as well as their current position and running a distributed navigation algorithm (presented below) to arrive at the goal node. Each agent runs the same algorithm but uses its own partial knowledge of the environment. A distributed algorithm is in general more fault tolerant and is expected to reduce communication, compared to a centralized algorithm [8]. A distributed navigation algorithm is naturally scalable to a large number of navigators.

1.2 Communication models

There are many models of communication in multi-agent systems that allow agents to cooperate and coordinate their decision making. The most trivial model is a complete knowledge sharing where any new discovery of an agent is immediately shared with all the other agents. Other models restrict the level of communication.

The present study focuses on a natural and realistic model for the communication among agents where each agent can only communicate with a subset of the agents - its *neighboring agents* (or *neighbors* in short). The *neighbor* relation is symmetric. We assume that agents can communicate with one another only if the distance between them is below a given radius R. We call this the*radius constraint*. To satisfy these two constraints, we enforce neighboring agents to stay close to one another at all times in order to be able to communicate. Neighboring agents exchange relevant information, such as their current location and the length of the outgoing edges. In each step, agents receive messages from other agents, decide whether and where to move, and send updates to their neighboring agents.

We use two performance measures - the path traveled by the first agent arriving at the goal node and the quality of the best path found compared to the optimal path. Our agents are designed to reach the goal as fast as possible and to minimize their travel effort. This corresponds to the first measure. On the other hand, during navigation they also explore many nodes and the shortest path that will be known to them at the end might improve. This corresponds to the second measure. These two measures might be in conflict with each other. This is known as the exploration versus exploitation conflict. In this paper we show that even though we are mostly concerned with the travel effort measure, a good balance between the two measures can be achieved. Indeed, experimental results show that when the algorithm terminates the quality of the best known path is very close to the optimal path.

The DisNAV algorithm was implemented on a distributed asynchronous simulator. Experiments were run on different graphs and different number of agents and neighborhood schemes. Results show that the proposed distributed navigation algorithm generates an efficient spread of knowledge throughout the environment. Agents reach the goal node very fast and the length of the path that they find is close to that of the optimal path. We have considered three different communication schemes and show that using a constrained scheme on the defined neighborhood produces better results than using a more liberal scheme.

2. NEIGHBORHOOD SCHEMES

The main motivation of the present study is to look at multi-agent navigation as a distributed problem, where agents communicate only with their neighboring agents. Given an agent A, its *neighborhood* N is a subset of the agents. A must coordinate its moves with all the agents in N. We have defined three different neighborhood schemes: *static neighborhood*, *unconstrained dynamic neighborhood*, and *constrained dynamic neighborhood*. In all these schemes, we have defined a radius parameter R such that agents can only receive messages sent by an agent distanced away no more than R units. We call this the *radius constraint*. However, the schemes differ in the definition of neighborhood. Extreme values for the radius R significantly influence the entire behavior of the multi agent system. With large R, the radius constraint becomes meaningless and the communication scheme is similar to the full knowledge sharing. With small R, moves are limited and we get close to the behavior of a single agent. Thus, the value for R should be carefully chosen.

2.1 Static neighborhood

In order to simplify the definition of the distributed navigation problem, we have defined the notion of static neighborhood which enable an easy definition of a static connected communication network of agents. In a static neighborhood each agent is assigned a set of neighbors and this set is fixed throughout the entire navigation. An agent has to coordinate its moves with all its neighbors therefore must keep the radius constraint with all them. The rationale behind this restriction is that new discoveries of an agent will always be propagated to its neighbors. Static neighborhoods can cause some restriction on agent moves since the *radius constraint* may prevent agents from making some of their possible moves, if they move too far from their neighbors. Note that in the static neighborhood scheme an agent does

not communicate with an agent which is not its neighbor even if it is located nearby.

2.2 Constrained dynamic neighborhood

In this scheme, an agent needs to coordinate its moves with K agents within the radius R. However, unlike the *static neighborhood* version where the set of neighboring agents is predefined, here, any K agents suffice. The only requirement is to keep the *radius constraint* with any K agents whose identities can change over time. Agents broadcast their move proposal and wait for any K approvals from agents within the radius R.

2.3 Unconstrained dynamic neighborhood

This is the most liberal scheme. Here, when an agent chooses a move, it broadcasts its decision and performs that move without the need to coordinate it with anyone. Of course, only agents within the radius R actually receive its move decision and other messages. The pitfall of this scheme is that an agent can travel by itself to a distant location where it cannot communicate with any other agent. This will weaken the knowledge sharing of the entire multi-agent system as this agent becomes isolated.

3. DISTRIBUTED NAVIGATION

We now turn to describe the DisNAV algorithm, focusing first on the main case of *static neighborhood*. We then present the modifications that are needed for the *constrained dynamic neighborhood* and the *unconstrained dynamic neighborhood* cases.

3.1 DisNAV for static neighborhood

The DisNAV algorithm is run by each agent independently. Agents communicate with their neighbors, exchanging messages which contain information such as their current position, node discoveries and move proposals. Agent navigation is based on its known information which is gathered by its own discoveries as well by information received in messages from other agents. Agents keep their local information in three main data structures: *KnownGraph*, *AgentView* and *Open-list*, all are described below.

KnownGraph - The *KnownGraph* is the part of the world currently known to the agent. *KnownGraph* is a subgraph of the entire environment modeled as a weighted graph. We assume that all agents start from the same initial node. Since agents send each node discovery to their neighbors (as will be described below) the *KnownGraph* will always be a connected graph. Each agent uses its own *KnownGraph* to calculate distances between nodes during the execution of the algorithm. In addition, when the algorithm terminates, *KnownGraph* is used for calculating the best path between the initial node and the goal node.

AgentView - The *AgentView* data structure, contains important information about each of the neighbors of the agent.
AgentView is updated by messages received from the neighbors. In particular, *AgentView* of an agent contains the current position and the current target or the move proposals of each of its neighbors.

Open-list - Each agent maintains a local *Open-list* of nodes which are in the frontier of its *KnownGraph*. Nodes are inserted to the *Open-list* when they are first known to the agent either by its own discovery or by a message from one of its neighbors. Similarly, a node is being deleted from the *Open-list* when the agent (or one of its neighboring agents that has just sent him a message) physically visits it and discovers all its neighbors (which are themselves inserted to the *Open-list*). The *Open-list* is maintained as a priority queue according to an intelligent cost function described below. Agents choose the best node on the *Open-list* and propose to physically travel to it.

We now turn to present the details of the DisNAV algorithm. The pseudo-code for the algorithm is given in Figure 1. The navigation is done as follows. First, an agent picks a *target* node from its *Open-list* and proposes a *move* to that target. Each proposed move should be approved by all neighbors before its actual performance. When the move is approved by the neighbors, the agent physically navigates from its current node to the target node.

The main procedure run by each agent is ***DisNAV*** which is a message driven distributed algorithm. In each *cycle* of DisNAV an agent reads all the received messages (lines 5-6) and decides about a proper resulting operation. The resulting operation can be a message sending (e.g. NODE, APPROVAL etc.) and can also include a performance of an approved move. An agent running the algorithm can be in one of several possible logical states. Different events can change the state of an agent. The initial state of an agent is READY. Each agent runs a loop, which ends when the agent enters the TERMINATED state.

The basic state is the READY state. In this state the agent selects a *target* node from the *Open-list*. The *target* is chosen to be the best node on it its *Open-list* according to the cost function described below. Then, the agent sends an *OK?* message to its neighbors in which it proposes to move to this target and asks for their approval (***proposeMove***, lines 51-54). The state of the agent is now changed to WAIT_FOR_APPROVAL.

Note that we ensure that targets that break the *radius constraint* between an agent and the current location of its neighbors will not be selected by giving them a cost of infinity. Therefore, sometimes agents choose targets not in the front of the *Open-list* and might even decide to remain idle.

Receiving an OK? message (lines 19-23), an agent has two choices. It can either approve the proposal, replying with an APPROVAL message or reject the move using a NOGOOD message. We say that move proposals of two neighboring agents are in *conflict* if performing them both at the same time will break the *radius constraint* between them. Such a conflict is resolved in favor of the move with the lower cost. In such a case, a NOGOOD message is sent only by the agent with the better move. When that proposing agent receives a NOGOOD (lines 29-31), the state returns to READY and the move proposal is canceled.

When the proposing agent receives APPROVALs from all its neighbors (lines 25-28), its state changes to IN_TRANSIT. In this case the target node is removed from the *Open-list*, and the agent starts performing the *move* to the target. A *move* includes a sequence of navigation *steps*. At each *step* the agent chooses the neighbor w that minimizes the sum of the distances from the current node v to w and from w to the target node t. This cost function is called A*DFS in [4] since it uses a cost function which is similar to that of A*, i.e., $f(n) = g(n) + h(n)$. Note, that this cost function is used here locally to find a path from the current node to the target node.

Upon agent arrival to a node (line 37), the procedure ***uponArrivalToNode*** (lines 40-50) is called. If it is the goal node, the agent changes its state to TERMINATED and sends a TERMINATE message. Each agent receiving such a message (lines 32-34), sends it to its neighbors and changes the state to TERMINATED. Otherwise, if the node is not the goal, the agent learns about this node's neighbors and sends a NODE message, to inform its neighboring agents about its current position and about the node's neighbors. If the current node is in the *Open-list* this node is *expanded*, i.e., it is removed from the *Open-list*, and the node's neighbors which where not visited yet, are inserted to the *Open-list*. If the current node is the target node (lines 48-50) the state returns to READY and the agent considers another move proposal.

Agent receiving a NODE message (lines 8-18) updates its *AgentView* with the sender location and updates its *KnownGraph* with the new nodes and edges discovered. Of course, in case the node is in the *Open-list*, it is immediately expanded by the agent without the need for the agent to physically visit that node. This is the main contribution of data sharing among neighboring agents.

3.2 Unconstrained dynamic neighborhood

The unconstrained dynamic neighborhood scheme is a simplified variation of the static neighborhood algorithm. When an agent decides on its target (the best node in its *Open-list*), it broadcasts a MOVE message declaring it. The agent changes its state to IN_TRANSIT and removes the target from the *Open-list*. This step replaces the sending of an OK? move proposal and waiting for APPROVAL and NOGOOD messages as done in the static neighborhood (or the constrained dynamic neighborhood). The OK?, APPROVAL and NOGOOD messages, as well as the state WAIT_FOR_APPROVAL, become irrelevant in this scheme.

3.3 Constrained dynamic neighborhood

In this scheme, each agent has a dynamic set of *current neighbors* all of them are within the radius R. When all agents start at the same point, the set initially includes all the other agents. Otherwise, the set is modified according to agent positions as they are received in NODE messages. During navigation, agents are added and removed from the set according to the moves performed. A *neighbor* in this context is therefore a member of the *current neighbors* set. The changes that the constrained dynamic neighborhood algorithm imposes on the static neighborhood version are the following:

1. Agents propose only moves which are consistent with all their *current neighbors* locations.

2. Agents wait for k APPROVALs. When they are received, the agent broadcasts a MOVE message declaring its target, changes its state to IN_TRANSIT and removes the target from the *Open-list*.

3. If an OK? is received from an agent which is not a *neighbor* but within the radius R, send an APPROVAL and add the agent to the set of *current neighbors*.

4. If an APPROVAL is received from an agent which is not a *neighbor* and if the move proposal is preformed then add the agent to the *current neighbors*.

```
DisNAV:
1.  state ← READY; node ← initial; proposal ← null
2.  expand node
3.  update KnownGraph and Open-list
4.  while(state ≠ TERMINATED)
5.    while(not msgQueue.isEmpty)
6.      msg ← msgQueue.getFirst
7.      switch msg.type
8.        NODE:
9.          update KnownGraph, AgentView
9a.           and Open-list
10.         if(state=READY)
11.           proposeMove
12.         else if(state=WAIT_FOR_APPROVAL)
13.           if(move proposal is redundant)
14.             state ← READY
15.             ignore proposal
16.           else if(move proposal should be updated)
17.             update proposal
18.             send(OK?,neighbors)
19.       OK?:
20.         if(self already sent a better conflicting
20a.            move proposal)
21.           send(NOGOOD,sender)
22.         else
23.           send(APPROVAL,sender)
24.       APPROVAL:
25.         if(state=WAIT_FOR_APPROVAL
25a.           and all approved)
26.           state ← IN_TRANSIT
27.           target ← proposal
28.           remove target from Open-list
29.       NOGOOD:
30.         if(state=WAIT_FOR_APPROVAL)
31.           state ← READY
32.       TERMINATE:
33.         state ← TERMINATED
34.         send(TERMINATE,neighbors)
35.   if(state=IN_TRANSIT)
36.     move to a neighboring node, navigating to target
37.     uponArrivalToNode
38.   else if(state=READY and proposal=null)
39.     proposeMove

uponArrivalToNode:
40. if(node=goal)
41.   state ← TERMINATED
42.   send(TERMINATE,neighbors)
43. else
44.   discover node neighbors
45.   send(NODE,neighbors)
46.   if(node in Open-list or node=target)
47.     update KnownGraph and Open-list
48.     if(node=target)
49.       state ← READY
50.       proposeMove

proposeMove:
51. proposal ← get best node from Open-list
52. if(proposal is worthwhile)
53.   send(OK?,neighbors)
54.   state ← WAIT_FOR_APPROVAL
```

Figure 1: The DisNAV Algorithm

5. If a conflicting MOVE is received from a *neighbor*, remove the agent from the *current neighbors*.

4. COST FUNCTION HEURISTICS

We now turn to describe the cost function for choosing a target node. Each agent running the DisNAV algorithm, owns a local *Open-list* which includes nodes discovered during navigation, as well as nodes discovered by neighboring agents. The nodes are chosen as targets in a best-first order according to a cost function[1].

The ordinary cost function used by A* and other search algorithms for finding optimal paths is $f(n) = g(n) + h(n)$, where $g(n)$ is the distance traveled from the initial node to the node n, and $h(n)$ is a heuristic estimate of the cost of traveling from node n to the goal node. A reminiscent algorithm is Real-Time-A* (RTA*) [7]. RTA* selects the node with the best cost in the *Open-list*. The problem solver then moves one step along the path to that node, and the search continues from the new state of the problem solver. The cost function for an arbitrary node n in RTA* is also $f(n) = g(n) + h(n)$ but in this case $g(n)$ is an estimation of the distance of node n from the current node and not from the initial node.

DisNAV deals with physical graphs where nodes can be expanded only after an agent physically visits them and learns about their locations and outgoing edges. This is completely different from RTA* which assumes that a node can be expanded in the computer's memory without an agent having to physically visit that node. However, the cost function used in DisNAV treats $g(n)$ similarly to RTA* and is measured according to the current location of the agent. Thus, at every step the cost values of the entire set of nodes in the *Open-list* are changed so that g will reflect the current location. The best node in the open list according to the cost function is selected and becomes the target node of that agent.

A good cost function for our purpose needs to combine both navigation and exploration. In other words, we want to get to the goal node as fast as possible but also to learn about new regions of the graph in order to provide useful information to other agents. For this purpose, one can use combinations of the following two functions:

- $f(n) = g(n) + h(n)$ where g is measured from the current node and h is the Euclidean distance from n to the goal. This is the navigation part which leads the agent to the goal as fast as possible.
- $vc(n)$ - a counter that keeps track of the number of visits to the node n done by other agents. $vc(n)$ is initially set to 0 and is incremented every time the agent is informed of a visit in n.

When combining both cost functions then nodes with small f-value are preferred, but on the other hand, so are nodes with small vc-value. Such nodes are on a promising path to the goal and are distanced away from paths chosen by other agents. A number of combinations between f and vc were tried. The best results were obtained by using

$$c(n) = f(n) \cdot [1 + p \cdot vc(n)]$$

[1]When an agent knows that one of its neighbors is navigating to a target node, this node can be removed from the *Open-list*, so the agent will not choose the same target for choosing a target node from the open list. The constant p defines the weight of the exploration in the cost function. Our extensive empirical studies have shown that $p \approx 0.3$ produces the best performance.

5. CORRECTNESS

DisNAV terminates when the first agent reaches the goal node. An important role of the algorithm is to delete nodes from the open list and to add new nodes to it. Also, it is easy to see that a node cannot be inserted to the *Open-list* more than once. In other words, the frontier of the *KnownGraph* is always expanding. Therefore, as long as we keep doing this, more and more new nodes will be discovered and since the graph is finite we will eventually reach the goal node. There are, however, some "pathological" cases where the entire multi-agent system enters a *deadlock* and cannot continue because there is not even one agent with a legal target node in its *Open-list*. This happens when for each agent all the nodes in the *Open-lists* break the radius constraint with at least one of its neighboring agents. Of course, deadlocks are not possible in the unconstrained neighborhood scheme since the radius constraint is not kept.

5.1 Deadlock elimination

We want to eliminate such deadlocks and ensure the termination of the algorithm even when keeping the *radius constraint*. We provide a solution to the static neighborhood scheme. A solution to the dynamic constrained neighborhood scheme can be easily modified since the basic principle is similar. The basic idea for deadlock elimination is to choose a *leader* agent. When a deadlock occurs, the leader chooses a move from its open-list and perform it. The other agents are forced to move in such a way that the radius constraint is kept given that leader performs its chosen move. We choose the leader that will benefit the most by its chosen move. The deadlock elimination steps are the following:

1. **Initialization -** An agent with no legal move initializes the deadlock elimination process by sending an *INVITE* message to all its neighbors declaring its wish to start a deadlock elimination process.
2. **Joining -** Agents receiving the *INVITE* can *join* this process and invite their neighbors as well or continue their navigation as usual, as long as they have legal moves in their *Open-list*.
3. **Leader Election -** All the agents which joined the deadlock elimination process pick a leader. The agent with the move of the best cost (of all the *Open-lists*) is elected and allowed to perform its move. This is done using a leader election algorithm [1].
4. **Distributed Search -** The agents solve the distributed constraint satisfaction problem (DisCSP) [10, 12] of finding a legal move for each of them. They run a DisCSP search algorithm (e.g. ABT [11, 12]) where moves of an agent are sorted according to their cost.
5. **Moves are performed -** The agents perform their moves found by the distributed search.

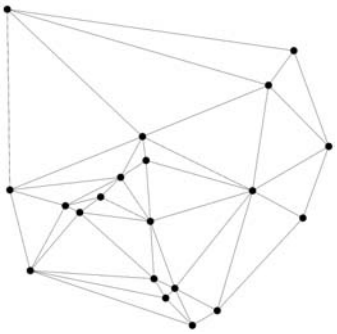


Figure 2: Delaunay graph of 20 nodes.

6. EXPERIMENTAL EVALUATION

DisNAV was evaluated experimentally on random Delaunay graphs [9] which are common testbeds (e.g., [4]) for simulating physical graphs. They are derived from Delaunay triangulations computed over a set of planar point, generated by a *Poisson point process* [9]. Points are distributed at random over a square, using a uniform probability density function. In regular Delaunay graphs, each node is connected by edges to its nearest neighbors as illustrated in figure 2. This property may not always apply to a real road map. For example, nearby geographic locations may not always be connected by a road segment, due to the existence of obstacles like a mountain or a river. To capture this additional characteristic, sparse Delaunay graphs [4] were considered. Instances of these variants can be easily obtained from regular Delaunay graphs by random deletion of edges. We define the *density* as the fraction of Delaunay edges left in the graph. The graphs were generated to be connected, by first adding a spanning tree edges and then adding random Delaunay edges up to the desired *density*.

In all our experiments we used Delaunay graphs which were randomly generated on a square with 100x100 units. For each experiment we generated 100 different graphs and selected a pair of nodes with maximal distance as the initial and goal nodes. Every data point in the figures below is the average over these 100 cases.

6.1 Comparing the different neighborhoods

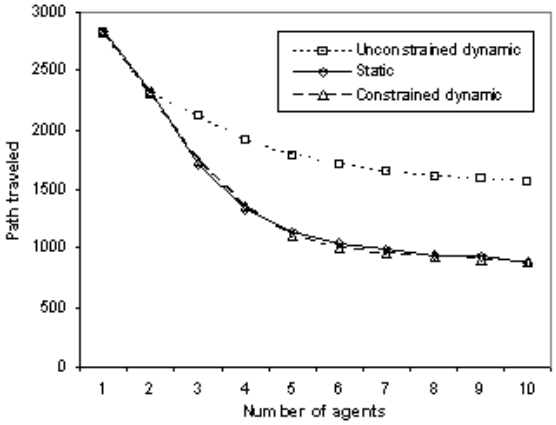


Figure 3: Different neighborhood schemes

Figure 3 presents the length of the path traveled by the first agent reaching the goal node, as a function of the number of participating agents on a Delaunay graph with 500 nodes. The radius R was set to 25 units, which is much smaller than the diameter of the problem and information is only shared locally. There are three curves in the figure, one for each neighborhood schemes. The neighborhood topology for the static neighborhood scheme was a chain where each agent was defined to be a neighbor of its predecessor and successor in the chain. Thus, each agent (except for the agents at the two ends of the chain) has two neighbors, but no 3-agent cliques exist. Similarly, the size of the dynamic constrained neighborhood was set to 2.

For all neighborhood schemes, as more agents are added the first agent reaches the goal node after traveling a shorter path. However, the figure clearly shows that the two constrained neighborhood schemes significantly outperform the unconstrained neighborhood scheme and that the path to the goal is found much faster with the constrained schemes. The two constrained schemes produced results which are not significantly different. We have performed an extensive number of experiments with all three neighborhood schemes and obtained results with the same tendency.

The unconstrained scheme performs rather poorly. If an agent is not constrained to actually stay close to other agents it might work on its own task and run away from the others. With this behavior, a valuable knowledge sharing is not guaranteed and the performance of the multi-agent system derogates. This teaches us the following valuable lesson in multi-agent communication - in order to have an efficient multi-agent system we should build a strong and constrained communication scheme to guarantee the knowledge sharing and data spread.

The two constrained schemes (static and dynamic) behave similarly. However, there are a number of advantages in the static neighborhood scheme. First, this scheme is simpler as an agent always has the same set of neighbors and thus we have a fixed and connected communication graph. This allows a simpler and faster termination of the entire task. Furthermore, the number of messages sent in this scheme is much smaller than the dynamic scheme. In the static scheme an agent only sends messages to the specific set of neighbors while in the dynamic neighborhood scheme the agents actually broadcast their discoveries and proposals to the entire set of agents. Therefore, for the remaining part of the experiments we focus on the static neighborhood results.

6.2 Static neighborhood

The next set of experiments compares different densities on sparse Delaunay graphs with 1000 nodes. The radius R was set to 75 units. Note that since agents communicate only with a predefined (static) set of agents, the relatively large radius (75) does not cause global information sharing. The number of agents in the experiments was varied from one to ten and the neighborhood topology was again a chain where each agent has two neighbors.

Figure 4 shows the length of the path traveled by the first agent reaching the goal node, as a function of the number of participating agents. The three curves correspond to graph densities of 0.3, 0.35 and 0.4. Again, adding more agents improves the results. It is also clear that as the density of the graph grows, the path traveled by the first agent is shorter. In graphs with high densities, agents have more alternative

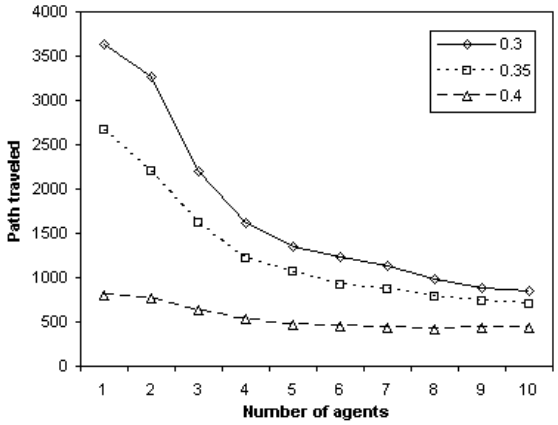


Figure 4: Path traveled (1000 nodes)

paths to the goal and there are only a few dead-ends. Such graphs are easy for navigating and agents can find their way by simple directional navigation. They do not need a lot of information from their neighbors. Thus, the improvement for adding more agents is small and a single agent is almost as good as many agents. When the graph is sparse and there are many dead-ends, the distributed algorithm outperforms a single agent. In other words, the first agent from the group reaches the goal much earlier than a single agent.

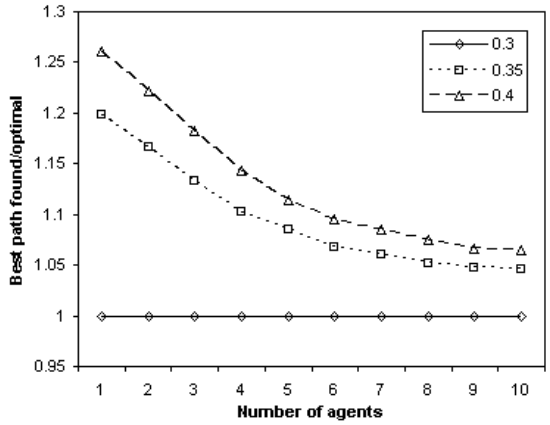


Figure 5: Best path found (1000 nodes)

Figure 5 presents the ratio between the length of the best path known when the first agent reaching the goal node and the optimal path. It is clear from figure 5 that as the number of participating agents grows, the first agent to reach the goal node knows a better path, compared to the optimal. In graphs with low densities (0.3 and below) the best known path is actually the optimal path. This is an interesting result. In such graphs agents cover a much larger portion of the graph due to dead-ends, so on arriving to the goal node, their *KnownGraph* covers a larger portion of the complete graph. In other words, the higher cost needed to reach the goal node (figure 4) reveals a path that is closer to the optimal. The combined result of figures 4 and 5 can be stated as follows. Using a minimal number of neighbors and using a large number of agents, the harder problems (low density graphs) are solved close to optimality.

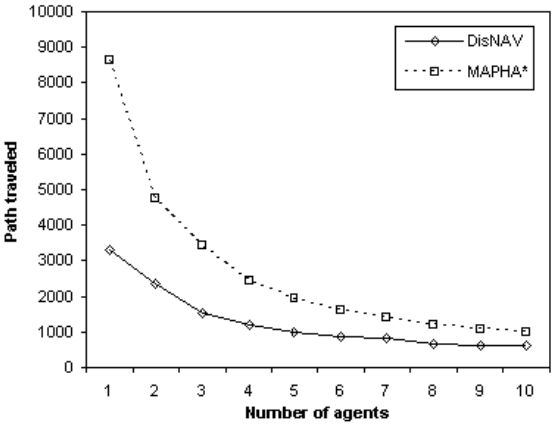


Figure 6: DisNAV vs. MAPHA* (500 nodes)

Figure 6 presents a comparison between DisNAV and MAPHA*, the multi-agent physical A* algorithm of [4], on a graph with 500 nodes and a density of 0.3. Note again, that MAPHA* actually activates a systematic global search and is guaranteed to return an optimal solution. The two curves correspond to the length of the path traveled by the first agent running DisNAV that reached the goal node, and the path traveled by the agents performing MAPHA*. As shown in figure 6, the first DisNAV agent that reaches the goal node, travels a shorter path than the MAPHA* agent. Recall from figure 5 that for a density of 0.3 the best path found by DisNAV is in practice the optimal path. This means that in graphs with low densities it is more beneficial to use DisNAV agents than MAPHA* agents as they both find the optimal path in practice.

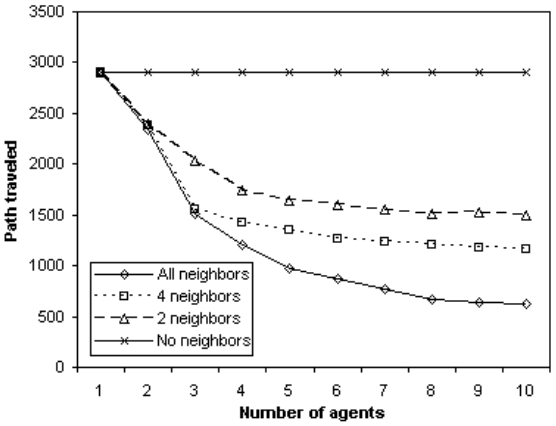


Figure 7: Different number of neighbors (500 nodes)

In another experiment the number of neighbors connected to each agent was varied. Figure 7 shows the length of the path traveled by the first agent reaching the goal node, as a function of the number of agents. The four curves correspond to cases where there are no neighbors, when the number of neighboring agents is 2 and 4 and when all agents

are connected to each other. The results show that adding neighbors improves the total knowledge sharing and reduce the travel effort. We see a diminishing return when adding more neighbors to each agent and the biggest jump is between the case of no neighbors and the case where each agent has two neighboring agents.

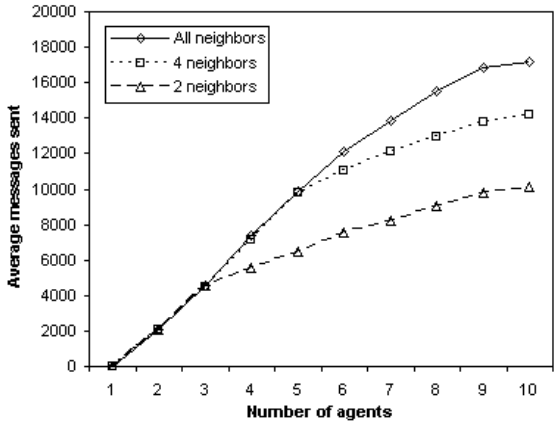


Figure 8: Communication load (500 nodes)

Figure 8 presents the average overall number of messages sent by the agents. The three curves correspond to different number of neighboring agents. It is clear from the results that as the number of neighbors increases, the average number of messages sent by an agent increases as well.

7. DISCUSSION

A multi-agent distributed algorithm for navigation in an unknown physical environment was presented. Our approach assumes that agents have limited communication abilities, and each of them is able to communicate only with a predefined set of agents. In our algorithm each agent is only navigating to the goal and we do not activate a systematic global search to find the optimal path. Former studies (MAPHA* [4]) activate a systematic global search and use complete knowledge sharing between the agents. Therefore, the travel effort and communication overhead of our agents are much smaller than MAPHA*. Even with the above limits, the experiments have demonstrated that our agents, using a simple distributed knowledge sharing, work together very efficiently. The path found by them in practice, is very close (and in many cases equal) to the optimal path. Thus, in practice, unless the optimal path is a hard requirement, our distributed algorithm is preferable. Furthermore, in many cases an agent that arrives at the goal node can prove that the best path known to it is actually the optimal path. This can be done by activating A* from the initial node to the goal. If all nodes generated by A* are included in the *KnownGraph* then the agent can prove that it has the optimal path. Our experimental results show that in small, sparse graphs (e.g., 200 nodes and a density of 0.3) the agents always found the optimal path and in 40% of the cases they could even prove that it is optimal.

Our experiments show that as the number of participating agents grows, the path traveled by the first agent arriving at the goal node is shorter and the best known path is also shorter. This benefit is more pronounced as the graph density decreases. In dense graphs even single agent would travel a relatively short path and will find a path which is a good approximation of the optimal path and adding more agents does not improve the results. However, in sparse graphs, the best path known to a single agent is far from the optimal path and adding more agents improves the results dramatically.

In addition, our experiments with the different neighborhood schemes teach us an important lesson about the influence of the communication constraints on the behavior of the entire multi-agent systems. Indeed, such constraints add strength to the knowledge sharing of the system and the entire multi-agent system performs much better.

Future work will examine how do the communication network topology, the neighborhood radius and the connectivity affect the overall performance. Another interesting question is to examine the cases where agents are starting from different initial nodes and when all agents are required to arrive at the goal node in order to complete the task.

8. REFERENCES

[1] H. Attiya and J. Welch. *Distributed Computing*. McGraw-Hill, 1998.

[2] M. A. Bender, A. Fernandez, D. Ron, A. Sahai, and S. P. Vadhan. The power of a pebble: Exploring and mapping directed graphs. In *Proc. ACM Symposium on the Theory of Computing*, pages 269–278, Dallas, Texas, May 1998.

[3] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 2005.

[4] A. Felner, R. Stern, A. Ben-Yair, S. Kraus, and N. Netanyahu. PHA*: Finding the shortest path with a* in unknown physical environments. *JAIR*, 21:631–679, 2004.

[5] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, SCC-4(2):100–107, 1968.

[6] S. Koenig and M. Likhachev. D* lite. In *Proc. AAAI-02*, pages 476–483, Edmonton, Canada, July–August 2002.

[7] R. E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(3):189–211, 1990.

[8] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Series, 1997.

[9] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations, Concepts, and Applications of Voronoi Diagrams*. Wiley, Chichester, UK, 1992.

[10] G. Solotorevsky, E. Gudes, and A. Meisels. Modeling and solving distributed constraint satisfaction problems (dcsps). In *Constraint Processing-96*, October 1996.

[11] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Data and Knowledge Engineering*, 10(5):673–685, September 1998.

[12] M. Yokoo and K. Hirayama. Algorithms for distributed constraint satisfaction: A review. 3:189–212, 2000.

Accident or Intention: That Is the Question (in the Noisy Iterated Prisoner's Dilemma)

Tsz-Chiu Au
Department of Computer Science
University of Maryland
College Park, MD 20742
chiu@cs.umd.edu

Dana Nau
Department of Computer Science
University of Maryland
College Park, MD 20742
nau@cs.umd.edu

ABSTRACT

This paper focuses on the Noisy Iterated Prisoner's Dilemma, a version of the Iterated Prisoner's Dilemma (IPD) in which there is a nonzero probability that a "cooperate" action will accidentally be changed into a "defect" action and vice versa. Tit-For-Tat and other strategies that do quite well in the ordinary (non-noisy) IPD can do quite badly in the Noisy IPD.

This paper presents a technique called *symbolic noise detection*, for detecting whether anomalies in player's behavior are deliberate or accidental. The key idea is to construct a model of the other agent's behavior, and watch for any deviation from this model. If the other agent's next action is inconsistent with this model, the inconsistency can be due either to noise or to a genuine change in their behavior; and we can often distinguish between two cases by waiting to see whether this inconsistency persists in next few moves.

We entered several different versions of our strategy in the 2005 Iterated Prisoner's Dilemma competition, in Category 2 (noisy environments). Out of the 165 contestants in this category, our programs consistently ranked among top ten. The best of our programs ranked third, and it was beaten only by two "master-slave strategy" programs that each had a large number of "slave" programs feeding points to them.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*; I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Gaming*

General Terms

Design, Experimentation, Measurement, Theory

Keywords

Adaptation; Cooperation; Coordination; Game theory; Learning; Multi-agent systems; Noise; Opponent modeling; Prisoner's Dilemma; Repeated game; Tit-for-tat

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

The Iterated Prisoner's Dilemma (IPD) has become well known as an abstract model of a class of multi-agent environments in which agents accumulate payoffs that depend on how successful they are in their repeated interactions with other agents. An important variant of the IPD is the *Noisy* IPD, in which there is a small probability, called the *noise level*, that accidents will occur. In other words, the noise level is the probability of executing "cooperate" when "defect" was the intended move, or vice versa.

Accidents can cause difficulty in cooperations with others in real-life situations, and the same is true in the Noisy IPD. Strategies that do quite well in the ordinary (non-noisy) IPD may do quite badly in the Noisy IPD [4, 6, 7, 18, 19, 20]. For example, if two players both use the well-known Tit-For-Tat (TFT) strategy, then an accidental defection may cause a long series of defections by both players as each of them punishes the other for defecting.

In this paper, we describe a technique called *symbolic noise detection* for dealing with noise. The key idea is to build a symbolic model of how the other agent behaves, and watch for any deviation from this model. If the other agent's next move is inconsistent with their past behavior, this inconsistency can be due either to noise or to a genuine change in their behavior; and we can often distinguish between these two cases by waiting to see if this inconsistency persists or not in the next few iterations of the game.

To test our technique, we wrote several programs that used it, and entered them as contestants in Category 2 (the Noisy IPD) of the 2005 Iterated Prisoner's Dilemma competition [14]. In the competition, the noise level was 10%. The result of the competition showed an impressive performance for symbolic noise detection (Table 1). Seven of our nine symbolic noise detection programs placed among top ten according to their overall average score. Our best program, DBSz, placed third, and it was the best-performing non-master-slave program in Category 2.[1]

[1]Each participant in the competition was allowed to submit up to 20 programs as contestants. Some participants took advantage of this to submit collections of programs that worked together in a conspiracy in which 19 of their 20 programs (the "slaves") worked to give as many points as possible to the 20th program (the "master"). In contrast, DBS is not a master-slave program. Instead, DBS builds behavioral models of the other contestants in order to coop-

This paper describes the design of DBS, particularly the implementation of the symbolic noise detection in DBS. Section 2 presents the motivation for the approach, Section 3 gives the basic definitions, and Section 4 gives an overview of how DBS works. Section 5 how the opponent modeling is done and in particular the symbolic noise detection works. Section 6 discusses DBS's performance in the 2005 Iterated Prisoner's Dilemma competition, and contrasts DBS to the notorious master-and-slaves strategy. Section 7 discusses related work, and Section 8 is the conclusion.

2. MOTIVATION AND APPROACH

The techniques used in DBS are motivated by a British army officer's story that was quoted in [3, page 40]:

> I was having tea with A Company when we heard a lot of shouting and went out to investigate. We found our men and the Germans standing on their respective parapets. Suddenly a salvo arrived but did no damage. Naturally both sides got down and our men started swearing at the Germans, when all at once a brave German got onto his parapet and shouted out: "We are very sorry about that; we hope no one was hurt. It is not our fault. It is that damned Prussian artillery." (Rutter 1934, 29)

Such an apology was an effective way of resolving the conflict and preventing a retaliation because it told the British that the salvo was not the intention of the German infantry, but instead was an unfortunate accident that the German infantry did not expect nor desire. The reason why the apology was convincing was because it was consistent with the German infantry's past behavior. The British had was ample evidence to believe that the German infantry wanted to keep the peace just as much as the British infantry did.

More generally, an important question for conflict prevention in noisy environments is *whether a misconduct is intentional or accidental.* A deviation from the usual course of action in a noisy environment can be explained in either way. If we form the wrong belief about which explanation is correct, our response may potentially destroy our long-term relationship with the other player. If we ground our belief on evidence accumulated before and after the incident, we should be in a better position to identify the true cause and prescribe an appropriate solution. To accomplish this, DBS uses the following key techniques:

1. **Learning about the other player's strategy.** DBS uses an induction technique to identify a set of rules that model the other player's recent behavior. The rules give the probability that the player will cooperate under different situations. As DBS learns these probabilities during the game, it identifies a set of *deterministic* rules that have either 0 or 1 as the probability of cooperation.

2. **Detecting noise**. DBS uses the above rules to detect anomalies that may be due either to noise or a genuine change in the other player's behavior. If a move is different from what the deterministic rules predict, this inconsistency triggers an *evidence collection process* that will monitor the persistence of the inconsistency in the next few iterations of the game. The purpose of the evidence-collection process is to determine whether the violation is likely to be due to noise or to a change in the other player's policy.

3. **Temporarily tolerating possible misbehaviors by the other player.** Until the evidence-collection process finishes, DBS assumes that the other player's behavior is still as described by the deterministic rules. Once the evidence collection process has finished, DBS decides whether to believe the other player's behavior has changed, and updates the deterministic rules accordingly.

Since DBS emphasizes the use of deterministic behaviors to distinguish noise from the change of the other player's behavior, it works well when the other player uses a pure (i.e., deterministic) strategy or a strategy that makes decisions deterministically most of the time. Fortunately, deterministic behaviors are abundant in the Iterated Prisoner's Dilemma. Many well-known strategies, such as TFT and GRIM, are pure strategies. Some strategies such as Pavlov or Win-Stay, Lose-Shift strategy (WSLS) [15, 16, 17, 21] are not pure strategies, but a large part of their behavior is still deterministic. The reason for the prevalence of determinism is discussed by Axelrod in [2]: clarity of behavior is an important ingredient of long-term cooperation. A strategy such as TFT benefits from its clarity of behavior, because it allows other players to make credible predictions of TFT's responses to their actions. We believe the success of our strategy in the competition is because this clarity of behavior also helps us to fend off noise.

The results of the competition show that the techniques used in DBS are indeed an effective way to fend off noise and maintain cooperation in noisy environments. When DBS defers judgment about whether the other player's behavior has changed, the potential cost is that DBS may not be able to respond to a genuine change of the other player's behavior as quickly as possible, thus losing a few points by not retaliating immediately. But this delay is only temporary, and after it DBS will adapt to the new behavior. What is more important is that the techniques used in DBS greatly reduce the probability that noise will cause it to end a cooperation and fall into a mutual-defect situation. Our experience has been that it is hard to re-establish cooperation from a mutual-defection situation, so it is better avoid getting into mutual defection situations in the first place. When compared with the potential cost of ending an cooperation, the cost of temporarily tolerating some defections is worthwhile.

Temporary tolerance also benefits us in another way. In the noisy Iterated Prisoner's Dilemma, there are two types of noise: one that affects the other player's move, and the other affects our move. While our method effectively handles the first type of noise, it is the other player's job to deal with the second type of noise. Some players such as TFT are easily provoked by the second type of noise and retaliate immediately. Fortunately, if the retaliation is not a permanent one, our method will treat the retaliation in the same way as the first type of noise, thus minimizing its effect.

3. DEFINITIONS

3.1 Iterated Prisoner's Dilemma with Noise

In the Iterated Prisoner's Dilemma, two players play a finite sequence of classical prisoner's dilemma games, whose

erate with them as effectively as possible.

payoff matrix is:

		Player 2	
		Cooperate	Defect
Player 1	Cooperate	(u_{CC}, u_{CC})	(u_{CD}, u_{DC})
	Defect	(u_{DC}, u_{CD})	(u_{DD}, u_{DD})

where $u_{DC} > u_{CC} > u_{DD} > u_{CD}$ and $2u_{CC} > u_{DC} + u_{CD}$. In the competition, u_{DC}, u_{CC}, u_{DD} and u_{CD} are 5, 3, 1 and 0, respectively.

At the beginning of the game, each player knows nothing about the other player and does not know how many iterations it will play. In each iteration, each player chooses either to cooperate (C) or defect (D), and their payoffs in that iteration are as shown in the payoff matrix. We call this decision a *move*, hence $\{C, D\}$ is the set of possible moves. After both players choose a move, they will each be informed of the other player's move before the next iteration begins.

If $a_k, b_k \in \{C, D\}$ are the moves of Player 1 and Player 2 in iteration k, then we say that (a_k, b_k) the *outcome* of iteration k. If there are N iterations in a game, then the total scores for Player 1 and Player 2 are $\sum_{1 \le k \le N} u_{a_k b_k}$ and $\sum_{1 \le k \le N} u_{b_k a_k}$, respectively.

The *Noisy Iterated Prisoner's Dilemma* is a variant of the Iterated Prisoner's Dilemma in which there is a small probability called the *noise level*[2] that a player's moves will be mis-implemented. The noise level is the probability of executing C when D was the intended move, or vice versa. The incorrect move is recorded as the player's move, and determines the outcome of the iteration.[3] Furthermore, neither player has any way of knowing whether the other player's move was executed correctly or incorrectly.[4]

3.2 Strategies and Policies

A *history* H of length k is the sequence of outcomes of all iterations up to and including iteration k. We write $H = \langle (a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k) \rangle$. We let $\mathcal{H} = \{(C, C), (C, D), (D, C), (D, D)\}^*$ be the set of all possible histories. A *strategy* $M : \mathcal{H} \to [0, 1]$ associates with each history H a number $M(H) \in [0, 1]$ called the *degree of cooperation*. $M(H)$ is the probability that M chooses to cooperate at iteration $k + 1$, where k is H's length.

As an example, suppose the Player 2 is TFT. We can model this as a function M_{TFT} such that $M_{TFT}(H) = 1$ if H's length is zero or $a_k = C$, and $M_{TFT}(H) = 0$ otherwise.

A *condition* $Cond : \mathcal{H} \to \{\mathsf{True}, \mathsf{False}\}$ is a mapping from histories to boolean values. A history H *satisfies* a condition $Cond$ if $Cond(H) = \mathsf{True}$. A *policy schema* is a set of conditions such that each history in $\mathcal{H}$ satisfies exactly one of the conditions. A *rule* is a pair $(Cond, p)$ which we will write as $Cond \to p$, where $Cond$ is a condition and p is a degree of cooperation. A *policy* is a set of rules whose conditions constitute a policy schema. We will use a policy to approximate the behavior exhibited by a strategy.

[2]The noise level in the competition was 0.1.

[3]Hence, a mis-implementation is different from a misperception, which would not change the outcome of the iteration. The competition included mis-implementations but no misperceptions.

[4]As far as we know, the definitions of "mis-implementation" used in the existing literature are ambiguous about whether either of the players should know that an action has been mis-executed.

```
Procedure DerivedBeliefStrategy()
1.  R_d ← R_TFT      // the default rule set
2.  R_c ← ∅          // the current rule set
3.  a_0 ← C ; b_0 ← C ; H ← ⟨(a_0, b_0)⟩; π = R_d
4.  a_1 ← MoveGen(π, H) ; k ← 1 ; v ← 0
5.  Loop until the end of the game
6.     Output a_k and obtain the other player's move b_k
7.     r+ ← ((a_{k-1}, b_{k-1}) → b_k)
8.     r− ← ((a_{k-1}, b_{k-1}) → ({C, D} \ {b_k}))
9.     If r+, r− ∉ R_c
10.       If ShouldPromote(r+) = true, insert r+ into R_c.
11.    If r+ ∈ R_c, set the violation count of r+ to zero
12.    If r− ∈ R_c and ShouldDemote(r−) = true
13.       R_d ← R_c ∪ R_d; R_c ← ∅; v ← 0
14.    If r− ∈ R_d, then v ← v + 1
15.    If v > RejectThreshold, or (r+ ∈ R_c and r− ∈ R_d)
16.       R_d ← ∅; v ← 0
17.    R_p ← {(C → p') ∈ ψ_{k+1} : C not appear in R_c or R_d}
18.    π ← R_c ∪ R_d ∪ R_p   // R_p is the probabilistic rule set
19.    H ← ⟨H, (a_k, b_k)⟩; a_{k+1} ← MoveGen(π, H) ; k ← k + 1
```

Figure 1: An outline of the DBS strategy. ShouldPromote first increases r^+'s promotion count, and then if r^+'s promotion count exceeds the promotion threshold, ShouldPromote returns true and resets r^+'s promotion count. Likewise, ShouldDemote first increases r^-'s violation count, and then if r^-'s violation count exceeds the violation threshold, ShouldPromote returns true and resets r^-'s violation count. ψ_{k+1} in Line 17 is calculated from Equation 1.

We can model M_{TFT} as a policy as follows: we write $Cond = (a, b)$ as a condition about the outcomes of the last iteration of a history, such that $Cond(H) = \mathsf{True}$ if and only if $a_k = a$ and $b_k = b$. For simplicity, if the length of H is zero, then $Cond(H) = \mathsf{True}$ if and only if $a = C$ and $b = C$. The policy for M_{TFT} is $R_{TFT} = \{(C, C) \to 1.0, (C, D) \to 1.0, (D, C) \to 0.0, (D, D) \to 0.0\}$.

4. DERIVED BELIEF STRATEGY

Figure 1 outlines the DBS strategy. DBS maintains a *hypothesized policy* to model the other player's strategy. DBS initially assumes that the other player's policy is TFT, but at each iteration of the game, DBS updates the hypothesized policy according to how the player has actually played. Section 5 describes how the updating works.

DBS uses the current hypothesized policy to decide what move to make. For this, it uses the MoveGen procedure on line 4 of the figure. MoveGen is bascially a game-tree search procedure for maximizing DBS's utility, assuming that the hypothesized policy π does not change. We omit the details due to lack of space; for details see [1].

An important question is how large a policy schema to use for the hypothesized policy. If the other player's strategy is complicated and the policy schema is too small, the policy schema won't provide enough detail to give useful predictions of the other player's behavior. But if the policy schema is too large, DBS will be unable to compute an accurate approximation of each rule's degree of cooperation, because the number of iterations in the game will be too small. In the competition, we used a policy schema of size 4: $\{(C, C), (C, D), (D, C), (D, D)\}$. We have found this to be good enough for modeling a large number of strategies.

5. LEARNING HYPOTHESIZED POLICIES IN NOISY ENVIRONMENTS

We now describe how DBS learns another player's strategy. Section 5.1 describes how DBS maintains a *discounted frequency* for each behavior: instead of keeping an ordinary frequency count, DBS applies discount factors based on how recent each occurrence of the behavior was. Sections 5.2–5.5 describe why discounted frequencies alone are not sufficient to deal with noise, and how DBS uses symbolic noise detection and temporary tolerance to help overcome the difficulty.

5.1 Learning by Discounted Frequencies

Given a history $H = \{(a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)\}$, a real number α between 0 and 1 (called the *discount factor*), and an initial hypothesized policy $\pi_0 = \{Cond_1 \rightarrow p_1^0, Cond_2 \rightarrow p_2^0, \ldots, Cond_n \rightarrow p_n^0\}$ whose policy schema is $\mathcal{C} = \{Cond_1, Cond_2, \ldots, Cond_n\}$, the *probabilistic policy* at iteration $k+1$ is $\psi_{k+1} = \{Cond_1 \rightarrow p_1^{k+1}, Cond_2 \rightarrow p_2^{k+1}, Cond_n \rightarrow p_n^{k+1}\}$, where p_i^{k+1} is computed by the following equation:

$$p_i^{k+1} = \frac{\sum_{0 \le j \le k} (\alpha^{k-j} g_j)}{\sum_{0 \le j \le k} (\alpha^{k-j} f_j)} \quad (1)$$

and where

$$g_j = \begin{cases} p_i^0 & \text{if } j = 0, \\ 1 & \text{if } 1 \le j \le k,\ Cond_i(H_{j-1}) = \mathsf{True} \text{ and } b_j = C, \\ 0 & \text{otherwise;} \end{cases}$$

$$f_j = \begin{cases} p_i^0 & \text{if } j = 0, \\ 1 & \text{if } 1 \le j \le k,\ Cond_i(H_{j-1}) = \mathsf{True}, \\ 0 & \text{otherwise;} \end{cases}$$

$$H_{j-1} = \begin{cases} \emptyset & \text{if } j = 1, \\ \{(a_1, b_1), (a_2, b_2), \ldots, (a_{j-1}, b_{j-1})\} & \text{otherwise.} \end{cases}$$

In short, the current history H has $k+1$ possible prefixes, and f_j is bascially a boolean function indicating whether the prefix of H up to the $j-1$'th iteration satisfies $Cond_i$. g_j is a restricted version of f_j.

When $\alpha = 1$, p_i is approximately equal to the frequency of the occurrence of $Cond_i \rightarrow p_i$. When α is less than 1, p_i becomes a weighted sum of the frequencies that gives more weight to recent events than earlier ones. For our purposes, it is important to use $\alpha < 1$, because it may happen that the other player changes its behavior suddenly, and therefore we should forget about its past behavior and adapt to its new behavior (for instance, when GRIM is triggered). In the competition, we used $\alpha = 0.75$.

It is important to have a good initial hypothesized strategy because at the beginning of the game the history is not long enough for us to derive any meaningful information about the other player's strategy. In the competition, the initial hypothesized policy is TFT: $\{(C, C) \rightarrow 1.0, (C, D) \rightarrow 1.0, (D, C) \rightarrow 0.0, (D, D) \rightarrow 0.0\}$.

5.2 The Deficiency of using Discounted Frequency to Deal with Noise

It may appear that the probabilistic policy learnt by the discounted-frequency learning technique should be inherently capable of tolerating noise, because it takes many, if not all, moves in the history into account: if the number of terms in the calculation of the average or weighted average is large enough, the effect of noise should be small. However, there is a problem with this reasoning: it neglects the effect of multiple occurences of noise within a small time interval.

A mis-implementation that alters the move of one player would distort an established pattern of behavior observed by the other player. The general effect of such distortion to the Equation 1 is hard to tell—it varies with the value of the paramters and the history. But if several distortions occur within a small time interval, the distortion may be big enough to alter the probabilistic policy and hence change our decision about what move to make. This change of decision may potentially destroy an established pattern of mutual cooperation between the players.

At first glance, it might seem rare for several noise events to occur at nearly the same time. But if the game is long enough, the probability of it happening can be quite high. The probability of getting two noise events in two consecutive iterations out of a sequence of i iterations can be computed recursively as $X_i = p(p + qX_{i-2}) + qX_{i-1}$, providing that $X_0 = X_1 = 0$, where p is the probability of a noise event and $q = 1 - p$. In the competition, the noise level was $p = 0.1$ and $i = 200$, which gives $X_{200} = 0.84$. Similarly, the probabilities of getting three and four noises in consecutive iterations are 0.16 and 0.018, respectively.

In the 2005 competition, there were 165 players, and each player played each of the other players five times. This means every player played 825 games. On average, there were 693 games having two noises in two consecutive iterations, 132 games having three noises in three consecutive iterations, and 15 games having four noises in four consecutive iterations. Clearly, we did not want to ignore situations in which several noises occur nearly at the same time.

The *symbolic noise detection* and temporary tolerance outlined in Section 2 provide a way to reduce the amount of susceptibility to multiple occurrences of noise in a small time interval. The basic idea is to partition the set of rules in the current hypothesized policy into two sets: the set of *deterministic rules* (whose degrees of cooperation are either zero or one) and the set of *probabilistic rules* (whose degrees of cooperation are between zero and one). The deterministic rules are then used to detect noise in the way that we mentioned in Section 2.

In the following subsections, we first present the induction technique we used to identify the deterministic rules in the other player's behavior. Then we describe the evidence-collection process in the symbolic noise detection. After that, we discuss how to cope with the ignorance of the other player's new behavior when the evidence-collection process decides that there is a change in the other player's behavior.

5.3 Identifying Deterministic Rules Using Induction

As we discussed in Section 2, deterministic behaviors are abundant in the Iterated Prisoner's Dilemma. Deterministic behaviors can be modeled by deterministic rules, whereas random behavior would require probabilistic rules.

Deterministic rules have only two possible degrees of cooperation: zero or one. Therefore, there should be ways to learn deterministic rules that are much faster than the discounted frequency method described earlier. For example, if we knew at the outset which rules were deterministic, it would take only one occurrence to learn each of them: each time the condition of a deterministic rule was satisfied, we could assign a degree of cooperation of 1 or 0 depending on

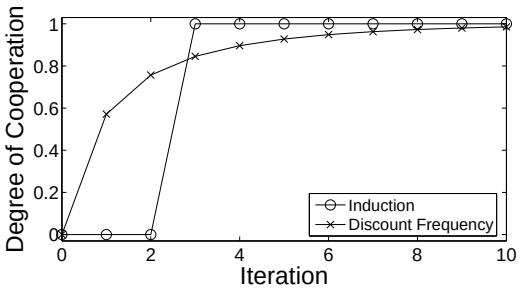


Figure 2: Learning speeds of the induction method and the discounted frequency method when the other player always cooperates. The initial degree of cooperation is zero, the discounted rate is 0.75, and the promotion threshold is 3.

whether the player's move was C or D.

The trick, of course, is to determine which rules are deterministic. We have developed an inductive-reasoning method to distinguish deterministic rules from probabilistic rules during learning and to learn the correct degree of cooperation for the deterministic rules.

In general, induction is the process of deriving general principles from particular facts or instances. To learn deterministic rules, the idea of induction can be used as follows. If a certain kind of behavior occurs repeatedly several times, and during this period of time there is no other behavior that contradicts to this kind of behavior, then we will hypothesize that the chance of the same kind of behavior occurring in the next few iterations is pretty high, regardless of how the other player behaved in the remote past.

More precisely, let $K \geq 1$ be a number which we will call the *promotion threshold*. Let $H = \langle (a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k) \rangle$ be the current history. For each condition $Cond_j \in \mathcal{C}$, let I_j be the set of indexes such that for all $i \in I_j$, $i < k$ and $Cond_j(\langle (a_1, b_1), (a_2, b_2), \ldots, (a_i, b_i) \rangle) = \mathsf{True}$. Let $\hat{I}_j$ be the set of the largest K indexes in I_j. If $|I_j| \geq K$ and for all $i \in \hat{I}_j$, $b_{i+1} = C$ (i.e., the other player chose C when the previous history up to the i'th iteration satisties $Cond_j$), then we will hypothesize that the other player will choose C whenever $Cond_j$ is satisfied; hence we will use $Cond_j \rightarrow 1$ as a deterministic rule. Likewise, if $|I_j| \geq K$ and for all $i \in \hat{I}_j$, $b_{i+1} = D$, we will use $Cond_j \rightarrow 0$ as a deterministic rule. See Line 7 to Line 10 in Figure 1 for an outline of the induction method we use in DBS.

The induction method can be faster at learning deterministic rules than the discounted frequency technique (see Figure 2). A faster learning speed allows us to infer the deterministic rules with a shorter history. The promotion threshold K controls the speed of the identification of deterministic rules. The larger the value of K, the slower the speed of identification, but the less likely we will mistakenly hypothesize that the other player's behavior is deterministic.

5.4 Symbolic Noise Detection and Temporary Tolerance in DBS

Once DBS has identified the set of deterministic rules, it can readily use them to detect noise. As we said earlier, if the other player's move violate a deterministic rule, it can be caused either by noise or by a change in the other player's behavior, and DBS uses an evidence collection process to figure out which is the case. More precisely, once a deterministic rule $Cond_i \rightarrow o_i$ is violated (i.e., the history up to the previous iteration satisfies $Cond_i$ but the other player's move in the current iteration is different from o_i), DBS keeps the violated rule but marks it as violated. Then DBS starts an *evidence collection process* that in the implementation of our competition entries is a violation counting: for each violated probabilistic rule DBS maintains a counter called the *violation count* to record how many violations of the rule have occurred (Line 12). In the subsequent iterations, DBS increases the violation count by one every time a violation of the rule occurs. However, if DBS encounters a positive example of the rule, DBS resets the violation count to zero and unmark the rule (Line 11). If any violation count excesses a threshold called the *violation threshold*, DBS concludes that the violation is not due to noise; it is due to a change of the other player's behavior. In this case, DBS invokes a special procedure (described in Section 5.5) to handle this situation (Line 13).

The drawback of the discount frequency method is that it fails to deal with several misinterpretations caused by noise within a small time interval. The way to avoid this drawback is to set a larger violation threshold. But if the threshold is too large, it will slow down the speed of adaptation to changes in the other player's behavior. In the competition, we entered DBS several times with several different violation thresholds; and in the one that performed the best, the violation threshold was 4.

5.5 Coping with Ignorance of the Other Player's New Behavior

When the evidence collection process detects a change in the other player's behavior, DBS knows little about the other player's new behavior. How DBS copes with this ignorance is critical to its success.

DBS maintains two sets of deterministic rules: the *current rule set* R_c and the *default rule set* R_d. Symbolic noise detection and temporary tolerance makes use of the rules in R_c but not the rules in R_d. However, DBS makes use of the rules in both R_c and R_d when DBS decides the next move (Line 18). More precisely, when DBS constructs a hypothesized policy π for move generation, it uses every rule in R_c and R_d. In addition, for any missing rule (i.e., the rule those condition are different from any rule's condition in R_c or R_d), we regard it as a probabilistic rule and approximate its degree of cooperation by Equation 1 (Line 17). These probabilistic rules form the probabilistic rule set $R_p \subseteq \psi_{k+1}$.

The default rule set is designed to be *rejected*: we maintain a violation count to record the number of violations to any rule in R_d. Every time any rule in R_d is violated, the violation count increased by 1 (Line 14). Once the violation count exceeds a *rejection threshold*, we drop the default rule set entirely (set it to an empty set) and reset the violation count (Line 15 and Line 16). We also reject R_d whenever any rule in R_c contradicts any rule in R_d (Line 15).

At the beginning of a game, R_d is R_{TFT} and R_c is empty (Line 1 and Line 2). While DBS can insert any newly found deterministic rule in R_c, it insert rules into R_d *only when* the evidence collection process detects a change of the other player's behavior. When it happens, DBS copies all the rules in R_c to R_d, and then set R_c to an empty set (Line 13).

Table 1: Scores of the top 20 programs, averaged over the five runs in Competition 2.

Rank	Program	Avg. score
1	BWIN	433.8
2	IMM01	414.1
3	DBSz	408.0
4	DBSy	408.0
5	DBSpl	407.5
6	DBSx	406.6
7	DBSf	402.0
8	DBStft	401.8
9	DBSd	400.9
10	lowESTFT_classic	397.2
11	TFTIm	397.0
12	Mod	396.9
13	TFTIz	395.5
14	TFTIc	393.7
15	DBSe	393.7
16	TTFT	393.4
17	TFTIa	393.3
18	TFTIb	393.1
19	TFTIx	393.0
20	mediumESTFT_classic	392.9

We preserve the rules in R_c mainly for sake of providing a smooth transition: we don't want to convert all deterministic rules to probabilistic rules at once, as it might suddenly alter the course of our moves and lead to uncertain outcomes. Furthermore, some of these rules may still hold, and we don't want to learn them from scratch.

6. COMPETITION RESULTS

The 2005 IPD Competition was actually a set of four competitions, each for a different version of the IPD. The one for the Noisy IPD was Competition 2, which used a noise level of 0.1.

Within Competition 2 there were five runs of 200 iterations each. Table 1 shows the average scores of the top twenty programs, averaged over the five runs. The competition website at http://www.prisoners-dilemma.com gives a more extensive set of tables showing each program's ranking for each of the five runs.

We entered nine different versions of DBS into the competition, each with a different set of parameters or different implementation. The one that performed best was DBSz, which makes use of the set of features we mentioned in this paper. Versions of DBS that had fewer features or additional features than DBSz did not do as well, but all nine of versions of DBS were in the top 25, and they dominated the top ten places. This implies that DBS's performance is insensitive to the parameters in the programs and the implementation details of an individual program.

DBSz is ranked third in the table; it lost only to BWIN and IMM01. BWIN and IMM01 both used the *master-and-slaves* strategy, in which a collection of *slave* programs sacrifice their own performance in order to give as many points as possible to another program called the *master* program.

Each participant was allowed to submit 20 programs. Each of the three master-and-slaves strategies (the third one, CNGF, was not good enough to make it into our table) has one master and 19 slaves. At the start of a run, the master and slaves would each make a series of moves using a predefined protocol, in order to identify themselves to each other. From then on, the master program would always play "defect" when playing with the slaves and the slave programs would always play "cooperate" when playing with the master, so that the master would gain the highest possible payoff at each iteration. Furthermore, a slave would alway plays "defect" when playing with a program other than the master, in order to try to minimize that player's score.

DBS does not use a master-slave strategy, nor does it conspire with other programs in any other way. Nonetheless, DBS remained competitive with the master-slave strategies in the competition, and performed much better than the master-slave strategies if the score of each master is averaged with the scores of its slaves. In particular, the average score for BWIN and its slaves was 379.9, and the average score for IMM01 and its slaves was 351.7. Furthermore, a more extensive analysis [1] shows that if the size of each master-and-slaves team had been limited to less than 10, DBSz would have outperformed the best master-slave strategy in the competition, even without averaging the score of each master with its slaves.

To show the destructive effect of the master-and-slaves strategies, Table 2 gives the percentages of each of the four possible outcomes when any program from one group plays with any program from another group. Note that:

- When BWIN and IMM01 play with their slaves, about 64% and 47% of the outcomes are (D, C), but when non-master-and-slave strategies play with each other, only 19% of the outcomes are (D, C).
- When the slave programs play with non-master-and-slaves programs, over 60% of outcomes are (D, D), but when non-master-and-slaves programs play with other non-master-and-slaves programs, only 31% of the outcomes are (D, D).
- The master-and-slaves strategies decrease the overall percentage of (C, C) from 31% to 13%, and increase the overall percentage of (D, D) from 31% to 55%.

Next, we consider how DBSz performs against TFT and TFTT, (the latter, Tit-For-Two-Tats, is like TFT except that it retaliates only when the other player defected in both of the last two iterations).

- Table 2 shows that when playing with another cooperative

Table 2: Percentages of different outcomes. *"All but M&S"* means all 105 programs that did not use master-and-slaves strategies, and *"all"* means all 165 programs in the competition.

Player 1	Player 2	(C, C)	(C, D)	(D, C)	(D, D)
BWIN	BWIN's slaves	12%	5%	64%	20%
IMM01	IMM01's slaves	10%	6%	47%	38%
CNGF	CNGF's slaves	2%	10%	10%	77%
BWIN's slaves	*all but M&S*	5%	9%	24%	62%
IMM01's slaves	*all but M&S*	7%	9%	23%	61%
CNGF's slaves	*all but M&S*	4%	8%	24%	64%
TFT	*all but M&S*	33%	20%	20%	27%
DBSz	*all but M&S*	54%	15%	13%	19%
TFTT	*all but M&S*	55%	20%	11%	14%
TFT	*all*	23%	19%	16%	42%
DBSz	*all*	36%	14%	11%	39%
TFTT	*all*	38%	21%	10%	31%
all but M&S	*all but M&S*	31%	19%	19%	31%
all	*all*	13%	16%	16%	55%

player, TFT cooperates ((C, C) in the table) 33% of the time, DBSz does so 54% of the time, and TFTT does so 55% of the time. Furthermore, when playing with a player who defects, TFT defects ((D, D) in the table) 27% of the time, DBSz does so 19% of the time, and TFTT does so 14% of the time. From this, one might think that DBSz's behavior is somewhere between TFT's and TFTT's.

- But on the other hand, when playing with a player who defects, DBSz cooperates ((C, D) in the table) only 15% of the time, which is a lower percentage than for TFT and TFTT (both 20%). Since cooperating with a defector generates no payoff, this makes TFT and TFTT perform worse than DBSz overall. DBSz's average score was 408 and it ranked 3rd, but TFTT's and TFT's average scores were 388.4 and 388.2 and they ranked 30th and 33rd.

7. RELATED WORK

Early studies of the effect of noise in the Iterated Prisoner's Dilemma focused on how TFT, a highly successful strategy in noise-free environments, would do in the presence of noise. TFT is known to be vulnerable to noise; for instance, if two players use TFT at the same time, noise would trigger long sequences of mutual defections [18]. A number of people confirmed the negative effects of noise to TFT [4, 6, 7, 18, 19, 20]. Axelrod found that TFT was still the best decision rule in the rerun of his first tournament with a one percent chance of misperception [2, page 183], but TFT finished sixth out of 21 in the rerun of Axelrod's second tournament with a 10 percent chance of misperception [10]. In Competition 2 of the 2005 IPD competition, the noise level was 0.1, and TFT's overall average score placed it 33rd out of 165.

The oldest approach to remedy TFT's deficiency in dealing with noise is to be more forgiving in the face of defections. A number of studies found that more forgiveness promotes cooperation in noisy environments [7, 19]. For instance, Tit-For-Two-Tats (TFTT) retaliates only when it receives two defections in two previous iterations. TFTT can tolerate isolated instances of defections caused by noise and is more readily to avoid long sequences of mutual defections. However, TFTT is susceptible to exploitation of its generosity and was beaten in Axelrod's second tournament by TESTER, a strategy that may defect every other move. In Competition 2 of the 2005 IPD Competition, TFTT ranked 30—a slightly better ranking than TFT's. In contrast to TFTT, DBS can tolerate not only an isolated defection but also a sequence of defections caused by noise, and at the same time DBS monitors the other player's behavior and retaliates when exploitation behavior is detected (i.e., when the exploitation causes a change of the hypothesized policy, which initially is TFT).

[18] proposed to mix TFT with ALLC to form a new strategy which is now called Generous Tit-For-Tat (GTFT) [22]. Like TFTT, GTFT avoids an infinite echo of defections by cooperating when it receives a defection in certain iterations. The difference is that GTFT forgives randomly: for each defection GTFT receives it randomly choose to cooperate with a small probability (say 10%) and defect otherwise. DBS, however, does not make use of forgiveness explicitly as in GTFT; its decisions are based entirely on the hypothesized policy that it learned. But temporary tolerance can be deemed as a form of forgiveness, since DBS does not retaliate immediately when a defection occurs in a mutual cooperation situation. This form of forgiveness is carefully planned and there is no randomness in it.

Another way to improve TFT in noisy environments is to use contrition: unilaterally cooperate after making mistakes. However, this is less useful in the Noisy IPD since a program does not know whether its action is affected by noise or not.

A family of strategies called "Pavlovian" strategies, or simply called Pavlov, was found to be more successful than TFT in noisy environments [15, 16, 17, 21]. When an accidental defection occurs, Pavlov can resume mutual cooperation in a smaller number of iterations than TFT [15, 16]. Pavlov learns by conditioned response through rewards and punishments; it adjusts its probability of cooperation according to the previous outcome. Like Pavlov, DBS learns from its past experience. DBS, however, has an intermediate step between learning from experience and decision making: it maintains a model of the other player's behavior, and uses this model to reason about nosie.

The use of opponent modeling is common in games of imperfect information such as Poker [5, 8, 9] and RoShamBo [12]. There have been many works on learning the opponent's strategy in the non-noisy IPD [11, 13, 23]. By assuming the opponent's next move depends only on the outcomes of the last few iterations, these works model the opponent's strategy as probabilistic finite automata, and then use various learning methods to learn the probabilities in the automata. In contrast, DBS does not aim at learning the other player's strategy completely; instead, it learns the other player's recent behavior, which is subject to change. We believe that the other player may alter its strategy at the middle of a game, and therefore it is difficult for any learning method to converge. It is essentially true in noisy IPD, since noise can provoke the other player. Moreover, the length of a game is usually too short for us to learn any non-trivial strategy completely.

To the best of our knowledge, ours is the first work on using opponent models in the IPD to detect errors in the execution of another agent's actions.

8. SUMMARY AND FUTURE WORK

For conflict prevention in noisy environments, a critical problem is to distinguish between situations where another player has misbehaved intentionally and situations where the misbehavior was accidental. That is the problem that DBS was formulated to deal with.

DBS's impressive performance in the 2005 Iterated Prisoner's Dilemma competition occurred because DBS was better able to maintain cooperation in spite of noise than any other program in the competition.

To distinguish between intentional and unintentional misbehaviors, DBS uses a combination of symbolic noise detection plus temporary tolerance: if an action of the other player is inconsistent with the player's past behavior, we continue as if the player's behavior has not changed, until we gather sufficient evidence to see whether the inconsistency was caused by noise or by a genuine change in the other player's behavior.

Since clarity of behavior is an important ingredient of long-term cooperation in the IPD, most IPD programs have behavior that follows clear deterministic patterns. The clarity of these patterns made it possible for DBS to construct policies that were good approximations of the other players'

strategies, and to use these policies to fend off noise.

We believe that clarity of behavior is also likely to be important in other multi-agent environments in which agents have to cooperate with each other. Thus it seems plausible that techniques similar to those used in DBS may be useful in those domains.

In the future, we are interested in studying the following issues:

- The evidence collection process takes time, and the delay may invite exploitation. For example, the policy of temporary tolerance in DBS may be exploited by a "hypocrite" strategy that behaves like TFT most of the time but occasionally defects even though DBS did not defect in the previous iteration. DBS cannot distinguish this kind of intentional defection from noise, even though DBS has built-in mechanism to monitor exploitation. We are interested to seeing how to avoid this kind of exploitation.

- In multi-agent environments where agents can communicate with each other, there is a probability for the agents to detect noise by a predefined communication protocol. However, we believe there is no protocol that can guarantee to tell which action has been affected by noise, as long as the agents cannot completely trust each other. It would be interested to compare these alternative approaches with symbolic noise detection and see how symbolic noise detection could enhance these methods or vice versa.

- The type of noise in the competition assumes that no agent know whether an execution of an action has been affected by noise or not. Perhaps there are situations in which some agents may be able to obtain partial information about the occurrence of noise. For example, some agents may obtain a plan of the malicious third party by counter-espionage. We are interested to see how to utilitize these information into symbolic noise detection.

9. ACKNOWLEDGMENTS

This work was supported in part by ISLE contract 0508268818 (subcontract to DARPA's Transfer Learning program), UC Berkeley contract SA451832441 (subcontract to DARPA's REAL program), and NSF grant IIS0412812. The opinions in this paper are those of the authors and do not necessarily reflect the opinions of the funders.

10. REFERENCES

[1] T.-C. Au and D. Nau. An Analysis of Derived Belief Strategy's Performance in the 2005 Iterated Prisoner's Dilemma Competition. Technical Report CSTR-4756/UMIACS-TR-2005-59, University of Maryland, College Park, 2005.

[2] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.

[3] R. Axelrod. *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton University Press, 1997.

[4] R. Axelrod and D. Dion. The further evolution of cooperation. *Science*, 242(4884):1385–1390, 1988.

[5] L. Barone and L. While. Adaptive learning for poker. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 566–573, 2000.

[6] J. Bendor. In good times and bad: Reciprocity in an uncertain world. *American Journal of Politicial Science*, 31(3):531–558, 1987.

[7] J. Bendor, R. M. Kramer, and S. Stout. When in doubt... cooperation in a noisy prisoner's dilemma. *The Journal of Conflict Resolution*, 35(4):691–719, 1991.

[8] D. Billings, N. Burch, A. Davidson, R. Holte, and J. Schaeffer. Approximating game-theoretic optimal strategies for full-scale poker. In *IJCAI*, pages 661–668, 2003.

[9] A. Davidson, D. Billings, J. Schaeffer, and D. Szafron. Improved opponent modeling in poker. In *Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI'2000)*, pages 1467–1473, 2000.

[10] C. Donninger. *Paradoxical Effects of Social Behavior*, chapter Is it always efficient to be nice?, pages 123–134. Heidelberg: Physica Verlag, 1986.

[11] D. W. Dyer. Opponent modelling and strategy evolution in the iterated prisoner's dilemma. Master's thesis, School of Computer Science and Software Engineering, The Univ. of Western Australia, 2004.

[12] D. Egnor. Iocaine powder explained. *ICGA Journal*, 23(1):33–35, 2000.

[13] P. Hingston and G. Kendall. Learning versus evolution in iterated prisoner's dilemma. In *Proceedings of the Congress on Evolutionary Computation*, 2004.

[14] G. Kendall, P. Darwen, and X. Yao. The iterated prisoner's dilemma competition. http://www.prisoners-dilemma.com, 2005.

[15] D. Kraines and V. Kraines. Pavlov and the prisoner's dilemma. *Theory and Decision*, 26:47–79, 1989.

[16] D. Kraines and V. Kraines. Learning to cooperate with pavlov an adaptive strategy for the iterated prisoner's dilemma with noise. *Theory and Decision*, 35:107–150, 1993.

[17] D. Kraines and V. Kraines. Evolution of learning among pavlov strategies in a competitive environment with noise. *The Journal of Conflict Resolution*, 39(3):439–466, 1995.

[18] P. Molander. The optimal level of generosity in a selfish, uncertain environment. *The Journal of Conflict Resolution*, 29(4):611–618, 1985.

[19] U. Mueller. Optimal retaliation for optimal cooperation. *The Journal of Conflict Resolution*, 31(4):692–724, 1987.

[20] M. Nowak and K. Sigmund. The evolution of stochastic strategies in the prisoner's dilemma. *Acta Applicandae Mathematicae*, 20:247–265, 1990.

[21] M. Nowak and K. Sigmund. A strategy of win-stay, lose-shift that outperforms tit-for-tat in the prisoner's dilemma game. *Nature*, 364:56–58, 1993.

[22] M. A. Nowak and K. Sigmund. Tit for tat in heterogeneous populations. *Nature*, 355:250–253, 1992.

[23] R. Powers and Y. Shoham. Learning against opponents with bounded memory. In *IJCAI*, 2005.

On using Multi-agent Systems in Playing Board Games

Stefan J. Johansson
Department of Software Systems
School of Engineering
Blekinge Institute of Technology
sja@bth.se

ABSTRACT

Computer programs able to play different kinds of games (aka bots) is a growing area of interest for the computer game industry as the demand for better skilled computerized opponents increase. We propose a general architecture of a Multi-agent System (MAS) based bot able to play complex board games and show that this solution is able to outperform other bots in two quite different games, namely no-press Diplomacy and Risk. Based on these results, we formulate a hypothesis of the applicability of MAS based bots in the domain of board games and identify the need for future investigations in the area.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: [Multiagent systems, Coherence and coordination]

General Terms

Algorithms

Keywords

Multiagent system architecture, Board games, RISK, Diplomacy

1. INTRODUCTION

Creating programs able to play classical strategic games such as Othello, Chess, Risk, Diplomacy and Go has since the early days of AI been a challenge [6]. Some of these, Chess, and especially Othello, Checkers, Backgammon are now considered *solved* in the sense that the programs are able to beat the top human players [1, 10, 14, 19], while games such as Go, Risk and Diplomacy still remain among the ones that we have a hard time to find an optimal computerized player for. The main problem of the latter games is the branching of possible moves. To take one example, the branching factor of chess is 20 for the first move ($20^2 = 400$ for the first round of moves), whereas in Diplomacy, there are 4430690040914420 unique openings [11]. This makes it intractable to use traditional minimax-based algorithms to search for good moves in the games.

Previous attempts to make agent based solutions to games include:

- Chess - Drogoul [3] and Fransson [4] have tried to create MASs playing Chess based on various kinds of negotiations between the Chess pieces, rather than traditional minimax search algorithms, but the performance of these systems is not overwhelming.
- Stratego - Treijtel made a MAS based Stratego player where the system implemented a distributed rule based decision system [22].
- Diplomacy - Kraus and Lehmann [9] and lately also Shaheed [15] have made heterogeneous MASs of agents able to negotiate with other players.
- No-press Diplomacy - Johansson and Håård show promising results in this version of Diplomacy with a bot based on a homogeneous MAS [7].
- Risk - Johansson and Olsson made a MAS playing Risk which showed to be successful [8].

The latter two will be described more in detail in Section 5. In the next section we will describe a general model of games, followed by a general MAS architecture in Section 3. We will then in Section 6 discuss how to evaluate MAS based bots before we finish with a discussion and draw some conclusions.

2. GENERAL MODEL OF THE GAME

We have chosen to limit this investigation to games which have the following properties:

1. It has a set of starting players S, where $|S|$ is either fixed, or within a certain interval.
2. The game has a set of territories T with an adjacency set $A \subseteq T \times T$.
3. The game has a set P of different types of pieces.
4. Each type of piece $p \in P$, may move according to $A_p \subseteq A$
5. The game is divided into a number of synchronized turns $T = \{t_0, t_1, ...\}$.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

6. Each turn may be carried out in the sequence of the players, or in parallel.

7. The turns are divided into (at least) three phases: initial phase, action phase(s), and finish phase.

8. In the initial phase, I, new resources (if available) are entered into the system and preparations are made for the actions of the coming phases.

9. In an action phase, A, the player decides what actions to take and these are then performed according to the rules of the game.

10. In the finish phase, F, the player make preparations for the next phase.

In addition to this, each game may have a number of other properties, e.g.

- Typically, the state of the game restricts A_p even more.
- The state of the game may be partially unknown to the players.
- The territories may be occupied by none, one or several pieces at the same time.
- The adjacency matrix may be symmetric, or arbitrary.
- The game may have non-deterministic outcomes of the actions of the players.
- Each turn may consist of a single move, or a (dynamically set) sequence of moves.
- The game can be exogenous, zero-sum, or endogenous (in the sense that resources are enriched by the game, staying equal throughout the game, or being consumed in the game).

2.1 An example of a game

As an exemplification of our general model, let us take a closer look at the well-known game of chess. It is a game with six types of pieces (King, Queen, Bishop, Knight, Tower and Pawn). Each type has its own adjecency matrix[1], but the moves that are possible to make are further restricted by the state of the game (e.g. a pawn may only move straight forward unless the space in front of it is not occupied). Each turn, which are carried out in sequence, consists of an empty initial phase, a one-move action phase, and an empty finish phase (unless the players are using a clock). The outcome of an action is deterministic and the state of the game is known to its two players. The game is of the endogenous type and the territories may at most hold at most one piece at a time.

3. GENERAL MODEL OF THE BOT

The general idea is to have a multi-agent system where each important unit of the player on the board corresponds to an agent in the system. These agents may either correspond to the pieces or the territories of the player(s). The agents will then negotiate about what actions to take in the system, possibly helped by a mediator agent taking care of the actual negotiation (e.g. carry out an auction) through a negotiator mechanism. The architecture is depicted in Figure 1.

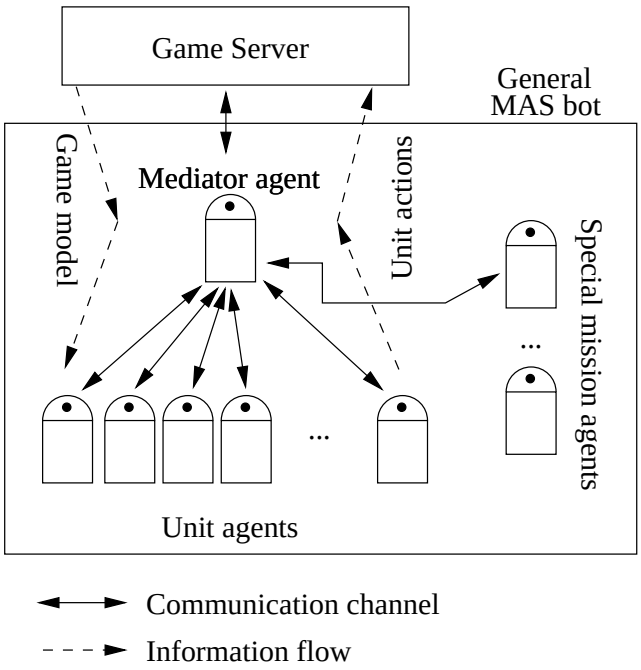


Figure 1: The agent system of the general model.

Typically, the mediator agent is also propagating the information from the agent system (i.e. what actions to perform) to the game server, as well as spreading the information from the server (e.g. the moves of the opponents) to the agent system (see the information router in Figure 2). This includes translating the information from its internal representation, to the external format used by the present game server and vice versa. This task is dedicated to the game server interface.

There may also be other agents in the system with dedicated missions, such as keeping track of the time (if it is a time critical game) or holding properties of the player that is not associated with a certain unit agent, e.g. opponent modeling.

In addition, there may be environment variables or parameters common to the whole MAS. These may be static throughout the game, or dynamically changed by the game server, or the agents of the system. Examples of such variables may be attack and defence weights, territory valuations, or algorithmic parameters (such as search depths).

Note that the proposed architecture is not restricted to the general model of the game outlined in Section 2. It would be possible to have play games that involve skipping turns or change the order of the play.

4. TWO INSTANCES OF GAMES

We will now (briefly) describe two games, no-press Diplomacy and Risk[2], which on the surface seems to be similar, but that are quite different in the way they are played. Among the similarities, the goal of the players in both games

[1]In fact, the bishop has two: one for the white squares, and one for the black squares

[2]Both games are ©Hasbro inc.

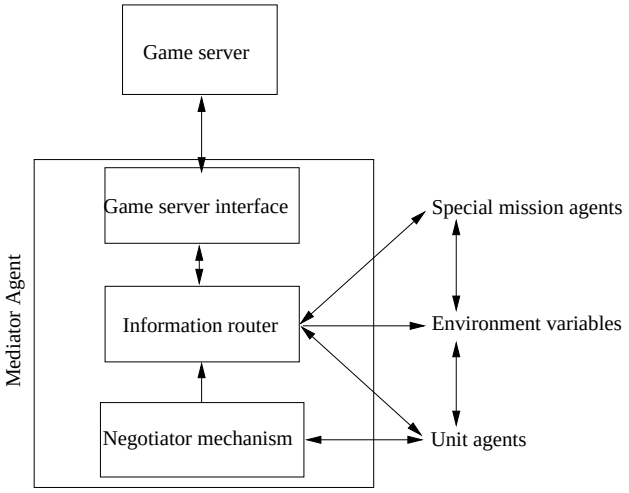


Figure 2: The mediator agent of the general bot

Property	Diplomacy	Risk	Chess
# players	7	2–6	2
Phases	$IAFAF$	IA^*F	A
Moves	parallel one-step	sequential multi-step	sequential one-step
Outcome	opponent dependent	dice-based	deterministic
Pieces	two types	single type	six types
Board	map	map	8x8
Pieces/prov.	0–1	1–∞	0–1

Table 1: Differences between the games Diplomacy, Risk and Chess. The phases are *Initial, Action* and *Finish* (IAF).

is to conquer the map by defeating their opponents, but that is where the differences begin. As a reference we have included Chess as well in the comparison (see Table 1).

We will now describe some central details of the games.

4.1 The Characteristics of no-press Diplomacy

In no-press Diplomacy, seven players each control one of the great powers in the 1901 Europe (see Figure 3). The players control armies and fleets and the goal is to take control over the majority of some special territories (the production centers). All pieces of all players act in parallell in the action phases, either by moving, supporting, cutting opponent support (all of these to neighbouring territories only), or standing still. For a move to be successful, the moving piece must be superior in uncut support if the opponent also tries to move to that position (or already resides there). After each phase where the pieces try to act, a finish phase decides what moves that were successful, and what moves that were not. After two rounds of movements, the production center ownerships decide what powers that will get new pieces to place at their territories (this finish one year in the game). To avoid stale mates in the games (especially when bots are playing), a common solution is to finish the game in 1950 or when no production centers have changed owners for the last five years.

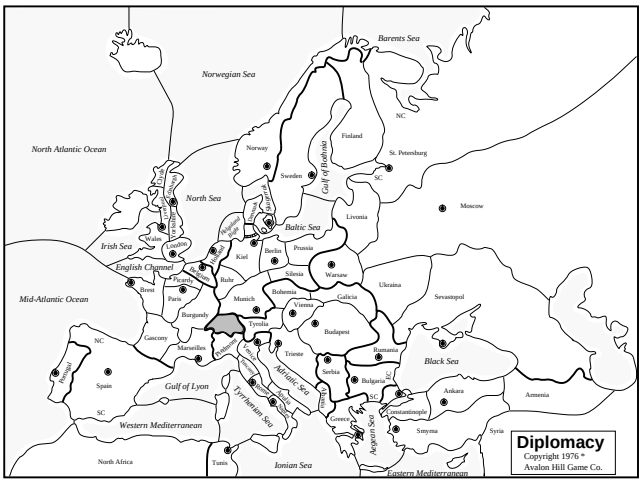


Figure 3: The map of the game of Diplomacy



Figure 4: The map of the game of RISK

4.2 The Characteristics of Risk

Risk is a game of two to six players, where the objective is to take control over the world (see Figure 4). This is achieved through invading opponent territories, and taking control over continents.[3] The players act in turns of three phases. In the initial phase, the player will get extra armies to place in its territories based upon the number of territories and the continents that they own. During the action phases, the player battles against opponents and tries to invade neighbour territories. This may be repeated for as long as the player has any armies left to attack with. In the finish phase, the player may move its armies. However, each army may only move one step and there must be at least one army left in each territory. If at least one battle has been successful, the player gets a bonus card that may be used in combination with other cards in the coming initial phases to get extra armies.

5. TWO INSTANCES OF BOTS

Following the general architecture of Section 3, we will now describe two bots able to play no-press Diplomacy and

[3]Many modern rule variations use special mission cards (where different players get different missions) to shorten the time it takes to play the game. However, we will not use missions cards in this work, but regard the total world domination as the common goal of all players.

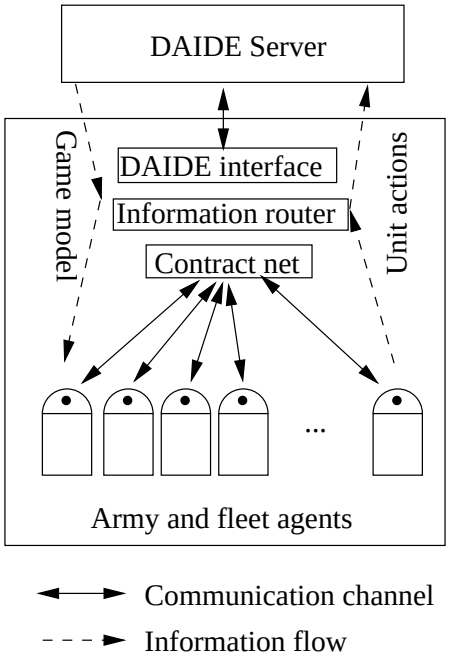


Figure 5: The no-press Diplomacy bot HaAI

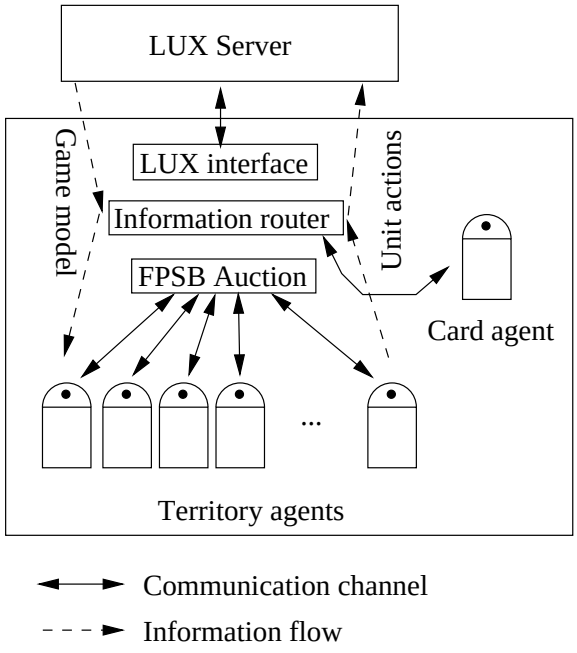


Figure 6: The RISK bot MARS

Risk respectively. For further details on the implementation we refer to previous work by Johansson and Håård [7] and Johansson and Olsson [8].

5.1 HaAI — A no-press Diplomacy Bot

The HaAI bot consists of a mediator agent and a set of unit agents. The mediator agent has an interface to the game server (in our case the DAIDE server [2]) and an information router that distributes the information going in and out of the system. To coordinate the actions of the units, a contract net-mechanism is used [17]. Here the units suggest prioritized tasks and adapt to support such tasks until all agents are doing the best they can to expand their territories and defend themselves from the opponent forces, see Figure 5.

The unit agents correspond to the armies and fleets of the player, so that each unit has the possibility to decide by itself what action to take (to move, to support the movement of another unit, or to cut the support of an opponent unit). A number of parameters are used in the evaluation of the different candidate actions [7].

5.2 MARS — A Risk Bot

The MARS bot consists of a mediator agent with a LUX interface, an information router, and a first price sealed bid (FPSB) auction mechanism that handles the coordination between the unit agents. In MARS, each territory has an agent (as opposed to each piece owned by the player, as was the case in the HaAI bot). A dedicated card agent keeps track of the cards of the player and exchange them for new armies in an optimal way (see Figure 6).

The initial phase starts by letting all territories (the own as well as the ones held by the opponents) calculate the value of holding that particular territory [8]. Such an approach, using the territories rather than the pieces, is as far as we know novel in the field of bots for board games. In the case of the own territory, this is a defensive bid, in the case of an opponent territory, the information of how many armies that defends it and the value of owning it is propagated to all neighbours. Their values and armies are added, the path is added (possible cycles are removed), and the information is propagated further in a recursive way (up to a predefined depth). When the paths with the values and number of defending armies reach an own territory, it is considered by the territory as an offensive bid. The agents of the own territories then calculate the accumulated probabilities of reaching the goals given the different numbers of armies that they may use in the attacks. The bids with the highest ratio of value/(expected number of armies needed to succeed) will get their share of the new armies. The card agent will also possibly cash in some extra armies for the system to use at this stage.

An action phase consists of a battle, where the plans of the winning bids of the initial phase are realised. Such plans may be single step plans, but often they consist of several battles carried out in sequence.

In the finish phase, the own territory agents use a distributed max flow algorithm to find the optimal movements of the armies. If MARS has won any battle during the action phase(s), the card agent will get a new card.

6. ON THE EVALUATION OF BOTS

We have used open bot tournaments as a way to measure the performance of our solutions. In both the Diplomacy and the Risk communities, there are bot developers that are willing to try their solutions in competition with others. Although previous work has presented successful attempts to use humans in training and/or evaluating bots (see e.g. [20]), there are a few reasons why we have chosen not to compare the solution with human players in these tournaments:

1. Bot tournaments can be batched, humans not. It is easy to set up a round robin tournament with say 1000 matches using a simple script, but to get a human player to participate to the same extent is impossible.

2. Bots are in general faster than human players, thus decreasing the time to get to the result.

3. Bots are blind and objective, while humans may decide to eliminate the bots first (just because they are bots). This *bots first strategy* is highly developed in the Risk (LUX) community. Unfortunately, such a strategy makes it hard to do a fair evaluation of the strengths of the different players.

In both cases we called for the participation of an open tournament through the fora and mailing lists of the communities. We invited the designers of bots for the largest game server platforms of the games (DAIDE and LUX respectively). The tournament was then run in a Round Robin fashion (i.e. everyone met every other set of opponents) and the scores of all matches were noted. We also measured the time it took for the bots to play the games in order to be able to compare the efficiency of the different solutions.

6.1 The Results of the no-press Diplomacy Tournament

A total of 592 matches were run on DAIDE out of which 487 ended with a single winner and 105 matches were draws. In case of a draw (no one has won after 50 simulated years), the score of seven points is divided between the remaining players. We see (in Table 2) that the two versions of HaAI manage quite well, ending up at the first and third place in the tournament.

What the efficiency is concerned, according to Table 3, HaAI does not have the best figures (not the worst either). It is clearly the fastest Java-based bot among the participators, but not as fast as the C++-implemented *DumbBot 2*.

6.2 The Results of the RISK Tournament

The RISK tournament was run on the LUX platform which is the platform for which there are the greatest number of bots available [16]. Twelve bots participated in addition to our MARS bot and played 792 matches each. MARS was a clear winner of this tournament with more than 50 per cent more wins than the second and third placed *Bort* and *EvilPixie*.

When it comes to efficiency, MARS still comes out at the top of the others. If we compare the amount of calculations needed (at an average) to win a game, we see that the ratio Wins/(average time per turn) is higher for MARS (even though it is among the slowest bots, see Table 5). This shows that MARS uses an efficient way of calculating good RISK moves.

Bot	Matches won	Solo score	Total score
HaAI 0.63 Berserk	109	763	866.05
DiploBot 1.2	95	665	763.10
HaAI 0.63 Vanilla	81	567	724.64
Man'Chi AttackBot	75	525	658.32
DumbBot 2	87	609	651.34
Man'Chi DefenceBot	40	280	444.27
RandBot	0	0	0.06

Table 2: Scores of the Diplomacy tournament. The total score include the scores of the draw matches.

Bot type	Orders/s	Impl. Language
DumbBot 2	55.9	C++
HaAI 0.63	19.6	Java
Man'Chi 7	12.7	Java
DiploBot 1.2	3.7	Java

Table 3: Diplomacy bot performance measured in number of orders per second.

Name	1st	2nd	3rd	4th	5th	6th	D
MARS	338	169	60	62	69	84	10
Bort	206	47	130	156	149	100	4
EvilP.	205	56	105	153	138	133	2
Boscoe	184	52	142	174	157	76	7
Yakool	135	57	118	162	167	146	7
Quo	177	103	102	132	134	142	2
Pixie	118	153	102	123	139	154	3
Shaft	117	370	157	61	36	38	13
Cluster	100	137	165	161	118	105	6
Nefar.	84	115	128	141	168	156	0
Comm.	16	278	199	130	85	74	10
Stinky	7	139	204	141	147	149	5
Angry	6	17	89	113	207	358	2

Table 4: The outcome of the RISK tournament. D is the number of draws. Note that each bot played 792 matches.

7. DISCUSSION

We will now comment the results, and discuss the possible general conclusions that may be drawn from the experiments.

7.1 Comments on the results

The results of the tournaments raise a lot of questions. Were they mostly due to an intelligent setting of the parameters of the environment? Did we use the best opponents? How did our choice of game platform effect the behavior of our solution? What are the drawbacks of the approach?

Let us start with the parameters. Both MARS and HaAI use a wide range of parameters.[4] In the case of HaAI, a first *ad hoc* setting of the parameters was made and minor adjustments led to the two versions that entered the tournament[5]. MARS used a slightly more structured approach where we iteratively narrowed the search space of the two major parameters — the bonus for being offensive and the threshold probability for attacking [13]. Of course the values of the parameters play an important role when it comes to the performance of the bot. However, we have no reason to believe that we are superior in this respect (compared to our competitors). On the contrary, we believe that great improvements can be achieved by using machine learning

[4]HaAI has about 11 main parameters, and MARS has about 14.

Name	$Win\%$	$\overline{t_{match}}$	$\frac{turns}{match}$	$\overline{t_{turn}}$	$\frac{Win\%}{\overline{t_{turn}}}$
MARS	42.7	79.72	32.0	2.49	17.1
Boscoe	23.2	40.04	27.1	1.48	15.7
EvilP.	25.9	44.10	25.1	1.76	14.7
Bort	26.0	45.75	25.5	1.79	14.5
Quo	22.3	33.87	21.6	1.57	14.2
Pixie	14.9	40.04	29.9	1.34	11.1
Yakool	17.0	37.62	22.7	1.66	10.3
Shaft	14.8	74.47	34.5	2.16	6.86
Cluster	12.6	54.32	25.4	2.14	5.90
Nefar.	10.6	46.82	18.7	2.51	4.22
Comm.	2.0	49.37	29.1	1.70	1.18
Stinky	1.1	48.88	24.0	2.04	0.54
Angry	1.0	41.33	14.7	2.82	0.36

Table 5: The win percentage, average runtimes (in seconds) per match and per turn, and number of wins per hour of processing time in the RISK tournament.

Game	# units	action space	players	action reliability
Chess	small	small	2	full
Diplomacy	medium	medium	7	depends on other players
Risk	high	medium	2–6	dice based
Go	high	medium	2	full
Settlers of Catan	small	medium	4	dice based
Monopoly	single	tiny	2–6	dice based
RoboRally	single	small	2–6	depends on other players
Backgammon	small	small	2	dice based

Table 6: Comparison between a number of different board games.

approaches to find the right set of parameters, but that has not been the focus of this project.

The platforms were chosen based on the number (and quality) of opponents, and the documentation of the system. In both cases, the chosen game platforms were the ones which had the largest number of competitive opponent bots (among the ones we chose from).

The set of opponents is of course questionable. We encouraged the communities to send in their bots to us for participation in the tournaments; thus we did what we could to get the best possible opponents. However, this does not mean that there are not any bots out there that for one reason or another did not wish to participate e.g. due to bugs, or the unwillingness to share their programming secrets with us (see e.g. Håård [5]).

In all we believe that the successful results in the tournaments are to a great extent the results of a well implemented distributed problem solving mechanism based on the MAS architecture outlined in Section 3. Possibly the choice of parameters and the set of opponents may have had an additional positive effect on the results.

Among the drawbacks of our solution are that it may be hard (in its pure form) to implement strategic sacrifices, since all unit agents are trying to maximize their own utilities and they do only use a single step look-ahead. The way to address this problem may be to use special mission agents that focus on strategic issues through dynamic updates of the utility functions.

So, what are the key factors that makes our approach work better than the opponent bots in the investigated domains?

Our hypothesis is that game domains holding the following properties are especially well suited for MAS-based bots (compared to traditional search tree-based bots):

- *Large number of units in the game.* When there are a lot of units (i.e. armies or territories), each of which behavior may be modelled separately by an agent, the MAS bot seems to be a dynamic, but yet easily understood solution.
- *Large action space.* In both Risk and Diplomacy, the number of actions a player may take in each turn is considerably larger than e.g. in chess. It does of course not *help* us to have many options, but since our solutions do not rely on opponent modeling and thereby a search in the tree of possible outcomes (c.f. minimax), they do not get caught in the exponential explosion of possible states that such a search leads to.
- *Large number of players.* A large number of players furthermore increases the difficulty to predict the future state of the game since it increase the branching factor.
- *Unreliable outcome of the performed actions.* The more precisely the effect of the actions a player take can be predicted, the easier it is to predict the future state of the game.

Support for this hypothesis is to be found both in previous work on bots for variants of Diplomacy [7, 9, 15] and Risk [8], but also in the less successful attempts to make bots based on MAS principles for playing e.g. Chess [3, 4]. However, it is still too early to draw any conclusions based upon these initial studies. More work is needed in the area before we may consider MAS based bots in general to be strong solutions in complex board game playing while being less appropriate to use in games requiring long look-ahead.

8. CONCLUSIONS

Our conclusion is that the MAS architecture described here may be a suitable candidate for creating competitive bots able to play games with high branching factors that are distributed in nature. We support this claim by reporting positive results from testing our solution in two different domains, namely the game of Diplomacy and the game of Risk. Despite the success in these isolated domains, we identify the need for more research within the area in order to draw any wider conclusions.

9. FUTURE WORK

There is support for the claim that complex domains such as the ones outlined in Section 7.1 above are favoured by multi-agent solutions [12, 18]. MAS based bots have shown to work well in Diplomacy and Risk, but what about the games which do not hold all the properties of these two

games? Go is one such game. It has got a large number of possible moves and a large number of stones[5]. However, the game has only two players and the moves are fully deterministic. It is therefore an open problem whether Go can be successfully played by a MAS based bot.

Settlers of Catan[6] is another game that has reached a large audience during the last years, especially through its online versions, e.g. *Java Settlers* by Robert Shaun Thomas [21]. Thomas used single agent bots and although he does not rule out the possibility to build bots with distributed architectures, he claims the single agent solution to be simpler (and thus better referring to the *less is more* principle) [20].

We will in our future work examine what game properties that are the most significant when it comes to the results of MAS based solutions. This will be done by trying a number of different suitable board games, e.g. Go and Settlers of Catan.

There are also a number of possible improvements regarding the roles of the special mission agents, e.g.:

- *Opponent short term modeling*, i.e. dynamic real time updates of the environment variables in order to adjust to the current opponent strategies.

- *Opponent classification* By learning the characteristics of different players and store the classification between the games, we may at an earlier stage identify what player that is playing what power and update the dynamic variables faster than what would have been the case if we used short term modeling only.

- *Parameter optimization*; the parameters of the environment may be updated based on the results of the games. This may effect the willingness to attack, or the utility of holding certain territories in the longer perspective.

- *Added functionality* In Diplomacy (without the prefix no-press), the players are allowed (and encouraged) to cooperate in order to achieve their goals. The system of e.g. Kraus and Lehmann [9] uses additional agents such as *minister of defence*, and *secretary of the prime minister* to deal with the negotiations with the opponent players. Such functionality may be added to the proposed architecture as special mission agents.

Last, but not least, we would like to carry out experiments with human players in order to measure the strength of our solution compared to the one of human players.

10. ACKNOWLEDGEMENTS

The author would like to thank Fredrik Håård and Fredrik Olsson for their hard work on the implementation of the bots and the creative discussions and co-authoring that led to our previous publications [7, 8]. We would also like to thank the game communities of DAIDE and LUX for the resistance in the tournaments, the reviewers of AAMAS'06 for the valuable comments and Blekinge Institute of Technology for funding this project.

[5]the pieces you put on the board in Go

[6]Trademark of Mayfair Games

11. REFERENCES

[1] Murray Campbell, Jr. A. Joseph Hoane, and Feng hsiung Hsu. Deep blue. *Artif. Intell.*, 134(1-2):57–83, 2002.

[2] DAIDE web page, http://www.daide.org.uk/, February 2006.

[3] A. Drogoul. When ants play chess (or can strategies emerge from tactical behviours?). In C. Castelfranchi and J.-P. Müller, editors, *From Reaction to Cognition — Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-93 (LNAI Volume 957)*, pages 13–27. Springer-Verlag: Heidelberg, Germany, 1995.

[4] H. Fransson. Agentchess — an agent chess approach – can agents play chess? Master's thesis, Blekinge Institute of Technology, 2003.

[5] Fredrik Håård. Multi-agent diplomacy: Tactical planning using cooperative distributed problem solving. Master's thesis, Blekinge Institute of Technology, 2004.

[6] T.P. Hart and D.J. Edwards. The Tree Prune (TP) algorithm. Technical report, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1961. Artificial Intelligence Project Memo 30.

[7] S.J. Johansson and F. Håård. Tactical coordination in no-press diplomacy. In *Proceedings of Autonomous Agents and Multi-agent Systems (*AAMAS*)*, 2005.

[8] S.J. Johansson and F. Olsson. Mars a multi-agent system playing risk. In *Proceedings of Pacific Rim International Workshop on Multi-agents (PRIMA)*, 2005.

[9] Sarit Kraus and Daniel Lehmann. Designing and building a negotiating automated agent. *Computational Intelligence*, 11(1):132–171, 1995.

[10] Kai-Fu Lee and Sanjoy Mahajan. The development of a world class othello program. *Artif. Intell.*, 43(1):21–36, 1990.

[11] Daniel E. Loeb. Challenges in multi-player gaming by computers. *The Diplomatic Pouch Zine*, S1995M, 1995.

[12] M. Luck, P. McBurnley, and C. Preist. *Agent Technology: Enabling Next Generation Computing - A Roadmap for Agent Based Computing*. AgentLink, 2003. ISBN 0854 327886.

[13] Fredrik Olsson. A multi-agent system for playing the board game risk. Master's thesis, Blekinge Institute of Technology, 2005.

[14] J. Schaeffer. Solving checkers: First result. *International Computer Games Association (ICGA) Journal*, 28(1):32–37, 2005.

[15] J. Shaheed. Creating a diplomat. Master's thesis, Imperial College, London, UK, 2004.

[16] SillySoft. Lux v4.3, 2005. http://sillysoft.net/ URL last visited on 2006-01-24.

[17] R.G. Smith. The contract net protocol: High level communication and control in a distributed problem solver. *IEEE Transaction on Computers*, C-29(12):1104–1113, 1980.

[18] K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng. Distributed intelligent agents. *IEEE Expert*, 11(6):36–46, 1996.

[19] G. Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3), 1995.

[20] R.S. Thomas. *Real-time Decision Making for Adversarial Environments Using a Plan-based Heuristic*. PhD thesis, Northwestern University, Evanston, Illinois, 2003.

[21] R.S. Thomas. Jsettlers home page, 2006. http://catan.jsettlers.org/ URL last visited February '06.

[22] Caspar Treijtel. Multi-agent stratego. Master's thesis, Delft University of Technology, 2000.

Solution Sets for DCOPs and Graphical Games

Jonathan P. Pearce
Computer Science Dept.
Univ. of Southern California
Los Angeles, CA 90089
jppearce@usc.edu

Rajiv T. Maheswaran
Information Sciences Institute
Univ. of Southern California
Los Angeles, CA 90292
maheswar@isi.edu

Milind Tambe
Computer Science Dept.
Univ. of Southern California
Los Angeles, CA 90089
tambe@usc.edu

ABSTRACT

A distributed constraint optimization problem (DCOP) is a formalism that captures the rewards and costs of local interactions within a team of agents, each of whom is choosing an individual action. When rapidly selecting a single joint action for a team, we typically solve DCOPs (often using locally optimal algorithms) to generate a single solution. However, in scenarios where a set of joint actions (i.e. a set of assignments to a DCOP) is to be generated, metrics are needed to help appropriately select this set and efficiently allocate resources for the joint actions in the set. To address this need, we introduce *k*-optimality, a metric that captures the desirable properties of diversity and relative quality of a set of locally-optimal solutions using a parameter that can be tuned based on the level of these properties required. To achieve effective resource allocation for this set, we introduce several upper bounds on the cardinalities of *k*-optimal joint action sets. These bounds are computable in constant time if we ignore the graph structure, but tighter, graph-based bounds are feasible with higher computation cost. Bounds help choose the appropriate level of *k*-optimality for settings with fixed resources and help determine appropriate resource allocation for settings where a fixed level of *k*-optimality is desired. In addition, our bounds for a 1-optimal joint action set for a DCOP also apply to the number of pure-strategy Nash equilibria in a graphical game of noncooperative agents.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Design, Theory

Keywords

constraint reasoning, DCOP, multiagent systems, k-optimality

1. INTRODUCTION

In a large class of multi-agent scenarios, a set of agents chooses a joint action (JA) as a combination of individual actions. Often, the locality of agents' interactions means that the utility generated by each agent's action depends only on the actions of a subset of the other agents. In this case, the outcomes of possible JAs can be compactly represented in cooperative domains by a distributed constraint optimization problem (DCOP)[12, 20] or, in noncooperative domains, by a graphical game [6]. Each of these models can take the form of a graph (or hypergraph) in which each node is an agent and each edge (or hyperedge) denotes a subset of agents whose actions, when taken together, incur costs or rewards, either to the agent team (in DCOPs) or to individual agents (in graphical games). In the case of DCOP, if each agent controls a single variable, then a single JA is a complete assignment of values to variables (i.e. every agent chooses an individual action for itself). We focus primarily on the team setting, using DCOP, whose applications include multi-agent plan coordination [4], sensor networks [12], and RoboCup soccer [17].

Traditionally, researchers have focused on obtaining a single JA, expressed as a single assignment of actions to agents in a DCOP. However, in this paper, we consider a multi-agent system that generates a *set* of JAs, i.e. multiple assignments to the same DCOP. Generating sets of JAs is useful in domains such as disaster rescue (to provide multiple rescue options to a human commander) [15], patrolling (to execute multiple patrols in the same area) [14], training simulations (to provide several options to a student) and others [16]. We provide three key contributions to address such domains. The first contribution is to determine the appropriate metric for evaluating a set of JAs. While high absolute reward is an appropriate metric in single-solution domains, reward alone is a poor metric for these multiple-JA domains, as it often yields clusters of very similar JAs, as shown in Section 2 of this paper. Clustering is undesirable, as diversity, the difference among JAs, is a key property for a JA set in many domains [16]. Diversity alone is undesirable, because it leads to low-quality solutions. Hence, this paper introduces a new metric, *k*-optimality, that naturally captures the diversity and relative quality of a JA set. A *k*-optimal JA has the highest reward within a neighborhood of other JAs that differ from it by at most *k* individual actions; i.e. no *k*-optimal JA can be improved if *k* or fewer agents change actions. Therefore, *k*-optimality quantifies the neighborhood in which a JA is optimal. In a *k*-optimal JA set, defined as a set of JAs, each of which is itself *k*-optimal, all JAs in the set are guaranteed a particular level of relative quality (best in their neighborhoods), as well as diversity (every two JAs must be separated by at least $k + 1$ individual actions).

Domains requiring repeated patrols in an area by a team of UAVs (unmanned air vehicles), UGVs (unmanned ground vehicles), or robots, for peacekeeping or law enforcement after a disaster, provide one key illustration of the utility of *k*-optimality. For example, given a team of patrol robots in charge of executing multiple joint

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

patrols in an area as in [14], each robot may be assigned a region within the area. Each robot is controlled by a single agent, and hence, for one joint patrol, each agent must choose one of several possible routes to patrol within its region. A joint patrol is a JA, where each agent's action is the route it has chosen to patrol, and rewards and costs arise from the combination of routes patrolled by agents in adjacent or overlapping regions. For example, if two nearby agents choose routes that largely overlap on a low-activity street, the constraint between those agents would incur a cost, while routes that overlap on a high-activity street would generate a reward. Agents in distant regions would not share a constraint. If reward alone is used as a metric to select joint patrols, then all selected joint patrols could be the same, except for the action of one agent. This set of patrols would be repetitive and predictable to adversaries. If we pick some diverse joint patrols at random, they may be very low-quality patrols. Using *k*-optimality directly addresses such circumstances; *k*-optimality ensures that all joint patrols differ by at least $k + 1$ agents' actions, as well as ensuring that this diversity would not come at the expense of obviously bad joint patrols, as each is optimal within a radius of at least *k* agents' actions.

After introducing *k*-optimality, our second key contribution in this paper is addressing efficient resource allocation for the multiple JAs in a *k*-optimal set, by defining tight upper bounds on the number of *k*-optimal JAs that can exist for a given DCOP. These bounds are necessitated by two key features of the typical domains where a *k*-optimal set is applicable. First, each JA in the set consumes some resources that must be allocated in advance. Such resource consumption arises because: (i) a team actually executes each JA in the set, as in our patrolling example above, or (ii) the JA set is presented to a human user (or another agent) as a list of options to choose from, requiring time. In each case, resources are consumed based on the JA set size. Second, while the existence of the constraints between agents is known *a priori*, the actual rewards and costs on the constraints depend on conditions that are not known until runtime, and so resources must be allocated before the rewards and costs are known and before the agents generate the *k*-optimal JA set. In the patrolling domain, constraints are known to exist between patrol robots assigned to adjacent or overlapping regions. However, their costs and rewards depend on recent field reports of adversarial activity that are not known until the robots are deployed. At this point the robots must already be fueled in order for them to immediately generate and execute a set of *k*-optimal patrols. The resource to be allocated to the robots is the amount of fuel required to execute each patrol; thus it is critical to ensure that enough fuel is given to each robot so that each JA found can be executed, without burdening the robots with wasted fuel that will go unused. Consider another domain involving a team of disaster rescue agents that must generate a set of *k*-optimal JAs in order to present a set of diverse options to a human commander, where each option represents the best JA within a neighborhood of similar JAs. The commander will choose one JA for the team to actually execute. Constraints exist between agents whose actions must be coordinated (i.e. members of subteams) but their costs and rewards depend on conditions on the ground that are unknown until the time when the agents must be deployed. Here, the resource is the time the commander has to make the decision. Presenting too many options will cause the commander to run out of time before considering them all, and presenting too few may cause high-quality options to be omitted.

Because each JA consumes resources, knowing the maximal number of *k*-optimal JAs that could exist for a given DCOP would allow us to allocate sufficient resources for a given level of *k*. Unfortunately, we cannot predict this number because the costs and rewards for the DCOP are not known in advance. Despite this uncertainty, reward-independent bounds can be obtained on the size of a *k*-optimal JA set, i.e. to safely allocate enough resources for a given value of *k* for any DCOP with a particular graph structure. We first identify a mapping to coding theory, yielding bounds independent of both reward and graph structure. We then provide a method to use the structure of the DCOP graph (or hypergraph of arbitrary arity) to obtain significantly tighter bounds.

The third key contribution in this paper is to establish a connection to noncooperative settings by proving that our bounds for 1-optima also apply to the number of pure-strategy Nash equilibria in any graphical game on a given graph, which remains an open problem. In addition to their uses in resource allocation, these bounds provide insight into the problem landscapes that can exist in both cooperative and noncooperative settings.

2. *k*-OPTIMALITY

We introduce the notion of *k-optimality* as a metric that captures both relative quality and diversity when selecting a set of JAs. We begin with our model of the multi-agent team problem, which is a DCOP in which each agent controls a single variable to which it must assign a value. These values correspond to individual actions that can be taken by the agents. Subgroups of agents, whose combined actions generate a cost or reward to the team, define the constraints between agents. Because we assume that each agent controls a single variable, we will use the terms "agent" and "variable" interchangeably. More formally, for a set of agents $\mathcal{I} := \{1, \ldots, I\}$, the ith agent takes action $a_i \in \mathcal{A}_i$. We denote the joint action of a subgroup of agents $S \subset \mathcal{I}$ by $a_S := \times_{i \in S} a_i \in \mathcal{A}_S$ where $\mathcal{A}_S := \bigtimes_{i \in S} \mathcal{A}_i$ and the joint actions (JAs) of the entire multi-agent team by $a = [a_1 \cdots a_I] \in \mathcal{A}$ where $\mathcal{A} := \bigtimes_{i \in \mathcal{I}} \mathcal{A}_i$. The team reward for taking a particular JA, a, is an aggregation of the rewards obtained by subgroups in the team: $R(a) = \sum_{S \in \theta} R_S(a) = \sum_{S \in \theta} R_S(a_S)$ where S is a minimal subgroup that generates a reward (or incurs a cost) in an n-ary DCOP (i.e. a constraint), θ is the collection of all such minimal subgroups for a given problem and $R_S(\cdot)$ denotes a function that maps $\mathcal{A}_S$ to $\mathbb{R}$. By minimality, we mean that the reward component R_S cannot be decomposed further. Mathematically: $\forall S \in \theta, R_S(a_S) \neq R_{S_1}(a_{S_1}) + R_{S_2}(a_{S_2})$ for any $R_{S_1}(\cdot) : \mathcal{A}_{S_1} \to \mathbb{R}, R_{S_2}(\cdot) : \mathcal{A}_{S_2} \to \mathbb{R}, S_1, S_2 \subset \mathcal{I}$ such that $S_1 \cup S_2 = S, S_1, S_2 \neq \emptyset$. It is important to express the constraints minimally to accurately represent dependencies among agents.

To evaluate JA sets, specifically JAs with respect to each other, we need notions of neighborhood and distance among JAs. For two JAs, a and $\tilde{a}$, we define the following terms. The *deviating group* is $D(a, \tilde{a}) := \{i \in \mathcal{I} : a_i \neq \tilde{a}_i\}$, i.e. the set of agents whose actions in JA $\tilde{a}$ differ from their actions in JA a. The *distance* is $d(a, \tilde{a}) := |D(a, \tilde{a})|$ where $|\cdot|$ denotes the cardinality of the set. The *relative reward* of a JA a with respect to another JA $\tilde{a}$ is $\Delta(a, \tilde{a}) := R(a) - R(\tilde{a}) = \sum_{S \in \theta : S \cap D(a,\tilde{a}) \neq \emptyset} [R_S(a_S) - R_S(\tilde{a}_S)]$. In this summation, only the rewards on constraints incident on deviating agents are considered, since the other rewards remain the same. We assume every subgroup of agents G has a unique optimal subgroup joint action a^*_G for any context, where a context consists of a_{G^C}, the actions of the agents not in G, i.e. the complement of G. Mathematically, if $G \subset \mathcal{I}$ where $G \neq \emptyset$ and $G \neq \mathcal{I}$, then $\exists\, a^*_G \in \mathcal{A}_G \; s.t. \; R(a^*_G; a_{G^C}) > R(a_G; a_{G^C}) \forall a_G \neq a^*_G$. Here the notation $R(a_G; a_{G^C})$ is used to indicate the overall team reward generated when subgroup G takes the JA a_G with respect to a fixed context of a_{G^C}. The above assumption is natural for domains where rewards come from precise measurements, and is common in work on bounds and estimates for numbers of local optima [2] and Nash equilibria [11]. Given this assumption, we now classify a as a *k*-

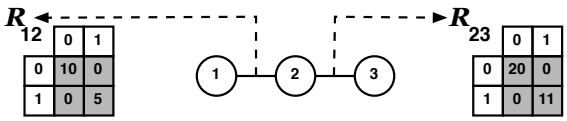


Figure 1: DCOP example

optimal JA or *k-optimum* if $\Delta(a, \tilde{a}) > 0 \ \forall \tilde{a}$ s.t $d(a, \tilde{a}) \leq k$. Equivalently, if the set of agents have reached a k-optimum, then no subgroup of cardinality $\leq k$ can improve the overall reward by choosing different actions; every such subgroup is acting optimally with respect to its context.

A collection of k-optimal JAs must be mutually separated by a distance $\geq k+1$ as they each have the highest reward within a radius of k. Thus, a higher k-optimality of a collection implies a greater level of relative reward and diversity. Let $A_q(I,k) = \{a \in \mathcal{A} : \Delta(a, \tilde{a}) > 0 \ \forall \tilde{a}$ s.t $d(a, \tilde{a}) \leq k\}$ be the set of all k-optimal JAs for a team of I agents with domains of cardinality q. It is straightforward to show $A_q(I, k+1) \subseteq A_q(I, k)$.

EXAMPLE 1. *Figure 1 is a binary DCOP in which agents choose actions from* $\{0, 1\}$*, with rewards shown for the two constraints (minimal subgroups)* $S_1 = \{1, 2\}$ *and* $S_2 = \{2, 3\}$*. The assignment* $a =$ [1 1 1] *is* 1*-optimal because any single agent that deviates reduces the team reward. However,* [1 1 1] *is not* 2*-optimal because if the group* $\{2, 3\}$ *deviated, making the assignment* $\tilde{a} =$ [1 0 0]*, team reward would increase from 16 to 20. The globally optimal solution,* $a^* =$ [0 0 0] *is k-optimal for all* $k \in \{1, 2, 3\}$.□

We now show, in an experiment, the advantages of k-optimal JA sets as capturing both diversity and high reward compared with JA sets chosen by other metrics. The lower half of Figure 2(a) shows a DCOP graph representing a team of 10 patrol robots, each of whom must choose one of two routes to patrol in its region. The nodes are agents and the edges represent binary constraints between agents assigned to overlapping regions. The actions (i.e. the chosen routes) of these agents combine to produce a cost or reward to the team. For each of 20 runs, the edges were initialized with rewards from a uniform random distribution. The set of all 1-optima was enumerated. Then, for the same DCOP, we found equal-sized sets of JAs using two other metrics. In one metric, the set of JAs with highest reward are included, and in the next, JAs were selected purely for diversity by the following method. We repeatedly cycled through all possible JAs in lexicographic order, and included a JA in the set if the distance between it and every JA already in the set was not less than a specified distance; in this case 2. The average reward and the diversity (expressed as the minimum distance between any pair of JAs in the set) for the JA sets chosen using each of the three metrics over all 20 runs is shown in the upper half of Figure 2(a). While the sets of 1-optima come close to the reward level of the sets chosen purely according to reward, they are clearly more diverse (T-tests for this claim showed a significance within .0001%). If a minimum distance of 2 is required in order to guarantee diversity, then using reward alone as a metric is insufficient; in fact the JA sets generated using that metric had an average minimum distance of 1.21, compared with 2.25 for 1-optimal JA sets (which guarantee a minimum distance of $k+1 = 2$). The 1-optimal JA set also provides significantly higher average reward than the sets chosen to maintain a given minimum distance, which had an average reward of 0.037 (T-test significance within .0001%.). Similar results with equal significance were observed for the 10-agent graph in Figure 2(b) and the nine-agent graph in Figure 2(c). Note also that this experiment used $k = 1$, the lowest possible k. Increasing k would, by definition, increase the diversity of the k-optimal JA set as well as the neighborhood size for which each JA is optimal.

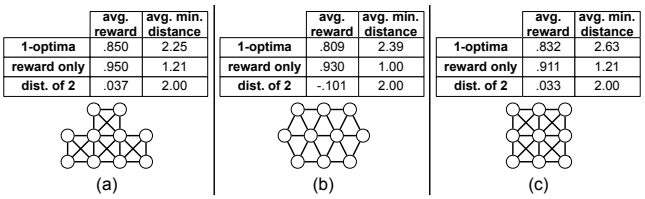


Figure 2: 1-optima vs. JA sets chosen using other metrics

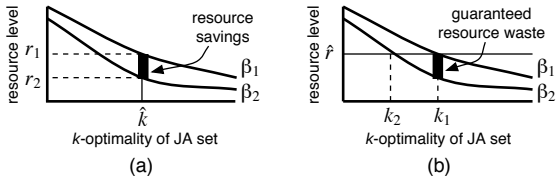


Figure 3: Hypothetical example illustrating the advantages of tighter bounds

In addition to categorizing local optima in a DCOP, k-optimality provides a natural classification for DCOP algorithms. Many known algorithms are guaranteed to converge to k-optima for some $k > 0$, including DBA [20], DSA [5], and coordinate ascent [17] for $k = 1$, MGM-2 and SCA-2 [9] for $k = 2$, and Adopt [12], OptAPO [10] and DPOP [13] for $k = I$. For $k < I$, random restarts of these "k-optimal algorithms" can be used to find sets of k-optimal JAs.

3. UPPER BOUNDS ON k-OPTIMA

Upper bounds on the number of possible k-optimal JAs, $|A_q(I,k)|$ are useful for two reasons: they yield resource savings in domains where a particular level of k-optimality is desired, and help determine the appropriate level of k-optimality to prevent guaranteed waste of resources (fuel, time, etc.) in settings with fixed resources.

First, a particular level of k-optimality may be desired for a JA set: a high k will include JAs that are more diverse, and optimal within a larger radius, but high-k algorithms have significantly higher coordination/communication overheads [12, 10, 9]; hence lower k is preferable under time pressure. Lower k may also be preferable if an agent team or a human user wants a more detailed set of JAs, for example, more joint patrols, more rescue options, etc. For a given level of k-optimality, bounds indicate the maximum resource requirement for any k-optimal JA set. Thus, tighter bounds provide savings by allowing fewer resources to be allocated *a priori* while ensuring enough will be available for all k-optimal JAs, regardless of the rewards and costs on the constraints. Figure 3 is a hypothetical example, with k on the x-axis and the number of resources to be allocated on the y-axis. β_1 and β_2 are two different upper bounds on the number of k-optimal JAs that can exist for a given DCOP. Part (a) shows how the tighter bound β_2 indicates that a resource level of r_2 is sufficient for all $\hat{k}$-optimal JAs, if each JA consumes one resource, yielding a savings of $r_1 - r_2$ over using β_1.

Second, if resource availability is fixed, tighter bounds help us choose an appropriate level of k-optimality. If k is too low, we may exhaust our resources on bad JAs (similar JAs with poor relative quality). In contrast, fewer k-optimal JAs can exist as k increases, and so if k is too high, available resources that could be spent on additional JAs are guaranteed to go unused. Tighter bounds provide a more accurate measure of this kind of guaranteed waste and thus, allow a more appropriate k to be chosen. In Figure 3(b), under fixed resource level $\hat{r}$, the looser bound β_1 hides the resources guaranteed to go unused when k_1 is used. This waste is revealed by β_2, with the thick line indicating the resources that, if allocated, will never be used, as there cannot exist enough k-optima to use them all; instead, we now see that using k_2 will reduce this guaranteed waste.

To find the first upper bounds on the number of k-optima for a given DCOP graph, we discovered a correspondence to coding theory [8]. In error-correcting codes, a set of codewords must be chosen from the space of all possible words, where each word is a string of characters from an alphabet. All codewords are sufficiently different from one another so that transmission errors will not cause one to be confused for another. Finding the maximum possible number of k-optima can be mapped to finding the maximum number of codewords in a space of q^I words where the minimum distance between any two codewords is $d = k + 1$. We can map JAs (complete DCOP assignments) to words and k-optima to codewords as follows: A JA a taken by I agents each with a domain of cardinality q is analogous to a word of length I from an alphabet of cardinality q. The distance $d(a, \tilde{a})$ can then be interpreted as a Hamming distance between two words. Then, if a is k-optimal, and $d(a, \tilde{a}) \leq k$, then $\tilde{a}$ cannot also be k-optimal by definition. Thus, any two k-optima must be separated by distance $\geq k + 1$.

Three well-known bounds [8] on codewords are Hamming: $\beta_H = q^I / \left(\sum_{j=0}^{\lfloor k/2 \rfloor} \binom{I}{j}(q-1)^j\right)$, Singleton: $\beta_S = q^{I-k}$, and Plotkin: $\beta_P = \left\lfloor \frac{k+1}{k+1-(1-q^{-1})I} \right\rfloor$, which is only valid when $(1 - 1/q)n < k + 1$. Note that for the special case of $q = 2$, it is possible to use the relation $\beta_H(I, k, q) = \beta_H(I - 1, k - 1, q)$[8] to obtain a tighter bound for odd k using the Hamming bound. Now, to find a reward-independent bound on the number of 1-optima for three agents with $q = 2$, (e.g., the system in Example 1), we obtain $\min\{\beta_H, \beta_S, \beta_P\} = \beta_H = 4$, without knowing R_{12} and R_{23} explicitly.

Unfortunately, problems of even d (odd k), are not of interest for error-correcting codes, and β_H, the Hamming bound, is very loose or useless for DCOP when $q > 2$, e.g., for 1-optima (solutions reached by DSA) the bound is equal to the number of possible assignments in this case. Hence, for DCOP, we pursue an improved bound for $q > 2$ and odd k. β_H is derived by using a sphere-packing argument stating that the total number of words q^I must be greater than the number of codewords $A_q(I, k)$ multiplied by the size of a sphere of radius $\lfloor k/2 \rfloor$ centered around each codeword. A sphere $S_A(a^*, r)$ with center a^* and radius r is the set of JAs $\tilde{a}$ such that $d(a^*, \tilde{a}) \leq r$, and represents words that cannot be codewords (except for its center). It can be shown that $S_A(a^*, \lfloor k/2 \rfloor)$ contains exactly $\sum_{j=0}^{\lfloor k/2 \rfloor} \binom{I}{j}(q-1)^j$ words. If k is even, the tightest packing occurs with spheres of radius $k/2$ and each word can be uniquely assigned to the sphere of its closest codeword. If k is odd, it is possible for a word to be equidistant from two codewords and it is unclear how to assign it to a sphere. The Hamming bound addresses this issue by using the bound for $k-1$ when k is odd, which leads to smaller spheres and a bound larger than necessary. This ignores the contribution of a word that lies on the "boundary" between several spheres. These boundary assignments can be appropriately partitioned to achieve a tighter bound on the number of k-optima for odd k, called the *Modified Hamming bound*.

PROPOSITION 1. *For odd k, $A_q(I, k) \leq \min\{A_1, A_2\}$ where*

$$A_1 = \frac{q^I - \binom{I}{(k+1)/2}(q-1)^{(k+1)/2}}{\sum_{j=0}^{\lfloor k/2 \rfloor} \binom{I}{j}(q-1)^j}$$

$$A_2 = \frac{q^I}{\sum_{j=0}^{\lfloor k/2 \rfloor} \binom{I}{j}(q-1)^j + \binom{I}{(k+1)/2}(q-1)^{(k+1)/2}(I^{-1})}$$

Proof. Any word that has Hamming distance $\lfloor k/2 \rfloor$ or less from a codeword belongs in that codeword's sphere, because belonging to more than one sphere would violate the code's distance requirement. Given an odd value of k, each codeword will have $\binom{I}{(k+1)/2}(q-1)^{(k+1)/2}$ words that are a distance of $(k + 1)/2$ away from it. It cannot claim all these words for its sphere exclusively, as they may be equidistant from other codewords. We do know however that each of these words can be on the boundary of at most I spheres (i.e. can be equidistant from at most I codewords) because they are of length I. Furthermore, each of these words can be equidistant from at most $A_q(I, k)$ codewords, i.e. the total number of codewords in the space. Thus, each codeword can safely incorporate $1/\min\{I, A_q(I, k)\}$ of each of these boundary words into its sphere without any portion being claimed by more than one sphere. Aggregating over all the words on the boundary, we can increase the volume of the sphere by $\binom{I}{(k+1)/2}(q-1)^{(k+1)/2}/\min\{I, A_q(I, k)\}$. Using the sphere-packing argument with the portions of the boundary words added to each sphere, if $A_q(I, k) \leq I$, we have

$$q^I \geq A_q(I, k)\left[\sum_{j=0}^{\lfloor k/2 \rfloor} \binom{I}{j}(q-1)^j + \frac{\binom{I}{(k+1)/2}(q-1)^{(k+1)/2}}{A_q(I, k)}\right]$$

$$\Rightarrow A_q(I, k) \leq \frac{q^I - \binom{I}{(k+1)/2}(q-1)^{(k+1)/2}}{\sum_{j=0}^{\lfloor k/2 \rfloor} \binom{I}{j}(q-1)^j} \equiv A_1,$$

and if $A_q(I, k) \geq I$, we have

$$q^I \geq A_q(I, k)\left[\sum_{j=0}^{\lfloor k/2 \rfloor} \binom{I}{j}(q-1)^j + \frac{\binom{I}{(k+1)/2}(q-1)^{(k+1)/2}}{I}\right]$$

$$\Rightarrow A_q(I, k) \leq \frac{q^I}{\sum_{j=0}^{\lfloor k/2 \rfloor} \binom{I}{j}(q-1)^j + \binom{I}{(k+1)/2}(q-1)^{(k+1)/2}(I^{-1})} \equiv A_2.$$

We have $A_q(I, k) \leq I \Rightarrow A_q(I, k) \leq A_1$ and $A_q(I, k) \geq I \Rightarrow A_q(I, k) \leq A_2$. We can show that $A_1 \odot I \Leftrightarrow A_2 \odot I$, $\forall \odot \in \{<, >, =\}$. Furthermore, $A_1 \odot I, A_2 \odot I \Leftrightarrow A_1 \odot A_2$. Thus, when $A_1 \leq I$, then $A_2 \leq I$ and $A_1 \leq A_2$. So, $A_q(I, k) \leq A_1 = \min\{A_1, A_2\}$ when $A_1 \leq I$. And, when $A_1 > I$, then $A_2 > I$ and $A_1 > A_2$. So, $A_q(I, k) \leq A_2 = \min\{A_1, A_2\}$ when $A_1 > I$. Therefore, $A_q(I, k) \leq \min\{A_1, A_2\}$.■

We call the Modified Hamming bound β_{MH} and define $\beta_{HSP} = \min\{\beta_H, \beta_S, \beta_P, \beta_{MH}\}$, including the relation for β_H for $q = 2$; i.e. β_{HSP} gives the best of all the (graph-independent) bounds so far.

4. GRAPH-BASED ANALYSIS: *k*-OPTIMA

The β_{HSP} bound and its components depend only on I, k and q, regardless of how the team reward is decomposed onto constraints; i.e., the bounds are the same for all θ. For instance, the bound on 1-optima for Example 1 (found to be 4 in the previous section) ignored the fact that agents 1 and 3 do not share a constraint, and yields the same result independent of the DCOP graph structure. However, exploiting this structure (as captured by θ) can significantly tighten the bounds on $\{|A_q(I, k)|\}_{k=1}^{I}$. In particular, in obtaining the bounds in Section 3, pairs of JAs were mutually exclusive as k-optima (only one of the two could be k-optimal) if they were separated by a distance $\leq k$. We now show how some JAs separated by a distance $\geq k + 1$ must also be mutually exclusive as k-optima.

We define $D_G(a, \tilde{a}) := \{i \in G : a_i \neq \tilde{a}_i\}$ and $V(G) := \cup_{S \in \theta : G \cap S \neq \emptyset} S$. Intuitively, $D_G(a, \tilde{a})$ is the set of agents within the subgroup G who have chosen different actions between a and $\tilde{a}$, and $V(G)$ is the set of agents (including those in G) who are a member of some constraint $S \in \theta$ incident on a member of G (e.g., G and the agents who share a constraint with some member of G). Then, $V(G)^C$ is the set of all agents whose contribution to the team reward is independent of the values taken by G.

PROPOSITION 2. *Let there be a JA $a^* \in A_q(I, k)$ and let $\tilde{a} \in A$ be another JA for which $d(a^*, \tilde{a}) > k$. If $\exists\, G \subset \mathcal{I}$, $G \neq \emptyset$ for which $|G| \leq k$ and $D_{V(G)}(a^*, \tilde{a}) = G$, then $\tilde{a} \notin A_q(I, k)$.*

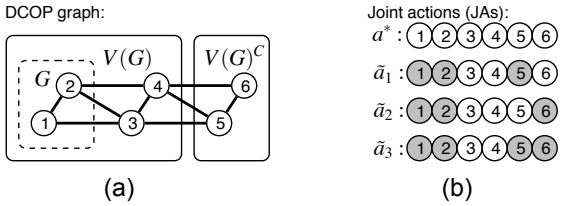


Figure 4: A visual representation of the effect of Proposition 2.

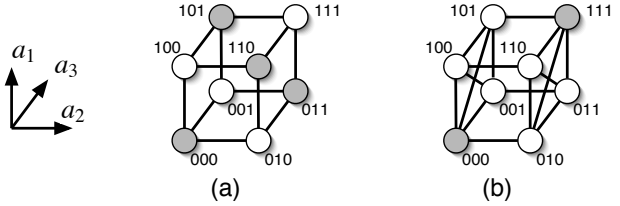


Figure 5: Exclusivity graphs for 1-optima for Example 1 with MIS shown in gray, (a) not using Proposition 2 and (b) using it.

Proof. Given a^*, $\tilde{a}$, and G with the properties stated above, we have that $\forall a : d(a^*, a) \leq k$, $\Delta(a^*, a) > 0$. If a is defined such that $a_i = \tilde{a}_i$ for $i \in V(G)$ and $a_i = a_i^*$ for $i \notin V(G)$, then $D(a^*, a) = G$ and $d(a^*, a) \leq k$ which implies

$$\Delta(a^*, a) = \sum_{S \in \theta : S \cap D(a^*, a) \neq \emptyset} R_S(a_S^*) - R_S(a_S) = \sum_{S \in \theta : S \cap G \neq \emptyset} R_S(a_S^*) - R_S(a_S)$$
$$= \sum_{S \in \theta : S \cap G \neq \emptyset} R_S(a_S^*) - R_S(\tilde{a}_S) > 0.$$

If $\hat{a}$ is defined such that $\hat{a}_i = a_i^*$ for $i \in V(G)$ and $\hat{a}_i = \tilde{a}_i$ for $i \notin V(G)$, then $D(\tilde{a}, \hat{a}) = G$ and $d(\tilde{a}, \hat{a}) \leq k$, and

$$\Delta(\tilde{a}, \hat{a}) = \sum_{S \in \theta : S \cap D(\tilde{a}, \hat{a}) \neq \emptyset} R_S(\tilde{a}_S) - R_S(\hat{a}_S) = \sum_{S \in \theta : S \cap G \neq \emptyset} R_S(\tilde{a}_S) - R_S(\hat{a}_S)$$
$$= \sum_{S \in \theta : S \cap G \neq \emptyset} R_S(\tilde{a}_S) - R_S(a_S^*) < 0,$$

thus, $\tilde{a} \notin A_q(I, k)$ because $\Delta(\tilde{a}, \hat{a}) < 0$ and $d(\tilde{a}, \hat{a}) \leq k$.■

Proposition 2 provides conditions where if a^* is k-optimal then $\tilde{a}$, which may be separated from a^* by a distance greater than $k+1$ may not be k-optimal, thus tightening bounds on k-optimal JA sets. With Proposition 2, since agents are typically not fully connected to all other agents, the *relevant context* a subgroup faces is not the entire set of other agents. Thus, the subgroup and its relevant context form a view (captured by $V(G)$) that is not the entire team. We consider the case where a JA $\tilde{a}$ has $d(a^*, \tilde{a}) > k$. We also have group G of size k within whose view $V(G)$, G are the only deviators between a^* and $\tilde{a}$ (although agents outside the view must also have deviated, because $d(a^*, \tilde{a}) > k$). We then know that $\tilde{a}$ contains a group G of size k or less that has taken a suboptimal subgroup joint action with respect to its relevant context and thus $\tilde{a}$ cannot be k-optimal, i.e. if the group G chose a_G^* instead of $\tilde{a}_G$ under its relevant context $V(G) \setminus G$ for $\tilde{a}$, then team reward would increase.

Figure 4(a) shows G, $V(G)$, and $V(G)^C$ for a sample DCOP of six agents with a domain of two actions, white and gray. Without Proposition 2, $\tilde{a}_1$, $\tilde{a}_2$, and $\tilde{a}_3$ could all potentially be 2-optimal. However, Proposition 2 guarantees that they are not, leading to a tighter bound on the number of 2-optima that could exist. To see the effect, note that if a^* is 2-optimal, then $G = \{1, 2\}$, a subgroup of size 2, must have taken an optimal subgroup joint action (all white) given its relevant context (all white). Even though $\tilde{a}_1$, $\tilde{a}_2$, and $\tilde{a}_3$ are a distance greater than 2 from a^*, they cannot be 2-optimal, since in each of them, G faces the same relevant context (all white) but is now taking a suboptimal subgroup joint action (all gray).

To explain the significance of Proposition 2 to bounds, we introduce the notion of an *exclusivity relation* $E \subset \mathcal{I}$ which captures the restriction that if deviating group $D(a, \tilde{a}) = E$, then at most one of a and $\tilde{a}$ can be k-optimal. An *exclusivity relation set* for k-optimality, $\mathcal{E}_k \subset \mathcal{P}(\mathcal{I})$, is a collection of such relations that limits $|A_q(I, k)|$, the number of JAs that can be k-optimal in a reward-independent setting (otherwise every JA could potentially be k-optimal). In particular, the set $\mathcal{E}_k$ defines an *exclusivity graph* H_k where each node corresponds uniquely to one of all q^I possible JAs. Edges are defined between pairs of JAs, a and $\tilde{a}$, if $D(a, \tilde{a}) \in \mathcal{E}_k$. The size of the maximum independent set (MIS) of H_k, the largest subset of nodes such that no pair defines an edge, gives an upper bound on $|A_q(I, k)|$. Naturally, an expanded $\mathcal{E}_k$ implies a more connected exclusivity graph and thus a tighter bound on $|A_q(I, k)|$.

Without introducing graph-based analysis, β_{HSP} for each k provides a bound on the MIS of H_k when $\mathcal{E}_k = \bigcup_{E \subset \mathcal{I} : 1 \leq |E| \leq k} E$. This set $\mathcal{E}_k$ captures only the restriction that no two JAs within a distance of k can both be k-optimal. Consider Example 1, but with unknown rewards on the links. Here, the exclusivity relation set for 1-optima without considering the DCOP graph is $\mathcal{E}_1 = \{\{1\}, \{2\}, \{3\}\}$, meaning that no two JAs differing only by the action taken by either agent 1, 2, or 3, can both be 1-optimal. This leads to the exclusivity graph in Figure 5(a) whose MIS implies a bound of 4.

The significance of Proposition 2 is that it provides additional exclusivity relations for solutions separated by distance $\geq k+1$, which arise only because we considered the structure of the DCOP graph, which will allow a tighter bound to be computed. This graph-based exclusivity relation set is $\widetilde{\mathcal{E}}_k = \bigcup_{E \subset \mathcal{I} : 1 \leq |E| \leq k} \bigcup_{F \in \mathcal{P}(V(E)^C)} [E \cup F]$ which is a superset of $\mathcal{E}_k$. Additional relations exist because multiple exclusivity relations ($\bigcup_{F \in \mathcal{P}(V(E)^C)} [E \cup F]$) appear the same to the subgroup E because of its reduced view $V(E)$. Now, for Example 1, the exclusivity relation set for 1-optima when considering the DCOP graph is $\widetilde{\mathcal{E}}_1 = \{\{1\}\{2\}, \{3\}, \{1, 3\}\}$, which now has the additional relation {1,3}. This relation, included because of the realization that agents 1 and 3 are not connected, says that no two JAs can both be 1-optimal if they differ only in the actions of both agent 1 and agent 3. This leads to the exclusivity graph in Figure 5(b) whose MIS implies a bound of 2. Algorithms for obtaining bounds using $\widetilde{\mathcal{E}}_k$ will be discussed in Section 6.

5. APPLICATION TO NASH EQUILIBRIA

Our graph-based bounds can be extended beyond agent teams to noncooperative settings. It is possible to employ the same exclusivity relations for 1-optimal DCOP assignments to bound the number of pure-strategy Nash equilibria in a graphical game (of the same graph structure) using any of our bounds for $|A_q(I, 1)|$. Bounds on Nash equilibria [11] are useful for design and analysis of mechanisms as they predict the maximum number of outcomes of a game.

We begin with a set of noncooperative agents $\mathcal{I} = \{1, \ldots I\}$, where the ith agent's utility is $U^i(a_i; a_{\{\mathcal{I} \setminus i\}}) = \sum_{S_i \in \theta_i} U^i_{S_i}(a_i; a_{\{S_i \setminus i\}})$ which is a decomposition into an aggregation of component utilities generated from minimal subgroups. Note that the combination of actions taken by any subgroup of agents may generate utility for any agent i, therefore the subgroups are denoted as S_i rather than S, as in the cooperative case, where the utility went to the entire team. The notation a_i and $a_{\{G \setminus i\}}$ refers to the ith agent's action and the actions of the group G with i removed, respectively. We refer to a as a joint action (JA), with the understanding that it is composed of actions motivated by individual utilities. Let the *view* of the ith agent in a noncooperative setting to be $V(i) = \cup_{S_i \in \theta_i} S_i$. The deviating group with respect to G is: $D_G(a, \tilde{a}) := \{i \in G : a_i \neq \tilde{a}_i\}$. Assuming every player has a unique optimal response to its context, then if a^* is a pure-strategy Nash equilibrium, and $d(a^*, a) = 1$,

$i = D(a^*, a)$, we know that $U^i(a_i^*; a^*_{\{\mathcal{I}\setminus i\}}) > U^i(a_i; a^*_{\{\mathcal{I}\setminus i\}})$ and a is not a pure-strategy Nash equilibrium. However, applying the graph (or hypergraph) structure of the game, captured by the sets $\{\theta_i\}$, we get exclusivity relations between JAs with distance > 1 as follows.

PROPOSITION 3. *If a^* is a pure-strategy Nash equilibrium, $\tilde{a} \in A$ such that $d(a^*, \tilde{a}) > 1$, and $\exists i \in \mathcal{I}$ such that $D_{V(i)}(a^*, \tilde{a}) = i$, then $\tilde{a}$ is not a pure-strategy Nash equilibrium.*

Proof. We have $U^i(\tilde{a}_i; \tilde{a}_{\{\mathcal{I}\setminus i\}})$

$$= \sum_{S_i \in \theta_i} U^i_{S_i}(\tilde{a}_i; \tilde{a}_{\{S_i \setminus i\}}) = \sum_{S_i \in \theta_i} U^i_{S_i}(\tilde{a}_i; a^*_{\{S_i \setminus i\}})$$
$$< \sum_{S_i \in \theta_i} U^i_{S_i}(a_i^*; a^*_{\{S_i \setminus i\}}) = \sum_{S_i \in \theta_i} U^i_{S_i}(a_i^*; \tilde{a}_{\{S_i \setminus i\}}) = U^i(a_i^*; \tilde{a}_{\{\mathcal{I}\setminus i\}}).$$

The first and last equalities are by definition. The second and third equalities are because $D_{V(i)}(a^*, \tilde{a}) = i$. The inequality is because a^* is a pure-strategy Nash equilibrium. The result is that $\tilde{a}_i$ is not an optimal response to $\tilde{a}_{\{\mathcal{I}\setminus i\}}$ and thus cannot be a Nash equilibrium. ■

Proposition 3 states that a^* and $\tilde{a}$ cannot both be Nash equilibria if $\exists i$, $D_{V(i)}(a^*, \tilde{a}) = i$, which is identical to the condition that prevents two JAs (in a team setting) from being 1-optimal. The commonality is that in both the cooperative and noncooperative settings, agents have optimal actions for any given context, and in both settings there is a notion of relevant context, $V(i) \setminus i$, which can be a subset of other agents $\{\mathcal{I} \setminus i\}$. The difference is that the views are generated in different manners: $V(i) = \cup_{S \in \theta : i \cap S \neq \emptyset} S$ in a cooperative setting, while $V(i) = \cup_{S_i \in \theta_i} S_i$ in a noncooperative setting. Given the views, we can generate the exclusivity relation set in the same manner, $\mathcal{E}_1 = \bigcup_{i \in \mathcal{I}} \bigcup_{F \in \mathcal{P}(V(i)^C)} [i \cup F]$. Given the exclusivity relation set, we can create an exclusivity graph for a noncooperative setting in a fashion similar to the one in Section 4. Thus, the bound on the number of Nash equilibria for a noncooperative graphical game is identical to the bound on 1-optimal JAs for a cooperative DCOP, if both share the same exclusivity relation set $\mathcal{E}_1$.

6. GRAPH-BASED BOUNDS

As seen earlier, the graph structure expands the exclusivity relation set for k-optimality in cooperative (DCOP) settings and Nash equilibria in noncooperative (graphical-game) settings. This set defines exclusivity graph H_k whose maximum independent set (MIS) provides a bound for the number of k-optimal JAs (or alternatively, for the number of Nash equilibria). Finding the size of the MIS is NP-complete in the general case [1], so we investigated heuristic techniques to obtain an upper bound on $|A_q(I,k)|$. We observe that any fully-connected subset (clique) of H_k can contain at most one k-optimum. Thus, the number of cliques in any clique partitioning of H_k also provides an upper bound on $|A_q(I,k)|$, where a partitioning yielding fewer cliques will provide a tighter bound. Hence, our first approach is the polynomial-time F_{CLIQUE} clique-partitioning algorithm, shown in [7] to outperform several competitors.

Our second heuristic technique to find a graph-based bound is Algorithm 1, the *Symmetric Region Packing bound*, β_{SRP}, which uses a packing method analogous to Proposition 1, where each k-optimum claims a region of the space of all possible JAs (the nodes of H_k). Because these regions are constructed to be disjoint and have identical volumes, dividing the space of all JAs by this volume yields a bound. Figure 6 shows β_{SRP} computed for 1-optima for Example 1. We choose an arbitrary JA $a \in A$ which we assume to be k-optimal (a = [0 0 0] in Figure 6), around which we will construct a region claimed by a.

Applying the exclusivity relations from $\widetilde{\mathcal{E}}_k$, we generate a set $B(a) = \cup_{E \in \widetilde{\mathcal{E}}_k} f(a, E)$ where $f(a, E)$ yields the JA that is excluded

Algorithm 1 for Symmetric Region Packing (SRP) bound

```
1: Ẽ_k = ⋃_{E⊂I : 1≤|E|≤k} ⋃_{F∈P(V(E)^C)} [E ∪ F]
2: a = [0 0 0]
3: |A_k| = 1
4: B(a) = ∪_{E∈Ẽ_k} f(a, E)
5: for all b ∈ B(a) do
6:     B̄(b) = (∪_{E∈Ẽ_k} f(b, E)) \ (a ∪ B(a))
7:     H̄_k(b).addNodes(B̄(b))
8:     for all b̄_1, b̄_2 ∈ B̄(b) do
9:         if D(b̄_1, b̄_2) ∈ Ẽ_k then
10:            H̄_k(b).addEdge(b̄_1, b̄_2)
11:    M_b = |cliquePartition(H̄_k(b))|
12:    |A_k| = |A_k| + 1/(1 + M_b)
13: β_SRP = (q^I)/|A_k|
```

$\widetilde{\mathcal{E}}_1 = \{$	$\{1\}$,	$\{2\}$,	$\{3\}$,	$\{1,3\}$ $\}$
$B([0\,0\,0]) = \{$	[1 0 0],	[0 1 0],	[0 0 1],	[1 0 1] $\}$
$\widetilde{\mathcal{E}}_1 = \{$	$\overline{B}([1\,0\,0])$	$\overline{B}([0\,1\,0])$	$\overline{B}([0\,0\,1])$	$\overline{B}([1\,0\,1])$
$\{1\}$,	[0 0 0]	[1 1 0]	[1 0 1]	[0 0 1]
$\{2\}$,	[1 1 0]	[0 0 0]	[0 1 1]	[1 1 1]
$\{3\}$,	[1 0 1]	[0 1 1]	[0 0 0]	[1 0 0]
$\{1,3\}\}$	[0 0 1]	[1 1 1]	[1 0 0]	[1 1 1]
$\overline{H}_1(b)$ (exclusivity subgraph)	110	110, {1,3}, {3}, 011, 111, {1}	011	111
$M_b =$	1	1	1	1
$1/(1+M_b) =$	1/2	1/2	1/2	1/2

Figure 6: Computation of β_{SRP} for Example 1

from being k-optimal by a and E. The first two rows of Figure 6 show $\widetilde{\mathcal{E}}_1$ and the set B([0 0 0]). Applying the exclusivity relations again for each $b \in B(a)$, and discarding JAs already included in a or $B(a)$, we generate a set $\overline{B}(b) = \cup_{E \in \widetilde{\mathcal{E}}_k} f(b, E)$ which contains all JAs that potentially exclude b from being k-optimal. In Figure 6, we apply $\widetilde{\mathcal{E}}_1$ to find $\overline{B}(b)$ for all $b \in B(a) = \{[1\,0\,0], [0\,1\,0], [0\,0\,1], [1\,0\,1]\}$ where the grayed out JAs are those discarded for being in $\{a\} \cup B(a)$. To ensure that the region that a claims is disjoint from the regions claimed by other k-optima, a should only claim a fraction of each $b \in B(a)$. This can be achieved if a shares each b equally with all other k-optima that might exclude b. These additional k-optima are contained within $\overline{B}(b)$. However, not all $\overline{b} \in \overline{B}(b)$ can actually be k-optimal as they might exclude each other. If we construct a graph $\overline{H}_k(b)$ with nodes for all $\overline{b} \in \overline{B}(b)$ and edges formed using $\widetilde{\mathcal{E}}_k$, and we find M_b, the size of the MIS, then a can safely claim $1/(1 + M_b)$ of b. We again use clique partitioning to safely estimate M_b. In Figure 6, for b = [0 1 0], $\overline{B}$([0 1 0]) leads to a three-node, three-edge exclusivity graph $\overline{H}_k$([0 1 0]). By adding the values of $1/(1 + M_b)$ for all $b \in B(a)$ (plus one for itself), we obtain that a can safely claim a region of size 3, which implies $\beta_{SRP} = \lfloor 2^3/3 \rfloor = 2$. Algorithm 1's runtime is polynomial in the number of possible JAs, which is a comparatively small cost for a bound that applies to every possible instantiation of rewards to actions. An exhaustive search for the MIS of H_k would be exponential in this number (doubly exponential in the number of agents).

7. EXPERIMENTAL RESULTS

We performed five evaluations in addition to the experiment described in Section 2. The first evaluates the impact of k-optimality for higher values of k. For each of the three DCOP graphs from Figure 2(a-c), Figure 7(a-c) shows key properties for 1-, 2- and 3-optima. The first column of each table shows $|\tilde{A}|$, the size of the

	$\lvert\tilde{A}\rvert$	avg. reward
1-opt.	10	.850
2-opt.	55	.964
3-opt.	175	.993

(a)

	$\lvert\tilde{A}\rvert$	avg. reward
1-opt.	10	.809
2-opt.	55	.961
3-opt.	175	.986

(b)

	$\lvert\tilde{A}\rvert$	avg. reward
1-opt.	9	.832
2-opt.	45	.977
3-opt.	129	.982

(c)

Figure 7: 1-optima vs. JA sets chosen using other metrics

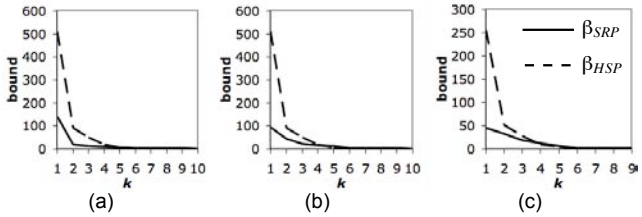


Figure 8: β_{SRP} vs. β_{HSP} for DCOP graphs from Figure 2

neighborhood containing all JAs within a distance of k from a k-optimal JA a, and hence of lower reward than a. For example, in the joint patrol domain described in Section 2, Figure 7(a) shows that, if agents are arranged as in the DCOP graph from Figure 2 (a), any 1-optimal joint patrol must have a higher reward than at least 10 other joint patrols. We see that as k increases, the k-optimal set contains JAs that each individually dominate a larger and larger neighborhood. The second column shows, for each of the three graphs, the average reward of each k-optimal JA set found over 20 problem instances, generated by assigning rewards to the links from a uniform random distribution. We define the reward of a k-optimal JA set as the mean reward of all k-optimal JAs that exist for a particular problem instance; each figure in the second column is therefore a mean of means. As k was increased, leading to a larger neighborhood of dominated JAs, the average reward of the k-optimal JA sets show a significant increase (T-tests showed the increase in average reward as k increased was significant within 5%.)

However as k increases, the number of possible k-optimal JAs decreases, and hence the next four evaluations explore the effectiveness of the different bounds on the number of k-optima. For the three DCOP graphs shown in Figure 2, Figure 8 provides a concrete demonstration of the gains in resource allocation due to the tighter bounds made possible with graph-based analysis. The x axis in Figure 8 shows k, and the y axis shows the β_{HSP} and β_{SRP} bounds on the number of k-optima that can exist. To understand the implications of these results on resource allocation, consider a patrolling problem where the constraints between agents are shown in the 10-agent DCOP graph from Figure 2(a), and all agents consume one unit of fuel for each JA taken. Suppose that $k = 2$ has been chosen, and so at runtime, the agents will use MGM-2 [9], repeatedly, to find and execute a set of 2-optimal JAs. We must allocate enough fuel to the agents *a priori* so they can execute up to all possible 2-optimal JAs. Figure 8(a) shows that if β_{HSP} is used, the agents would be loaded with 93 units of fuel to ensure enough for all 2-optimal JAs. However, β_{SRP} reveals that only 18 units of fuel are sufficient, a five-fold savings. (For clarity we note that on all three graphs, both bounds are 1 when $k = I$ and 2 when $I - 3 \leq k < I$.)

To systematically investigate the impact of graph structure on bounds, we generated a large number of DCOP graphs of varying size and density. We started with complete binary graphs (all pairs of agents are connected) where each node (agent) had a unique ID. To gradually make each graph sparser, edges were repeatedly removed according to the following two-step process: (1) Find the lowest-ID node that has more than one incident edge. (2) If such a node exists, find the lowest-ID node that shares an edge with it, and remove this edge. Figure 9 shows the β_{HSP} and β_{SRP} bounds

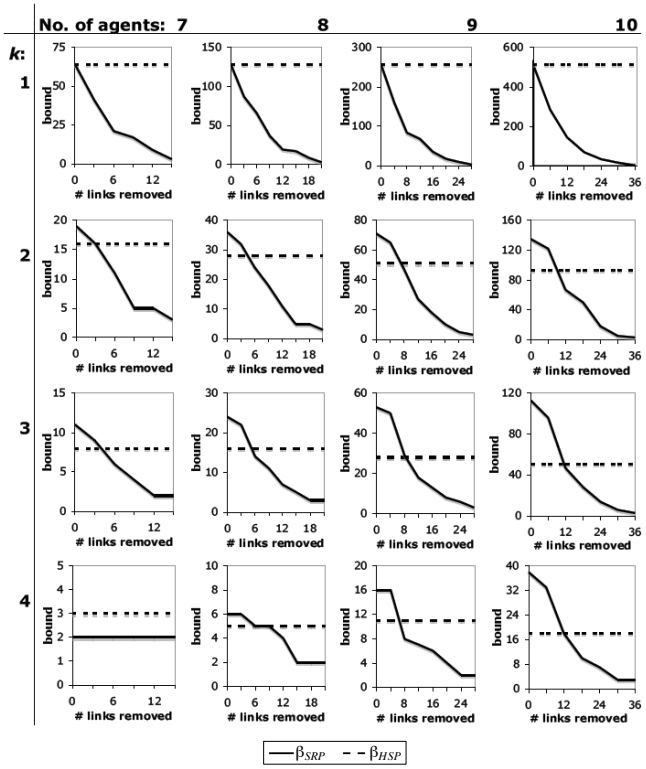


Figure 9: Comparisons of β_{SRP} vs. β_{HSP}

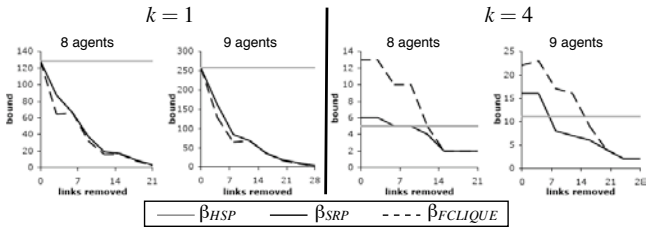


Figure 10: Comparisons of β_{SRP}, β_{HSP}, $\beta_{FCLIQUE}$

for k-optima for $k \in \{1, 2, 3, 4\}$ and $I \in \{7, 8, 9, 10\}$. For each of the 16 plots shown, the y axis shows the bounds and the x-axis shows the number of links removed from the graph according to the above method. While $\beta_{HSP} < \beta_{SRP}$ for very dense graphs, β_{SRP} provides significant gains for the vast majority of cases. For example, for the graph with 10 agents, and 24 links removed, and a fixed $k = 1$, β_{HSP} implies that we must equip the agents with 512 resources to ensure that all resources are not exhausted before all 1-optimal actions are executed. However, β_{SRP} indicates that a 15-fold reduction to 34 resources will suffice, yielding a savings of 478 due to the use of graph structure when computing bounds.

A fourth experiment compared β_{HSP} and β_{SRP} to the bound obtained by applying F_{CLIQUE}, $\beta_{FCLIQUE}$ to DCOP graphs from the previous experiment. Selected results are shown in Figure 10 for graphs of 8 and 9 agents. While $\beta_{FCLIQUE}$ is marginally better for $k = 1$, β_{SRP} has clear gains for $k = 4$. Identifying the relative effectiveness of various algorithms that exploit our exclusivity relation sets is clearly an area for future work.

Finally, Figure 11 compares the constant-time-computable graph-independent bounds from Section 3, in particular, showing the improvement of β_{MH} over $\min\{\beta_H, \beta_S, \beta_P\}$ for selected odd values of k, given three possible actions for each agent ($q = 3$). The x-axis shows I, the number of agents and the y-axis show s $100 \cdot (\min\{\beta_H, \beta_S, \beta_P\} - \beta_{MH}) / \min\{\beta_H, \beta_S, \beta_P\}$. For odd values of $k > 1$,

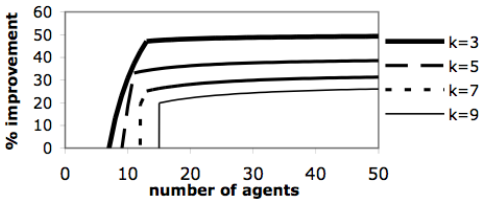


Figure 11: Improvement of β_{MH} on $\min\{\beta_H, \beta_S, \beta_P\}$

as I increased, β_{MH} provided a tighter bound on the number of k-optima. The most improvement was for $k = 3$; as I increased, β_{MH} gave a bound 50% tighter than the others.

8. RELATED WORK AND CONCLUSION

This paper provides a theoretical complement to the experimental analysis of local minima (1-optima) and landscapes in centralized constraint satisfaction problems (CSPs) [19] as well as incomplete DCOP algorithms [20, 9]. In contrast, we provide categorization and theoretical justification for k-optimality, bounds on the number of k-optima (both graph-independent and dependent), algorithms to compute graph-based bounds, and experimental analysis. We note that k-optimality can also apply to centralized constraint reasoning as a measure of the relative quality and diversity of solutions in a set. However, examining properties of solutions that arise from coordinated value changes of small groups of variables is especially useful in distributed settings, given the computational and communication expense of large-scale coordination.

Our research on bounding k-optimal solution sets for DCOPs is related to estimating numbers of local optima in centralized local search and evolutionary computing [2, 18]. The key difference is in the exploitation of constraint graph structure, not harnessed in previous work, to bound the number of optima.

Given that counting the number of Nash equilibria in a game with known payoffs is $\#\mathcal{P}$-hard [3], bounds have been investigated for particular types of games [11]. Graph structure is utilized in algorithms to expedite finding Nash equilibria for a given graphical game with known payoffs [6]. However, finding tight bounds on Nash equilibria over all possible games on a given graph (i.e., reward-independent bounds) remained an open problem.

In summary, in this paper, (1) we have introduced k-optimality as a metric that captures diversity and relative quality, properties desirable for evaluating sets of DCOP assignments, where each assignment represents a JA to be considered or executed. Finding bounds on k-optimal JA sets is useful for resource allocation problems associated with executing JA sets in sequence or presenting a JA set as a set of options. (2) We discover a correspondence to coding theory that yields bounds β_H, β_S, and β_P, independent of reward and graph structure and (3) introduce a tighter bound β_{MH} for odd k, all of which are computable in constant time. We introduced (4) a method to exploit DCOP graph structure to obtain tighter reward-independent bounds on the number of k-optima that can exist. (5) We also show that our method extends to noncooperative settings, as our bound for 1-optima in a DCOP can be used as a bound on the number of pure-strategy Nash equilibria in a graphical game of arbitary payoffs. Finally, (6) we develop techniques for computing bounds ($\beta_{SRP}, \beta_{FCLIQUE}$) using the graph-based exclusivity relation sets and (7) illustrate their utility on a diverse collection of graphs.

9. ACKNOWLEDGMENTS

Thanks to Gal Kaminka, Pragnesh Jay Modi, Praveen Paruchuri, and Paul Scerri for their helpful comments. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010.

10. REFERENCES

[1] N. Alon and N. Kahale. Approximating the independence number via the theta-function. *Mathematical Programming*, 80:253–264, 1998.

[2] R. Caruana and M. Mullin. Estimating the number of local minima in complex search spaces. In *IJCAI Workshop on Optimization*, 1999.

[3] V. Conitzer and T. Sandholm. Complexity results about Nash equilibria. In *IJCAI*, 2003.

[4] J. Cox, E. Durfee, and T. Bartold. A distributed framework for solving the multiagent plan coordination problem. In *AAMAS*, 2005.

[5] S. Fitzpatrick and L. Meertens. Distributed coordination through anarchic optimization. In V. Lesser, C. L. Ortiz, and M. Tambe, editors, *Distributed Sensor Networks: A Multiagent Perspective*, pages 257–295. Kluwer, 2003.

[6] M. Kearns, M. Littman, and S. Singh. Graphical models for game theory. In *Proc. UAI*, 2001.

[7] J. T. Kim and D. R. Shin. New efficient clique partitioning algorithms for register-transfer synthesis of data paths. *Journal of the Korean Phys. Soc.*, 40(4):754–758, 2002.

[8] S. Ling and C. Xing. *Coding Theory: A First Course*. Cambridge U. Press, 2004.

[9] R. T. Maheswaran, J. P. Pearce, and M. Tambe. Distributed algorithms for DCOP: A graphical-game-based approach. In *PDCS*, 2004.

[10] R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *AAMAS*, 2004.

[11] A. McLennan and I. Park. Generic 4 x 4 two person games have at most 15 Nash equilibria. *Games and Economic Behavior*, 26(1):111–130, 1999.

[12] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.

[13] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, 2005.

[14] S. Ruan, C. Meirina, F. Yu, K. R. Pattipati, and R. L. Popp. Patrolling in a stochastic environment. In *10th Intl. Command and Control Research Symp.*, 2005.

[15] N. Schurr, J. Marecki, P. Scerri, J. Lewis, and M. Tambe. The DEFACTO system: Training tool for incident commanders. In *IAAI*, 2005.

[16] A. Tate, J. Dalton, and J. Levine. Generation of multiple qualitatively different plan options. In *Proc. AIPS*, 1998.

[17] N. Vlassis, R. Elhorst, and J. R. Kok. Anytime algorithms for multiagent decision making using coordination graphs. In *Proc. Intl. Conf. on Systems, Man and Cybernetics*, 2004.

[18] D. Whitley, S. Rana, and R. B. Heckendorn. Representation issues in neighborhood search and evolutionary algorithms. In D. Quagliarella, et al., editor, *Genetic Algs. and Evolution Strategies in Eng. and Comp. Sci.*, pages 39–57. Wiley, 1998.

[19] M. Yokoo. How adding more constraints makes a problem easier for hill-climbing algorithms: Analyzing landscapes of CSPs. In *Int'l Conf. on Constraint Programming*, 1997.

[20] W. Zhang, Z. Xing, G. Wang, and L. Wittenburg. An analysis and application of distributed constraint satisfaction and optimization algorithms in sensor networks. In *AAMAS*, 2003.

Towards Autonomous Mobile Agents with Emergent Migration Behaviour

Tino Schlegel*
Swinburne University of Technology
Faculty of Information and Communication Technologies
Hawthorn, 3122 Victoria, Australia
tschlegel@ict.swin.edu.au

Peter Braun
Swinburne University of Technology
Faculty of Information and Communication Technologies
Hawthorn, 3122 Victoria, Australia
pbraun@ict.swin.edu.au

Ryszard Kowalczyk
Swinburne University of Technology
Faculty of Information and Communication Technologies
Hawthorn, 3122 Victoria, Australia
rkowalczyk@ict.swin.edu.au

ABSTRACT

Along with manifold advantages of distributed multi-agent systems, increased network traffic produced by highly communicative agents in large distributed systems has to be considered as their practical downside. We suggest to apply the programming paradigm of mobile agents to reduce this network overhead by allowing agents to meet at the same network node before commencing communication. A remote communication between two agents could then be replaced by one or two agent migrations, followed by local communication. Since only in trivial cases it is possible to decide at design time whether remote communication or agent migration with subsequent local communication would perform better, this decision has to be made at run-time based on environmental parameters and agents' past experience. We present an adaptive approach, which is inspired by a solution of the *El Farol* problem. Every agent forecasts the network load of the next communication step and applies a simple mathematical model to decide between the two alternatives at run-time. In addition, our approach does not only consider network load but also server load by enabling agents to dynamically forecast the number of agents migrating to a specific agent server. The approach is evaluated with simulation experiments in static and dynamic server load environments.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent agents

General Terms

Performance

*PhD student.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Keywords

mobile agents, collective and emergent agent behaviour

1. INTRODUCTION

In most distributed multi-agent systems, communication and coordination between agents is based on asynchronous message passing. If agents are located on distant agent servers, sending messages between agents generates network traffic between the corresponding network nodes. As every network traffic causes higher execution time by network transmission and latency compared with a centralized solution, a trade-off between advantages of agent distribution and drawbacks regarding their execution time has to be considered. For example, a promising application area of multi-agent systems is the management of Grid services [8, 13, 14]. Agents can act as representative of a service instance and supplement the service's functionality by intelligent techniques for negotiating QoS parameters [12]. Multi-step negotiation based on the iterative contract-net protocol between widely distributed agents can impose high network overhead, which might question the applicability of negotiations at large. We believe that this important problem has not sufficiently been recognized by the agent community so far and current agent development methodologies do not provide solutions to this problem.

Mobile agents have been introduced as a paradigm for distributed systems, in particular to reduce network traffic by moving code close to the place where the data is. A mobile agent is a program that can migrate from one host to another host in a network of heterogeneous computer systems and fulfil a user-given task. It can autonomously work and communicate with other agents and host systems. During the self-initiated migration, the agent carries its code and some kind of execution state with it [4, 21].

Mobile agents are a solution of our scenario of negotiating agents, because both agents could decide to meet at the same agent server and then communicate only locally without generating any network traffic except of the migration. Mobile agents can not only reduce network load and increase application response time but can also improve reliability by making the application more robust against network failures and reduce power consumption in case of mobile devices. In the last years, many research groups have carried out empirical evaluations to evaluate which of these two paradigms

(remote communication or migration followed by local communication) performs best regarding network load and processing time in various application scenarios [15, 17, 18].

It is obvious to see that mobile agents can only produce lower network load, if, simply speaking, the mobile agent's code and state that have to be transmitted are not larger than the amount of data that can be saved by the use of a mobile agent. It depends on the message size, the code and state size, and the network environment to decide whether mobile agents perform better than remote communication. In the following, we call this the *migration decision problem*. Most current solutions of the migration decision problem are based on mathematical models of the application's network load using parameter estimations (e.g. the number of negotiation steps, average message size, etc.) and lead to a decision at design time. We believe that in most application scenarios it will be beneficial to address the migration decision problem at run-time, because only then all influencing parameters (network throughput, latency, request message size, code size) can be actually known.

In this paper we present an approach to solve the migration decision problem at run-time involving dynamic prediction of message sizes based on historical information about previous communication acts. The agents do not only forecast message sizes but also the load of the remote agent server that an agent would migrate to. Although an agent might opt for migration when considering only message sizes, a remote communication could be the better alternative if also server load is taken into account. Our solution is inspired by the concept of inductive reasoning and bounded rationality introduced by Arthur [1] and does not rely on additional communication between the agents to decide which agent should migrate to which agent server. It performs well for large and highly dynamic systems and can easily be adapted to various system sizes. In addition, the overall system performance is not affected in case of agents fail or leave the system.

The remainder of the paper is structured as follows: The next section gives an overview of the related work already done in the area of network aware programs. Section 3 describes the model for communication and coordination costs that is used for this work and our assumptions and constraints. It is following a section describing our simulations and the experimental validation of the simulation results. An outlook to the future work concludes the paper.

2. RELATED WORK

Current approaches for deciding between remote communication and agent migration are based on static mathematical analysis of the application and network models. We argue that such static approaches are not sufficient for all kinds of application and should be complemented by adaptive decision models which enable the agent to decide between remote communication and migration during run-time autonomously.

To illustrate our discussion, we use the following simple model. In the remote communication approach, agent A_1 sends a request message of size B_{req} to the remote agent A_2, which answers with a result message of size B_{res}. The amount of bytes that are sent over the network equals $B_{RC} = B_{req} + B_{res}$. In the migration approach, agent A_1, which contains code of size B_c and state information of size $B_s + B_{req}$ migrates to the remote agent server (here we confine

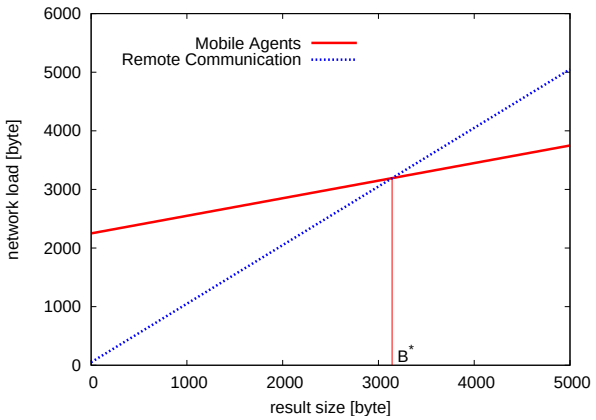


Figure 1: Evaluation of a simple mathematical model with [kB] $B_{req} = 50, B_c = 2000, B_s = 100, \varsigma = 0.7$ **and break-even point** $B^* \approx 3143$

to one mobile agent, instead of two as in the more general case mentioned in the previous section). At the remote agent server, the agent communicates locally with agent A_2, which does not produce any network load. Agent A_1 has code to filter or compress the reply message, so that only $(1-\varsigma)B_{res}$ must be carried to the agent's home agency. Note that the agent neither carries its code nor the request message back home, because the code is already available at the home server and the request message is no longer needed. However, state information must be sent back to the agent's home agency. Thus, the amount of bytes sent in the mobile agent approach equals $B_{MA} = B_c + 2B_s + B_{req} + (1-\varsigma)B_{res}$. When we evaluate this model using an artificial parameter setting, we see that there is a break-even point B^*, at which the migration overhead produced by mobile agents' code and state relocation is compensated by their ability for data reduction (compare Fig. 1).

Picco's approach is to estimate all influencing parameters and decide statically between these two design paradigms [16]. That approach is suitable for scenarios in which all the parameters are known to have constant size (e.g. the result message is a telephone number). Even if the size of parameters (e.g. the result size) are Poisson distributed with an expectation λ and $B^* \gg \lambda$ or $B^* \ll \lambda$ (i.e. the probability for a wrong decision is very low), a static decision is sufficient. In contrast, if B^* is close to λ or the distribution of reply sizes is not Poisson, changes over time or is simply unknown, a static decision on design paradigms is error-prone.

Another drawback of Picco's approach is that it does not take into account the possibility of a mixture of remote communications and agent migrations. It was already identified by Chia and Kannapan to be beneficial in terms of network load in the case of only two data servers [7]. Later, Strasser and Schwehm developed an algorithm to determine the optimal agent's itinerary of n data servers under the assumption of full knowledge [19]. In their approach, the agent only migrates to a subset of all data servers, whereas the other servers are accessed using remote procedure calls. The optimal agent itinerary depends on the size of requests and results, the number of communication steps and on the network quality between each pair of nodes. All these parameters are assumed to be estimated or known in advance.

Later, this approach was extended by a technique to provide information about network quality using distance maps before planning the optimal migration itinerary [20].

The second problem that has to be considered is the load of the agent server to which a mobile agent wants to migrate to. Even if a migration is beneficial from the perspective of network load, a migration has to be dismissed, if the destination agent server is already overloaded by other agents or server tasks. This problem has mainly been accounted from the server or system perspective so far. For example, Flüs [9] has proposed techniques for capacity planning of mobile agent servers and extended the Tracy [3] agent toolkit by techniques to avoid server overloading by refusing or queuing incoming agents. From a system perspective, the problem of load balancing between different agent servers is very important. Georgousopoulos et al. [10] present a load balancing mechanism based on a combination of state and model-based approaches. Cao et al. [5] describe a load balancing mechanism for Web servers based on the mobile agent paradigm.

A combined solution of both problems can be considered as a self-organizing approach to minimize network and server load of a multi-agent systems. To the best of our knowledge, there is no approach in the literature that combines both network load optimization and server load optimization so far.

3. MODEL DESCRIPTION

We model a distributed multi-agent system as a network of nodes $\mathcal{L} = \{L_1, \ldots, L_\mathrm{m}\}$ on each of which an agent server is running. The agent system comprises of agents $\mathcal{A} = \{A_1, \ldots, A_\mathrm{n}\}$, each located on its home agent server $h(A_i)$ at the beginning of its life-cycle. The map $\mathcal{P} : \mathcal{A} \rightarrow \mathcal{L}$ defines the location of each agent in general. Agents that are bound to large databases or otherwise immobile legacy systems are called stationary. All other agents are considered to be mobile.

Each agent has to process a list of communication steps $\mathcal{C} = \{c_\mathrm{i} \mid i : 1 \ldots p\}$ as part of its task. Each communication step $c_\mathrm{i} = \langle A_\mathrm{a}, A_\mathrm{b}, m_\mathrm{k}, m_\mathrm{j} \rangle_i$ defines that a request message m_k is sent from agent A_a (source) to agent A_b (destination), which responses with a reply message m_j. Each message m_k can be seen as an arbitrary sequence of bytes of the length $B_\mathrm{m}(m_\mathrm{k})$. The network cost for a remote communication step is calculated as follows:

$$B_{RC}(c_\mathrm{i}) = \begin{cases} 0 & if\ \mathcal{P}(A_\mathrm{a}) = \mathcal{P}(A_\mathrm{b}) \\ B_\mathrm{m}(m_\mathrm{k}) + B_\mathrm{m}(m_\mathrm{j}) & otherwise \end{cases} \quad (1)$$

Before each communication step, both involved agents (provided they are mobile) have to decide, whether to migrate from their current location $\mathcal{P}(A_\mathrm{i})$ to another location $\mathcal{P}(A_\mathrm{i})'$ before commencing the communication act. In the following we assume that both agents migrate to the same destination server. The network load only comprises of the cost for transferring the agents' code of size B_c and state of size B_s (which is equal to the size of the request message) but no costs for local communication. After the communication, both agents migrate back to their home agent servers (without code, because the code is already available at their home agent servers). Agent A_a carries back state information and results of size $(1-\varsigma)B_\mathrm{m}(m_\mathrm{j})$ with $0 \leq \varsigma \leq 1$, because the size of the original reply message could be reduced (e.g. filtered or compressed). The mobile agent communication network cost is calculated by

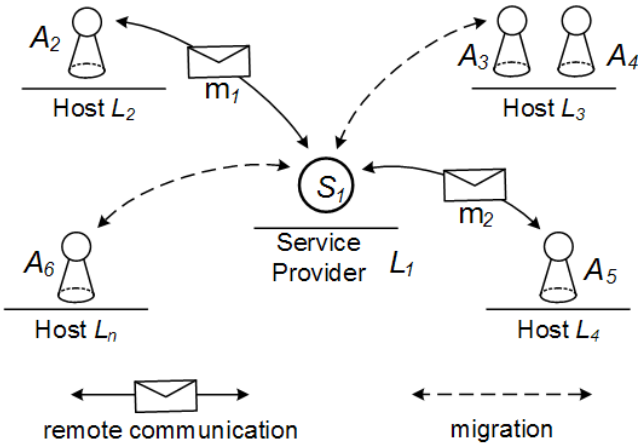


Figure 2: Communication model with multiple agents and a single server.

$$B_{MA}(c_\mathrm{i}) = \begin{cases} 0 & if \mathcal{P}(A_\mathrm{a}) = \mathcal{P}(A_\mathrm{b}) \\ B_c(A_\mathrm{a}) + B_\mathrm{m}(m_\mathrm{k}) + & \\ B_c(A_\mathrm{b}) + (1-\varsigma)B_\mathrm{m}(m_\mathrm{j}) & otherwise \end{cases} \quad (2)$$

if both agents are mobile. If only the sender agent is mobile, code transmission of agent A_b has to be disregarded.

The last formula is independent of the location of the destination agent server because we only consider network load and not transmission time at this stage. Formulas 1 and 2 can be used to decide between the two alternatives at run-time, provided that all message sizes are known. To further reduce the complexity of our model, we assume for our first experiments that all agents exchange messages with only one agent S_1, which is stationary and located on L_1, see Fig. 2.

4. TOWARDS EMERGENT MIGRATION BEHAVIOUR

4.1 Introduction – The El Farol Problem

We present a combined approach for reducing network load and agent server load, which is based on predicted environment parameters at run-time. As mentioned in the previous section, an agent has to decide before each communication step c_i between remote communication and agent migration. This decision must be based on limited knowledge, which only includes its own code size B_c and the request message size $B_\mathrm{m}(m_\mathrm{k})$. The size of the reply message $B_\mathrm{m}(m_\mathrm{j})$, the selectivity factor ς, and the current remote agent server load are not known in advance.

Naturally, we have to assume that neither of these values is distributed uniformly within their domain. In that case, there would be no chance to predict these values based on past experience. In order to apply machine learning algorithms for these values, they must render a certain pattern, for example some kind of probability distribution. One of our hypotheses is that in many application scenarios the reply size can be described by well-known distributions such as

Poisson or Gaussian distributions. It should be clear that this approach can be successfully applied to many application scenarios, for which a static analysis is possible as proposed by Picco [16] and Schwehm and Strasser [19]. In these cases, the reply message size, selectivity factor, and server load have a straightforward distribution (e.g. a constant value), which is easy to learn by our approach.

The prediction mechanism is inspired by inductive reasoning and bounded rationality principles introduced by Arthur [1] and demonstrated as solution of the El Farol bar problem. The original problem setting consists of 100 people to decide repetitively and independently, whether to go to the El Farol bar on a specific day of the week or not. All people have the same preference in order to decide: they will go to the bar, only if they expect less than 60 people to show up. Choices are unaffected from previous visits. There is no communication between people (e.g. to form coalitions or coordinate their decisions) and the only information available to every agent is the number of people that attended the bar in the past weeks.

Arthur's solution of the El Farol bar problem consists of a set of predictors for each patron, each able to forecast the number of patrons by applying very simple pattern based on past experience. At the beginning of the simulation, each person is assigned a randomly chosen set of predictors. In each step of the simulation the predictor that has performed best in the last week, is used to forecast the next number of attendants. The interesting result of Arthur's simulation shows a fast stabilizing process around the optimal number of 60 attendants. For more information about the El Farol bar problem, we refer to [6].

Our problem can be mapped to the El Farol bar problem by assuming agent S_1 (to which all communication is directed to) and its agent server to be the bar with limited resources. All other agents have to decide whether to migrate to that agent server or not. An agent will migrate only if the migration will reduce network costs and the destination agent server is not overloaded yet. The latter requirement can be simply modelled by a number of agents executed in parallel. We have to deal with similar effects as in the El Farol problem: if all agents believe that the server will not be overloaded, all agents will migrate and if all agents believe that the server will be overloaded, no agent will migrate – both experiences would invalidate the agents' beliefs in the environment.

The advantage of our approach is that no additional communication overhead is introduced because agents learn from their past experiences. In contrast to the original El Farol problem, we do not assume free information dissemination about the current server load to agents that did not migrate to the agent server recently. That is, only agents that have migrated to the agent server recently can benefit from up-to-date knowledge.

4.2 Network Cost Prediction

Each agent predicts environment parameters that are not know in advance but necessary for the migration decision based on its own historical data. For the prediction of message sizes and selectivity factors, this technique provides good results because we do not only use one predictor but a set of different predictors to provide a better accuracy.

The predictor that is used for the next prediction of the message size and the selectivity factor is called active predictor. After an agent has executed a communication step, its history information is updated with the measured values and all predictors will be rated again. The most accurate predictor in the last steps is chosen as the new active predictor because it is the most prospective for the next prediction. To reduce the computational overhead of the predictor evaluation, it is possible, to use the active predictor until it's relative error δ exceeds a pre-defined threshold.

As the migration decision of agent A_{j} based on the network costs is independent of any other agent A_{k}, it is fair to assign all agents the same set of predictors. Of course, the quality of the predictors influences the quality of the predicted values and it might be reasonable to introduce specialized predictors that can be applied to specific probability distributions. However, in our first experiments we were keen to learn about the quality of very simple predictors. Actually, it is not important to achieve a *correct prediction* of the environment parameters rather than making the *correct decision* based on the predicted values. The best predictor will be chosen automatically, while less accurate predictors will not be considered in the future.

For our experiments we have developed the following predictors:

- same value as n^{th}last communication step,
- moving average of last n steps,
- the trend of the last n steps,
- random value in the interval between current minimum and maximum,
- Poisson distribution based on the current history data.

4.3 Agent Server Load Prediction

Agent server load prediction differs slightly from message size prediction, because all agents compete for the same resource and agents' decisions regarding server load are dependent on each other. If all agents would predict the same resource load, than this would invalidate their beliefs. Therefore, it is important to ensure agents make different and adaptive decisions in their environment.

The selection of the active predictor differs from the technique presented previously. In contrast to the message reply size and selectivity prediction, the selection of the next server load predictor is not deterministic but has a level of uncertainty. The probability that a predictor is chosen as the new active predictor increases with the predictor's accuracy. The accuracy of a predictor is calculated from the number of correct and wrong decisions, i.e. the agent has predicted a not overloaded resource and decided to migrate.

Only the last n values are considered for calculation of the predictors accuracy because this allows a faster adaptation to changes. The number n can be configured by the agent designer. For every correct decision (the agent migrated and the resource was not overloaded), a predictor gets a positive rating. For a wrong prediction only the responsible active predictor gets a negative rating. To improve the learning process for new predictors and predictors with old historical values, they get a positive rating for correct predictions, but no penalty for wrong predictions, until they reach m predictions with $m \leq n$. The overall rating is the sum of all positive and negative ratings. The relative error of the prediction is not considered for the resource load prediction.

For example, if predictor P_1 predicted in 60% of the considered cases that the resource is not overloaded and the agent migrates to the resource while predictor P_2 predicted only 30% of the cases correct the probability of choosing P_1 is twice the probability of choosing P_2.

Usually, the agent bases the migration decision on local knowledge which means the resource load information is updated only if the agent migrates to the resource. Some experiments were conducted with global knowledge in which we updated the resource load information after every migration decision to learn about the differences.

The current approach uses a set of simple predictors, similar to the El Farol approach. Developed predictors include:

- n-period cycle predictor: predicts the same as n^{th}-last history value.
- moving average predictor: predicts the average of all values in a window of the last n history values.
- linear or non-linear predictors: predicts interpolated values the considers the last n history values.

4.4 Rule Based Configuration

For the strategic migration decision the agent uses a rule based decision system. The rules are based on the parameters provided by the message size and server load prediction to check which paradigm offers a better performance. Different rule sets can be configured to provide different configurations and priorities. The owner of an agent can decide which of the rule sets from a number of predefined rule sets are to be used for the decision making to meet different user preferences.

4.4.1 Data Collecting Phase

In the initial phase when agents have no (or not enough) historical data it is necessary to use a different type of predictors–not based on historical information and not predicting message size, but simply deciding between remote communication and migration, such as:

- Random migration strategy: The agent migrates with a given probability to the destination host.
- Always migrate strategy: The agent always migrates to the destination host. This is the fastest way to get required information for the learning phase.
- Never migrate strategy: The agent always uses remote procedure calls. In this case, the agent will never switch to the learning phase because no resource load information is updated.

After the agent has collected enough data so that at least some of the predictors are working, the agent switches to the working phase.

4.4.2 Learning and Working Phase

In the working phase, the agent uses the predicted data of message reply sizes, agent selectivity, and the expected agent server load based on local knowledge to come to a migration decision. Based on the rule set that reflects best the user preferences, an agent makes an autonomous decision which paradigm offers the highest benefit for the user.

- Minimal Network Load: Only the network load is considered for the migration decision. If the resource is crowded, the agent accepts waiting times.
- Balanced strategy: The agent will use migration when it offers benefit compared to RPC and the resource won't be crowded.
- Best Performance: Same as minimal network load, except that the migration probability increases linear with the age of the newest historical data or the average age of historical data in the sliding window, if the resource load data is out of date. This prevents that an agent create the habit not to migrate, if it has decided once like this. The predictor always predicts not to migrate, based on the same historical values.

5. EXPERIMENTAL VALIDATION

5.1 Outline

We have developed a first simulation test-bed in the Java programming language for a distributed multi-agent system independent of any specific agent toolkit. Our simulation environment enables different configurations of the number of agents, the number of communication steps for each agent, various distributions of reply sizes and selectivity factors for each agent, etc. The simulation uses an event-driven model to trigger agent activities. Network load equals the number of bytes transmitted without taking network protocol overhead (IP, TCP, application layer) into account. The latter will be considered in the near future, when the simulation environment will be re-designed to work with a well-known network simulation tool such as NS-2 or Omnet++. Network transmission time is modelled by taking bandwidth and latency between each pair of network nodes into account.

We have tested a number of different configurations of our simulation environment and for the evaluation we measured the following system metrics:

Reply size: The size of the reply message.

Selectivity factor: The agent's selectivity factor for the reply message.

Agent server load: The number of agents that attended the server in the last simulation step.

Several sets of experiments were conducted with different simulated configurations. We repeated each experiment several times and show results of representative experiments and some average values for illustration.

The first set of experiments focuses on the accuracy of message size and selectivity factor prediction. Both form the foundation for the strategic migration decision based on network load. The second set of experiments focuses on prediction of message size as well as agent server load, and investigates if and how fast agents can adapt to varying levels of server availability.

Usually we run experiments while varying only one parameter and keeping all other parameters constant. A detailed description of the simulation environment and the observed parameters follows in the next sections.

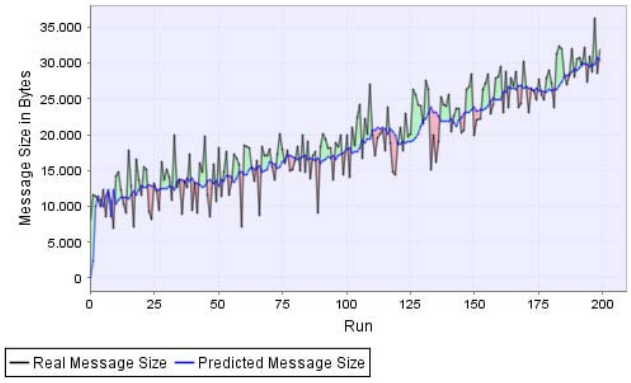


Figure 3: Comparison of the predicted message size and the real message size.

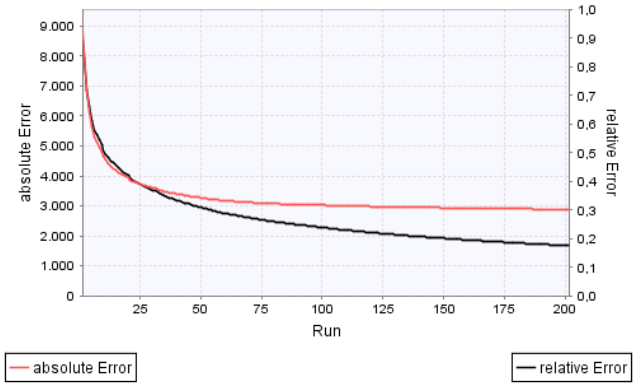


Figure 4: Average absolute and relative error of the reply message size prediction.

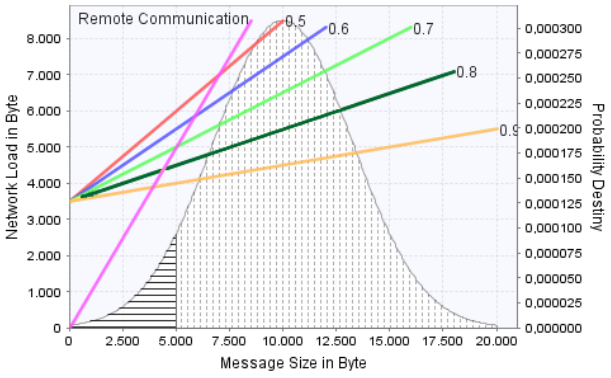


Figure 5: Comparison of mobile agent and remote communication paradigm.

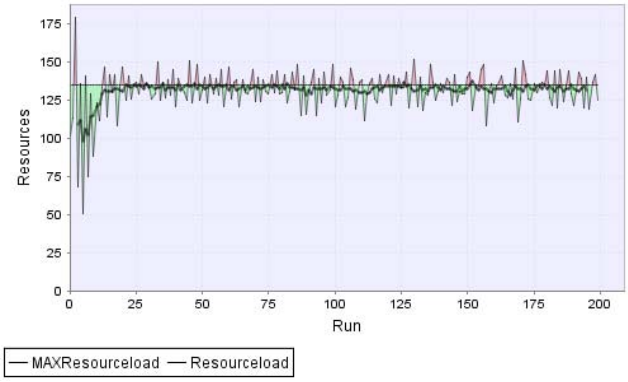


Figure 6: Development of server load for 200 cycles with resource limit of 135 units.

5.2 Message Size Prediction

In this section we report on results of message size and selectivity factor prediction. For this set of experiments we have selected different artificial Gaussian distributions of the message size with parameters $\mu = 5000$, 10000, 50000 Byte and $\sigma^2 = 10\%$, 20%, 40% of these mean values.

To validate the migration decision, we set the other parameters as follows: the agent size is 35 KByte, the agent performs 200 predictions using a set of 7 different predictors. The predictors for this experiment include 5 different period cycle predictors, a moving average predictor, and a linear predictor. Fig. 3 shows one example of message size prediction, in which the real message size is Gaussian distributed $N(10000Byte, 3250Byte)$ with an increasing mean value. We measured the absolute and relative error of the prediction and prediction correctness, i.e. whether the predicted value leads to the correct migration decision. Fig. 4 shows that the absolute error finally reaches 3000 KB and the relative error is below 20%.

In addition, we measured the cumulative values of these parameters. Figure 5 compares the network load of the remote communication and migration approach for different selectivity factors and message reply sizes for a representative message reply size distribution with $\mu = 10000$ $Byte$ and $\sigma = 32,5\% = 3250$ $Byte$. Considering only a statistical analysis of a fixed set of parameters and the given message reply size distribution, we can calculate that the break even point for the mobile agent paradigm with a selectivity of 70% is at 5000 Byte. An incorrect decision is made in less than 6,3% of all cases. We expect that real applications have certain patterns for the reply size distribution, and a small derivation.

5.3 Combined Prediction

In this section we report on results of combined message size and agent server load prediction mechanisms. In the first experiments we have selected a set of 200 agents with same agent resource requirements of 1 unit (memory, processor cycles). These agents want to use agent S_1 located on host L_1, which has available resources of $n = 135$ units. Fig. 6 shows the agent server load usage when the available server resources are constant. It can be seen that after only few executions, the average resource load balances below the maximum value. Only in few cases, the prediction over-estimates the real number of resources available. Fig. 7 shows that the number of migrations is similar for all agents. There is no group of agents that always migrates while other agents never migrate.

A second set of experiments uses the same parameters as the previous one, but the available server resources are varied at run-time after every 75 executions. The goal of this experiment is to see how fast the agents prediction mechanism can adapt to varying environmental parameter. We

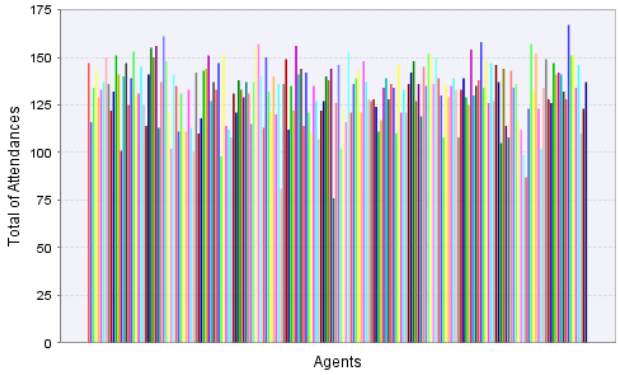


Figure 7: Attendance of mobile agents at the remote agent server for constant resource limit.

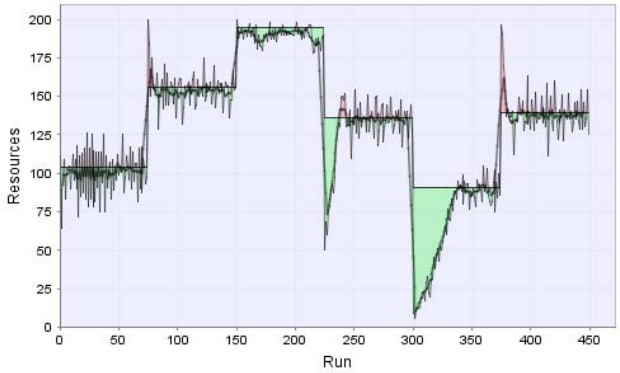


Figure 8: The adaptation of one server to varying resource capabilities.

set up an environment with different numbers of agents, and varying resource capabilities over time and measured the number of resources used by agents. We conducted experiments in which all agents have the same resource consumption as well as different resource consumption. Fig. 8 shows for one agent server that the adaption to varying server capabilities works surprisingly fast. Only in few cases, the available server capabilities are over-estimated, whereas in many cases the current server capabilities are close to the maximum values. In addition, Fig. 9 shows that even in the case of varying server resources the number of migrations is similar for all agents.

Finally, we tested two variants of updating the historical data of agents. At first, agents update their history values after every execution step, independent of having migrated to that server or not. This case resembles the original El Farol behaviour. Secondly, agents have limited and heterogeneous agent server load information. History values are only updated if an agent has migrated to the remote agent server.

Interestingly, the adaptive solution works better when agents have only limited and heterogeneous knowledge. Furthermore, the limited knowledge is preferred in real systems because it has no additional communication costs and it would be difficult to distribute resource information to potential agents.

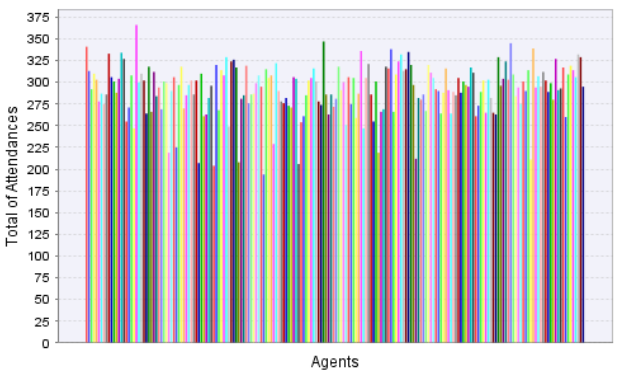


Figure 9: Attendance of mobile agents at the remote agent server for varying resource capabilities.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have presented an approach for reducing communication overhead in a distributed multi-agent system by agent mobility at run-time. Our approach also considers system performance by balancing agent server load. Our approach is based on mobile agents that decide at run-time between remote communication and migration. It was inspired by inductive learning and bounded rationality principles. The prediction of the required environment parameters enables mobile agents to autonomously make a strategic migration decision and adapt to the dynamic environment. From a system perspective, this solution can be considered as a self-organizing mechanism to minimize network load and server load in a multi-agent system.

We have presented and evaluated a number of simulation experiments involving agents in static and dynamic server load environments. The results show that this new approach for strategic migration optimization based on inductive reasoning and bounded rationality principles is very promising and justifies further investigations. The advantages of this approach are its fast adaptability to different system sizes, its simplicity, and its easy extensibility. These are known advantages, although in this particular application area rarely used. With different rule sets, agents can be configured for different user preferences.

A possible improvement would be to share historical data between all agents of one agency in order to increase the learning speed and provide better predictions since historical data are up-to-date. After we have proven that this approach works fine with the event driven model we will upgrade our Java implementation. Our next step is to switch to a more realistic asynchronous simulation environment. The simple event-driven *multiple agents using one resource* approach will be extended with techniques for multiples resources. Agents will not only migrate to one agent server, but can freely migrate in the whole multi-agent system. Each agent will have a set of tasks to fulfil that need a certain amount of system capabilities. A set of system providers located on some of the servers offer services that also consume system capabilities. These services are used by each agent to fulfil its task either using remote communication or agent migration.

Our approach for optimizing network load in multi-agent systems must be supplemented in the longer run with tech-

niques to optimize the migration behaviour of mobile agents. In this paper we have assumed rather simple migration techniques (the agent carries all its code and data information), despite of already known techniques to optimize the migration strategy of mobile agents [2, 11].

7. REFERENCES

[1] W. B. Arthur. Inductive Reasoning and Bounded Rationality. *American Economic Review (Papers and Proceedings)*, 84(2):406–411, May 1994.

[2] P. Braun and S. Kern. Towards adaptive migration techniques for mobile agents. In Z. Guessoum, editor, *Fifth Workshop on Adaptive Agents and Multi-Agent Systems (AAMAS 2005), Paris (France), March 2005*, 2005.

[3] P. Braun, I. Müller, T. Schlegel, S. Kern, V. Schau, and W. Rossak. Tracy: An extensible plugin-oriented software architecture for mobile agent toolkits. In M. Calisti, M. Klusch, and R. Unland, editors, *Software Agent-Based Applications, Platforms and Development Kits*, Whitestein Series in Software Agent Technologies, pages 357–382. Birkhäuser Verlag, 2005.

[4] P. Braun and W. R. Rossak. *Mobile Agents–Basic Concept, Mobility Models, and the Tracy Toolkit*. Morgan Kaufmann Publishers, 2005.

[5] J. Cao, Y. Sun, X. Wang, and S. K. Das. Scalable load balancing on distributed web servers using mobile agents. *Journal of Parallel Distributed Computing*, 63(10):996–1005, 2003.

[6] D. Challet, M. Marsili, and G. Ottino. Shedding light on El Farol. Technical Report 0406002, Economics Working Paper Archive at WUSTL, June 2004.

[7] T.-H. Chia and S. Kannapan. Strategically mobile agents. In K. Rothermel and R. Popescu-Zeletin, editors, *Proceedings of the First International Workshop on Mobile Agents (MA'97), Berlin (Germany), April 1997*, volume 1219 of *Lecture Notes in Computer Science*, pages 149–161. Springer-Verlag, 1997.

[8] L. Chunlin and L. Layuan. Apply agent to build Grid service management. *Journal of Network and Computer Applications*, 26:323–340, 2003.

[9] C. Flüs. *Capacity Planning of Mobile Agent Systems Designing Efficient Intranet Applications*. PhD thesis, Universität Duisburg-Essen (Germany), Feb. 2005.

[10] C. Georgousopoulos and O. F. Rana. Combining state and model-based approaches for mobile agent load balancing. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 878–885, New York, NY, USA, 2003. ACM Press.

[11] S. Kern and P. Braun. Towards adaptive migration strategies for mobile agents. In *Second GSFC/IEEE Workshop on Radical Agent Concepts (WRAC), NASA Goddard Space Flight Center, Greenbelt, MD (USA), September 2005*. Springer Verlag, 2005.

[12] A. Ludwig, P. Braun, R. Kowalczyk, and B. Franczyk. A framework for automated negotiation of service level agreements in service grids. In *1st International Workshop on Web Service Choreography and Orchestration for Business Process Management in conjunction with the Third International Conference on Business Process Management (BPM 2005), Nancy (France), September 2005*. Springer Verlag, 2005.

[13] B. D. Martino and O. F. Rana. Grid performance and resource management using mobile agents. In V. Getov, M. Gerndt, A. Hoisie, and A. M. B. Miller, editors, *Performance analysis and grid computing*, pages 251–263. Kluwer Academic Publishers, 2004.

[14] I. Müller, P. Braun, and R. Kowalczyk. A classification scheme for the integration of software agent and service-oriented paradigms. In L. Cavedon, R. Kowalczyk, Z. Maamar, D. Martin, and I. Müller, editors, *Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE'2005) in conjunction with 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2005), Utrecht (The Netherlands), July 2005*, pages 57–60, 2005.

[15] A. Outtagarts, M. Kadoch, and S. Soulhi. Client-Server and Mobile Agent: Performances Comparative Study in the Management of MIBs. In A. Karmouch and R. Impey, editors, *Mobile Agents for Telecommunication Applications, Proceedings of the First International Workshop (MATA 1999), Ottawa (Canada), October 1999*, pages 69–81. World Scientific Pub., 1999.

[16] G. P. Picco. *Understanding, Evaluating, Formalizing, and Exploiting Code Mobility*. PhD thesis, Politecnico di Torino (Italy), 1998.

[17] A. Puliafito, S. Riccobene, and M. Scarpa. Which paradigm should I use? An analytical comparison of the client-server, remote evaluation and mobile agent paradigms. *Concurrency and Computation: Practice and Experience*, 13(1):71–94, 2001.

[18] G. Samaras, M. D. Dikaiakos, C. Spyrou, and A. Liverdos. Mobile Agent Platforms for Web-Databases: A Qualitative and Quantitative Assessment. In D. S. Milojicic, editor, *Proceedings of the First International Symposium on Agent Systems and Applications (ASA'99)/Third International Symposium on Mobile Agents (MA'99), Palm Springs (USA), October 1999*, pages 50–64. IEEE Computer Society Press, 1999.

[19] M. Straßer and M. Schwehm. A performance model for mobile agent systems. In H. R. Arabnia, editor, *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97), Las Vegas (USA)*, volume 2, pages 1132–1140. CSREA Press, 1997.

[20] W. Theilmann and K. Rothermel. Dynamic distance maps of the internet. In *Proceedings IEEE INFOCOM 2000, The Conference on Computer Communications, Volume 1, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Reaching the Promised Land of Communications, Tel Aviv (Israel), March 2000*, pages 275–284. IEEE Computer Society Press, 2000.

[21] J. E. White. Mobile agents. In J. Bradshaw, editor, *Software Agents*, pages 437–472. The MIT Press, Menlo Park, CA, 1996.

Artifacts for Time-Aware Agents *

Cédric Dinont
USTL/LIFL and ISEN
59655 Villeneuve-d'Ascq, France
cedric.dinont@lifl.fr

Emmanuel Druon
ISEN 41 Bd Vauban
59046 Lille Cedex, France
emmanuel.druon@isen.fr

Philippe Mathieu
USTL/LIFL
59655 Villeneuve-d'Ascq, France
philippe.mathieu@lifl.fr

Patrick Taillibert
Thales Aerospace Division
78951 Elancourt, France
patrick.taillibert@fr.thalesgroup.com

ABSTRACT

Time-aware agents are agents capable of reasoning about their tasks duration and deadlines, and, more generally, to manage the temporal aspects of the execution of their tasks. We first focus on the case of agents in charge of long duration computations, sustaining that it is not acceptable for an autonomous agent to remain unaware of its environment for too long. We then consider deadline meetings when several time-aware agents share the same CPU. To achieve these goals, we recognize the importance of the artifact concept [16]. We introduce computational artifacts for long duration tasks and a coordination artifact for managing the CPU agenda and acting as an intermediary when agents negotiate CPU power. Control of computational artifacts is done thanks to a set of operating instructions dynamically computed by the coordination artifact.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms

Keywords

Coordination, CPU sharing, Artifacts

1. INTRODUCTION

The context of this paper is the design of intelligent agents that make use of algorithms stemming from Artificial Intelligence research. For example, planification problems in cognitive agents make use of complex algorithms, often heuristic in nature.

Using such algorithms with very high complexity leads to execution times highly variable. A problem search space can have critical regions: for a small variation of input data, the computing time necessary to find the solution can tremendously vary and even become prohibitive. Processor resources needs are thus very fluctuating and difficult to anticipate. That concerns a lot of problems which are solved by using more or less powerful heuristics.

The implementation of such systems should take into account the following difficulties:

(i) agents may "lose consciousness". Even if involved in long duration tasks, agents should stay permanently aware of their environment and should be able to process information sent by others. On the contrary, an agent that have to execute long duration algorithms would be deconnected from the outside world and would not even realize that its calculation has become useless if a received but not processed message had said so. It would unnecessarily use CPU power that could be useful to other tasks or other agents.

(ii) agents may be unable to meet their commitments. The notion of commitment is essential to the rationality of agents in a multi-agent context. When time is concerned, an agent must be able to guarantee results within a given deadline, even if it means modifying its strategy, using an alternative algorithm or giving a solution of less quality, while respecting its deadlines.

(iii) agents may share the same CPU. This is most often the case that several agents of a MAS run on the same CPU and therefore have to share CPU power. In that case, it is very difficult to ensure that time constraints will be respected. Solution have to be found since it is generally unacceptable to assume that agents do not share the same CPU.

A natural solution to the previous difficulties would be to split agents having to achieve long duration computation in two or more parts:

(i) a "main agent", that would remain aware of its environment all the time and hence open to information coming from other agents,

*This work is partially financed by Thales Aerospace Division and the région Nord-Pas-de-Calais.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

(ii) one or more "computational agents" in charge of the long duration computations, the initial agent was concerned with.

It sounds like a good approach, but one may ask if these computational agents are really autonomous? Their purpose is to work on potentially long computations, so they can stay away from their environment for long periods of time. Worse, in order to manage the CPU sharing, the main agent would need some control facilities allowing it to suspend, restart or even completely stop the execution of its associated computational agents if necessary.

So even if this solution is technically the good one, one may legitimately be reluctant to call an agent an entity with such a damaged autonomy. The solution is to rely upon the notion of artifact proposed in [20, 17] in place of computational agents. An artifact is a first class entity of any MAS giving some kind of functionality that agents *use* to achieve their goals.

We will recall artifacts and their modeling in section 2.1 and explain in section 2.2 the way they are used for embodying our computational agents. We can already mention that even if it will solve the first difficulty previously stated (avoiding agent "lost of consciousness"), meeting deadline commitments and sharing CPU still remain an open problem. This is why we propose in section 3 to use a coordination artifact which will be in charge of determining time slices allocated to each computational artifact and providing the main agent with a set of operating instructions to control its artifacts. Section 4 explains the way the coordination artifact behaves. Section 5 gives results obtained on our implementation and section 6 discusses related and future works in the context of time management in MAS.

2. COMPUTATIONAL ARTIFACTS

We are interested in time-aware agents when they have to run one or several long duration tasks while being able to meet the deadlines they are committed to. It requires that the long duration tasks are run in processes separated from the one running the agent since, in the other case, the agent might lose control over its own behaviour, obliged to wait for the end of the tasks before taking any decision.

A straightforward (but costly) solution would be to run each task on a separate processor. Nonetheless, in most of the cases, CPU has to be shared and hence the agent must be able to control the tasks, that is to decide when to start, suspend, restart or cancel them in order to meet its own deadlines. Long duration tasks could be implemented as agents but one can legitimately wonder about the autonomy of an agent that can be externally controlled!

That every item of a multiagent system can be implemented as an agent is a point that has already been raised by Ricci, Viroli and Omicini [20, 17, 16], especially in the case of coordination agents. Their answer was to introduce *artifacts* as a first-class abstraction used to design and program those aspects of multiagent systems for which the agent abstraction is not suitable. We suggest that the same approach has to be adopted for the time-aware agents we are concerned with.

The following section 2.1 summarizes the *artifact* concept, directly borrowing from [20] the process algebra based formalism they introduced. Section 2.2 proposes the *computational artifact* as an abstraction for implementing the computational activities of time-aware agents.

2.1 The Notion of Artifact

As presented in [20], if agents are conceived as goal-governed or goal-oriented systems, an artifact is an entity an agent *uses* to achieve its goals. Artifacts provide a certain function that can be exploited as a service.

An artifact is characterized by:

- *a usage interface (UI):* defined in terms of a set of operations. These operations are of two kinds: action execution and perception about the completion of an action.
- *a set of operating instructions (OI):* the (formal) description of how agents can use the artifact.
- *a behavior specification:* the (formal) description of the internal behavior of the artifact.

Note that, taking the agent point of view, to exploit a coordination artifact means to follow its operating instructions, on a step-by-step basis. Thus, the agent does not have to know *how* the functionality is implemented.

We now briefly present the formal model introduced in [20] that makes use of a process algebra to express operating instructions. An instruction $I \in \mathcal{I}$ is expressed as:

$$I ::= 0 \mid !\alpha \mid \underline{\alpha} \mid ?\pi \mid I; I \mid I + I \mid (I||I) \mid \mathcal{D}(t_1, \ldots, t_n)$$

It is defined out from atomic instructions through structured instructions. Atomic instructions include the void behavior (0), the execution of an action α ($!\alpha$), an action α being started but not yet completed ($\underline{\alpha}$) and a perception π of action completion ($?\pi$). Structured instructions are the sequential composition of two instructions by operator ";", choice by "+", and parallel composition by "||". An instruction can also be an invocation $\mathcal{D}(t_1, \ldots, t_n)$ of antoher operating instruction. Several structural congruence rules that we do not mention here express commutativity and transitivity of previous operators.

Operating instructions evolve as the agent interacts with the artifact. The possible interaction acts $\delta \in$ are the execution of action α (α) and the perception π of the completion of α ($\alpha : \pi$):

$$\delta ::= \alpha \mid \alpha : \pi$$

This language can be given operational semantics. In the following rules, notation $I \xrightarrow{\delta}_{\mathcal{I}} I'$ means that OI state $I \in \mathcal{I}$ moves to new state $I' \in \mathcal{I}$ by the occurrence of interaction act $\delta \in$. Notation $[t/t']$ is used to indicate the substitution of t by t' in the continuation of instructions.

$I\|\|I'' \xrightarrow{\delta}_{\mathcal{I}} I'\|\|I''$	if $I \xrightarrow{\delta}_{\mathcal{I}} I'$	[PAR]
$I + I' \xrightarrow{\delta}_{\mathcal{I}} I''$	if $I \xrightarrow{\delta}_{\mathcal{I}} I''$ and $I' \not\xrightarrow{\delta}_{\mathcal{I}} I'''$	[ECH]
$I + I' \xrightarrow{\delta}_{\mathcal{I}} I'' + I'''$	if $I \xrightarrow{\delta}_{\mathcal{I}} I''$ and $I' \xrightarrow{\delta}_{\mathcal{I}} I'''$	[LCH]
$!\alpha; I \xrightarrow{\alpha'}_{\mathcal{I}} \underline{\alpha'}; [\alpha/\alpha']I$		[ACT]
$\underline{\alpha}; ((?\pi; I) + I'); I'' \xrightarrow{\alpha:\pi'}_{\mathcal{I}} [\pi/\pi'](I; I'')$		[COMP]

Rule [PAR] specifies that only one OI is followed at a given time. Rule [ECH] defines the semantics of exclusive choice: if I is executed, choices inside I' are excluded. Rule [LCH]

instead defines late choice: if interaction δ allows multiple choices, δ is executed and the choice is deferred to a forthcoming interaction. Rule [ACT] specifies that if α is the next action to execute, a specialization α' of α can be executed, propagating the corresponding substitution in the continuation of instructions. Rule [COMP] similarly deals with perception of the completion of an action.

Viroli and Ricci proposed in [20] an extension to the previous algebra to take into account the notion of timeout in OIs. The definitions of instructions and interaction acts are extended as follows:

$$I ::= \ldots \mid T(n) \mid \epsilon \qquad \delta ::= \ldots \mid \tau(m)$$

$T(n)$ is the timeout instruction, with parameter n being the duration of the timeout. ϵ is an error state the OI moves to when a timeout has expired while the OI was waiting for an interaction. $\tau(m)$ is the perception of m clock ticks. As for previous instructions, following rules give an operational semantics to the timeout instruction.

$$T(n); I \xrightarrow{\tau(m)}_{\mathcal{I}} T(n-m); I \quad \text{if } n > m \quad \text{[T-A-PASS]}$$
$$T(m); I \xrightarrow{\tau(n)}_{\mathcal{I}} \epsilon \quad \text{if } n \geq m \quad \text{[T-A-EXP]}$$
$$\underline{\alpha}; T(n); I \xrightarrow{\tau(m)}_{\mathcal{I}} \underline{\alpha}; T(n-m); I \quad \text{if } n > m \quad \text{[T-P-PASS]}$$
$$\underline{\alpha}; T(m); I \xrightarrow{\tau(n)}_{\mathcal{I}} \epsilon \quad \text{if } n \geq m \quad \text{[T-P-EXP]}$$
$$T(n); !\alpha; I \xrightarrow{\alpha'}_{\mathcal{I}} \underline{\alpha'}; [\alpha/\alpha']I \quad \text{[T-ACT]}$$
$$\underline{\alpha}; T(n); ((\pi; I) + I'); I'' \xrightarrow{\alpha:\pi'}_{\mathcal{I}} [\pi/\pi'](I; I'') \quad \text{[T-COMP]}$$

Rule [T-A-PASS] deals with time passing when a timeout is not yet expired and an action is to be made. When the timeout actually expires, rule [T-A-EXP] makes the OI moving to the error state ϵ. Rules [T-P-PASS] and [T-P-EXP] have the same purpose when a perception is expected: [T-P-PASS] deals with time passing and [T-P-EXP] makes the OI moving to error state ϵ when the timeout expires. Rule [T-ACT] removes the timeout when the agent decides to execute specialization α' of action α. Rule [T-COMP] acts the same way when a perception is made before the timeout expires.

2.2 Computational Artifacts Properties

A generic computational artifact (CA) is an artifact that is given a computational job to do on some input data, giving back either results or failure, depending on the input data. Such a computational artifact needs to be controlled during its processing by the agent it is associated with. We define the minimal usage interface as follows: possible actions are `start(input_data)`, `pause`, `restart`, `stop` and possible perceptions are `finished(result)` and `failure(error_info)`. According to the capabilities of the computational artifacts really being implemented in some application, we may extend this basic model. For example, if we want agents to have the possibility to ask the progress report of the task currently being executed or being regularly informed of the progress, the following action and perception will be added: `query_progress_report`, `progress_report`.

As we will see when trying to have several artifacts sharing the same CPU, we need mechanisms for time representation and manipulation. Timeouts proposed in [20] intuitively mean that the agent has to do an action or that a perception must occur *before* a given point in time. We also need the possibility to state that the agent has to do an action *after* a given point in time.

Hence, as for the timeout, we extend the definition of an instruction with the new atomic instruction $W(n)$. The following operational rules are added:

$$W(n); I \xrightarrow{\tau(m)}_{\mathcal{I}} W(n-m); I \quad \text{if } n > m \quad \text{[W-T-PASS]}$$
$$W(m); I \xrightarrow{\tau(n)}_{\mathcal{I}} I \quad \text{if } n > m \quad \text{[W-T-EXP]}$$

Rule [W-T-PASS] deals with time passing when the waiting time is not over. Rule [W-T-EXP] deals with the expiration of the delay imposed before being able to deal with next instruction. Note that there is no [W-ACT] rule. In the timeout case, an action can be started before the timeout has expired with rule [T-ACT]. Here, the agent cannot start an action before the end of the waiting time.

To clarify these notions, here is the example of an OI that may be used to control and interact with a computational artifact:

```
OI := W(10); !start(data);
  ((T(5); !pause; W(10); !restart; T(5); !stop)
    + ?finished(result)
    + ?failure(error_info))
```

The agent has to wait 10 time ticks before ordering the artifact to start its task with input `data`. Then, it has to order the artifact to pause within 5 time ticks and to restart no less than 10 time ticks later. Finally, if the work did not finish (normally or with an error) within 5 time ticks, the agent has to order the artifact to stop its computational task.

3. A COORDINATION ARTIFACT FOR CPU SHARING

In this section, we investigate the necessary coordination between agents to ensure meeting deadlines when they share a single CPU. We need an entity that can manage the CPU schedule and that can be used by agents to agree on the use of CPU power. As said before, this is typically where artifacts are useful. We thus define a coordination artifact called CPU Resource Sharing Artifact (CRSA) that generates operating instructions agents have to use to ensure all artifacts can meet their deadlines and which acts as an intermediary when agents negotiate CPU power.

Figure 1 reports the detailed definitions of the CRSA operating instructions. `Def_CA_Management` is the OI used by agents to interact with the CRSA when they have a computational job to give to a computational artifact, while `Def_Negotiation_Initiator` is the OI used during the previous process if the CRSA detects inconsistencies raised by adding the new computational task to the agenda. In this case, the `Def_Negotiation_Participant` OI is engaged so that other agents can propose to sacrifice a part of the CPU power they use.

Def_CA_Management. An agent which wants to delegate a computational job to a computational artifact first executes action `request(r_content)`. We consider that `r_content` is a term with four variables: an identifier `id`, an estimation `load` of the CPU power needed to finish the task, a list `t_c` of timing constraints the work has to respect and `ca_oi`, the OI that is the object of the request and which is bounded by the CRSA when no inconsistencies were found in the scheduling process. It waits for an answer from the CRSA

```
Def_CA_Management :=
  !request(r_content);T(r_timeout);
  ((?accepted(r_content);((r_content.ca_oi+0)||Def_CA_Management)+
   (?rejected(i_rejection);Def_Negotiation_Initiator(i_rejection));
     Def_CA_Management));Def_CA_Management

Def_Negotiation_Initiator(i_rejection) :=
  !start_negotiation(i_negotiation, i_rejection);  T(n_timeout);(
    ?negotiation_failed+
    ?negotiation_succeeded(i_negotation))

Def_Negotiation_Participant :=
  !get_sacrifice_request;(?finished+(?new_sacrifice_request(s_content);(
    T(s_timeout)+
    !refuse(s_content)+
    (!propose_sacrifice(s_content);(
       ?commit_sacrifice(s_content)+
       ?cancel_sacrifice(s_content))))));Def_Negotiation_Participant
```

Figure 1: CRSA Operating Instructions

Action	Precondition
request	B possible(r_content) & ¬B have_oi(r_content)
start_negotiation	B work_importance(r_content.id, s_content.imp) & B remain_work_load(r_content, s_content.load)
get_sacrifice_request	-
refuse	B ¬more_important(s_content, my_work)
propose_sacrifice	B more_important(s_content, my_work) & B can_sacrifice_work_load(s_content.load)

Perception	Effect
accepted	B have_oi(r_content.ca_oi)
rejected	-
negotiation_failed	B ¬possible(r_content)
negotiation_succeeded	B have_oi(i_negotiation.r_content)
new_sacrifice_request	-
commit_sacrifice	-
cancel_sacrifice	-

Figure 2: Semantic link between CRSA OI and agents mental state

for `r_timeout` time ticks. Answer can be an acceptance or refusal. In first case, the agent can decide or not to follow the OI `r_content.ca_oi` created by the CRSA. When a refusal is received, the agent tries to negotiate with other agents by following the OI `Def_Negotiation_Initiator`. In each case, a new call to `Def_CA_Management` is made so that the agent can do a new computational job request.

Def_Negotiation_Initiator. To simplify the negotiation process, we consider that agents have a common means of expressing the importance of a computational work, so that an agent that receives a proposal to sacrifice some CPU power can compare the importance of its works with the importance of the work it is asked to sacrifice for. The term `i_negotiation` contains the information of `r_content` and the information `imp`, calculated by the agent, that represents the importance of the requested work. The agent simply executes action `start_negotiation` and waits for an answer for `n_timeout` time ticks. If negotiation succeeds, the CRSA generates an OI to control the computational artifact and binds it to `ca_oi` in `i_negotiation`. Note that the initiator does not have any view on the negotiation process: is it managed by CRSA internal behavior.

Def_Negotiation_Participant. A participant to the negotiation protocol executes in a recursive way the action `get_sacrifice_request`. The perceptions associated to this action are `finished`, indicating that the participation of the agent to the protocol is over; and `new_sacrifice_request`, indicating that a new sacrifice request `s_content` is available. The term `s_content` contains information from `i_negotiation` and information `s_result`, calculated by the participant, on the sacrifice it can make. When a new sacrifice request is received, the agent has `s_timeout` time ticks to answer. It can refuse to sacrifice itself or it can accept, providing a proposal that will be used by the CRSA to verify if, with the sacrifice, the new task could be allocated. De-

pending on the case, the CRSA asks the agent to commit or to forget the proposed sacrifice.

Operating instructions, given alone, are not sufficient to agents. This is indeed necessary to give to agents enough information to understand the OI. We thus provide a link between the operating instructions and the agents mental state. Figure 2 describes this link in the case of our coordination artifact CRSA. We use the mentalistic semantics to actions and perceptions described in [20]. Preconditions in terms of agent beliefs are attached to OI actions and effects on agent beliefs to OI perceptions.

Preconditions of actions. An agent issues a request to the CRSA if it believes that it is possible to achieve the task (`Bpossible(r_content)`) and if it does not have an operating instruction to interact with the computational artifact yet (`¬Bhave_oi(r_content)`). Before starting a negotiation, it fills some fields in the `s_content` term (the importance and the needed load on CPU of its work). An agent refuses to sacrifice itself if it believes that its work is more important than the one in the request (`B¬more_important(s_content, my_work)`). On the contrary, it proposes to sacrifice itself if it believes that its work is less important. The proposition it does in this case depends on its beliefs of how much work load it can sacrifice (`Bcan_sacrifice_work_load(s_content.load)`).

Effects of perceptions. If an agent gets the perception that its request has been accepted or that the negotiation has succeeded, it believes that `r_content.ca_oi` is bound to a valid OI it can use to interact with a computational artifact (`Bhave_oi(r_content)`). In case the negotiation failed, it updates its mental state not to believe it is possible to do the requested task (`B¬possible(r_content)`).

4. CRSA BEHAVIOR SPECIFICATION

In this section, we describe the internal behavior of the CRSA. It mainly consists of a scheduling algorithm used to create the operating instructions it will return to agents. We overview the properties of such an algorithm, we give some information about how the scheduling is done and we give some examples of real generated operating instructions.

The scheduler that we need must have particular properties. It is constructed on top of the operating system scheduler. It can take into account (at its level) that several tasks can be carried out in parallel. It must manage differently agents, that are always active, and computational artifacts whose tasks have deadlines and therefore may be paused to ensure that other computational artifacts can meet their deadlines.

We consider that it is interesting to begin tasks as soon as possible. The behavior of a scheduler like Earliest Deadline First (EDF) [12], which will entirely execute the task whose deadline is the earliest and then turn to the next task, is not adapted to the resolution of a problem by various agents running different heuristics in parallel. In this kind of application, it is interesting to start the various heuristics as soon as possible and to stop the resolution as soon as one has found the required solution.

We model this scheduling problem with intervals and subintervals that split the scheduling horizon[4]. Each subinterval is characterized by the tasks which are allowed to run in it. In a subinterval containing n tasks, each task receives a share of $1/n$ of the CPU power in this subinterval. The scheduling problem is reduced to fixing the duration of each subinterval and is expressed as a linear program solved by the simplex algorithm.

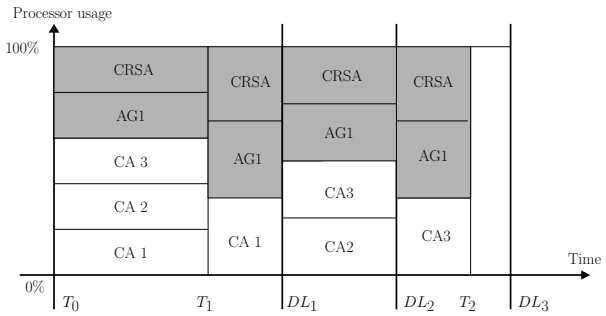


Figure 3: Schedule example

Figure 3 shows an example of such a schedule. There is an agent, the CRSA and 3 computational artifacts CA1, CA2 and CA3. Each computational artifact is assigned a task with respective deadlines DL_1, DL_2 and DL_3. We see that the main agent and the CRSA are considered to be always running (they are supposed to be fully time-aware), as they appear in each subinterval. The scheduler has determined that computational artifacts CA2 and CA3 could run until T_1 while ensuring that CA1 could respect its deadline DL_1. Here is the code of the OI generated by the CRSA for this example of schedule[1].

```
OI_CA1 := W(T0);!start(input_data);(
  (T(DL1);!stop)
  +?failure(error_info)+?finished(result))

OI_CA2 := W(T0);!start(input_data);(
  (T(T1);!pause;W(DL1);!restart;T(DL2);!stop)
  +?failure(error_info)+?finished(result))

OI_CA3 := W(T0);!start(input_data);(
  (T(T1);!pause;W(DL1);!restart;T(T2);!stop)
  +?failure(error_info)+?finished(result))
```

As this is not the main concern of the paper, we do not explain here all the details of the algorithm. The interested reader may find in [4] the algorithm details, explanations of implementation concerns and other examples of obtained schedules.

5. APPLICATION

We used the CPU resources sharing artifact previously described in a MAS which aim is to localize boats from an airplane by the sole use of the signal coming from the boat radars; it is then possible for the plane to stay hidden from the boats, which remain unaware of being spied. This procedure is called *passive localization*. It is useful in several practical situations such as the prevention of oil dumping from oil tankers – and also of course in lots of military applications.

[1]Note that, in this example, for simplicity reasons, we abuse notations for timeout and wait instructions by using absolute values of time points instead of durations as arguments to `T` and `W`. For example, we would rather use `T`($DL_1 - t_\alpha$) instead of T(DL_1) in `OI_CA1`, with t_α being the absolute time the last action was taken at.

The plane (the agent) gets the bearing of the boats on a regular basis corresponding to the periodicity of the boat radars. The bearings measured are imprecise of several degrees and possibly missing, due to the harsh operating conditions which generally prevail. This approach is comparable to goniometry but boats and plane are moving, which makes things really more difficult. The computation of a location for the plane is only possible when the plane is faster than the boats, which hopefully is generally the case. This speed difference is exploited by an interval propagation algorithm on continuous domain [10] which progressively narrows the possible area of the boats in combining measurements as and when they arrive. Interval propagation is especially well adapted for this task since it allows the incremental adjunction of constraints at each new measurement and also the use of non-linear constraints, in our case trigonometrical formulas.

Time-awareness is very important for an agent using such an algorithm since the duration of the computation might change dramatically due to possible slow convergence occurring in the propagation process [11]. Actions must be taken to avoid blocking the agent for a long time. Computational artifacts are a solution since they allow the uncoupling of the cognitive part of the agent and the time-unpredictable computation tasks it must carry out. It is then possible for the agent to decide whether or not the computation must be stopped, whether or not a new measurement must be considered...

Our system uses a proprietary Prolog-based interval constraint propagator (Interlog [3]) which makes it possible to specify a time contract when initiating a resolution step. When the contract expires, if the processing is not already achieved, Interlog stops and returns the present (interval) location of the corresponding boat. The agent can decide whether to resume the propagation and possibly improve the accuracy of the localization, or to introduce new constraints depending on the arrival or not of new measurements. Assuming that the period of the radar is known (which is practically the case after few measurements) it should be possible to meet the deadline requiring that the artifact is available when a new measurement arrives. Such a requirement is relevant since a new measurement conveys a more accurate information taking into account the last position of the boat and the plane. The new measurement is thus to be considered as soon as possible.

The problem is that the plane has to monitor several boats at the same time and hence several artifacts are competing for accessing the CPU. The number of boats under consideration can be chosen by the agent in order to ensure that all the required processing is feasible. Nonetheless, because of the de-synchronization between the radars of the boats (they all have their own period and run independently), it is impossible to guarantee the meeting of the deadlines without the use of a coordination mechanism such as the one proposed by the CRSA to manage the CPU.

More formally, the problem is the following:
Let B_i be the boats, the number of which can change over time, and p_i the period of the B_i radar (it is actually an upper bound since signals might be missing but it becomes more and more certain in the course of time).

When a new measurement j arrives for boat B_i, a new task $T_{i,j}$ is created, the deadline of which is: $DL_{i,j} = T + \tau_{i,j}$, with T the time of arrival and $\tau_{i,j}$ the processing duration.

Since few is known about processing duration, the same value PD is taken and used as a time contract (in CPU time) for the interval propagator for each new task. Hence: $\forall_i \forall_j : \tau_{i,j} = PD$.

At the end of the contract, a result is always available because, thanks to the monotonicity of the narrowing procedure used by the interval propagator, there is at anytime a valid interval encompassing the actual position of the boat that can be returned.

We ran experiments in a simplified version of the application described above to exhibit the benefits of the use of the CRSA in this application. We compared results obtained in the following cases:

- *when no coordination mechanism is used to manage the CPU power allocated to computational artifacts.* There is no guarantee that the computation for a measurement will be finished before a new measurement arrives. When this is the case, the artifact continues its computation until the time contract is reached and the new measurement is ignored.

- *when the CRSA services are used.* When the CRSA accepts to schedule a task, it guarantees that the computation will finish before the new measurement arrives. If the new task cannot be inserted in the schedule, the corresponding measurement is ignored.

We calculated in these two cases the percentages of measurements that were ignored. We used the following values for the parameters presented above:

- number of boats: 10,
- p_i: 5 to 30 seconds (real time),
- PD: 2 seconds (CPU time).

In the first case, 42% of measurements are ignored. Because of the de-synchronization between the radars, there are times when many measurements arrive almost at the same time, increasing the number of ignored measurements; some time later the CPU can be idlind for some seconds, all computations being finished and no new measurement arriving. The use of the CRSA coordination mechanism brings a significative improvement, with only 7% of ignored measurements and the CPU being constantly used.

6. RELATED AND FUTURE WORKS

In this section, we explore related work in time management in intelligent agent systems with the prospect of improving the notions and (formal) models that were used in this paper.

Time management is a transverse and vast concept in MAS [7]. From our point of view, it can be split into three concerns: *(i)* temporal reasoning, in which planning and scheduling are the two topics that interest us most, *(ii)* execution and monitoring and *(iii)* the link between these two phases. Item *(i)* is pointless if nothing is done to manage point *(ii)* and the most difficult problem is to unify these two viewpoints so that the modules that manage them can interoperate.

Time management already was a key point in Agent-0 [18]: the language allows agents to schedule commitments they

have to fulfill in the future and the framework is responsible of calling the corresponding actions when necessary. In Agent-0, there is a single control loop. On the contrary, InteRRaP [13] introduced several control layers that are respectively in charge of reactive behavior, planification and cooperation. There is a clear distinction between the three points of time management we cited before. To bridge the gap between temporal reasoning and execution and to allow temporal verifications, Concurrent MetateM[6] uses agent specifications that are directly executable, but is limited to reactive agents.

Several work as also been done to manage unpredictable hard and soft deadlines that are only known at execution time. The approach proposed by Adelantado et al. [1] to this problem uses two levels: a reactive meta-level that makes the agent aware of its environment and that controls the base level, which is *anytime* based. To our point of view, their approach cannot be applied to a wide range of applications because of the difficulty to find anytime algorithms for a particular problem. Moreover, our approach is based on *continuous* meta-level reasoning, which can give more flexibility than reactive approach, and the goal of computational artifacts is to embed legacy code that can have bad temporal properties.

To deal with this problem of deadline unpredictability, Lalanda et al. [9] propose to provide run-time flexibility to agents with multiple choices to realize an action. Agents have goals with hard or soft deadlines and goals have different priorities. The agents continually reason about their deadlines to choose the next action that they will execute. This reasoning is based on a scheduler that ensures deadlines meeting and a threshold based decision process to interleave the execution of the agent plans. Lalanda et al. applied their techniques to an office robot application. The robot is given tasks as book fetching and delivery, environment learning or environment inspection. The robot can decide to degrade its performance if it cannot fulfill all its goals. Our approach is more or less similar except that the tasks of our agents are computational tasks. We view time passing as the CPU time consumed by artifacts. They compete to use the CPU and agents have to cooperate to obtain the CPU power they need to meet their deadlines.

Omicini et al. describe in [15] an extension of the ReSpecT language to manage time in coordination artifacts. An extension of the well known *dining philosopher* problem is given as example. It manages the case when philosophers do not release back the chopsticks. The final coordination policy constraints the maximum time an agent can use the shared resources. Their approach is to allow *coordination artifacts* programs to contain timed instructions (wait, timeouts, timed insertions/deletions of reactions), while our approach is to allow *agents* reason about computation time passing and cooperate to achieve their goals.

An interesting field of research to improve this work is the one initiated by IMPACT [2] and extended by Temporal Agent Programs [5]. Interesting notions like checkpoints and history could be usefully exploited in our framework. More generally, having a formal model of algorithms used in computational artifacts is of great importance. This is the idea behind anytime computation [22]. However, finding an anytime algorithm to solve real-world problem is often impossible. Finding adaptable algorithms is nevertheless the good way to follow [8, 19]. Moreover, we must go towards flexibility. Agents must not be in a situation where they are unable to do anything. They need to have several choices to respect their temporal constraints and have the means to do the right choice [21, 14].

Last, but not least, MAS are distributed in nature. We have thus to extend our CPU Resource Sharing Artifact so that it can give information to agents about the load of other CPUs in order to let them decide to migrate.

7. CONCLUSION

In this paper, we investigated one aspect of time-awareness in multi-agent systems: being able to meet deadlines on heavily computational tasks in a context of CPU sharing, while remaining aware of the environment. We showed that modeling computational tasks with agents in this context is not satisfying. We used the notion of artifact developed in [20] to introduce computational artifacts, entities that provide computational functionalities to agents that use them to achieve their goals. As this is not sufficient to ensure deadlines meeting, we also introduced a coordination artifact (the CPU Resource Sharing Artifact) which purpose is to manage the CPU agenda and to act as an intermediary when agents need to negotiate to share the available CPU power. It is worth noting that the CRSA only centralizes information on the CPU agenda but does not decide anything when a conflict appears in the schedule. Agents keep entire control on the sacrifice decision process.

8. REFERENCES

[1] M. Adelantado and S. de Givry. Reactive/anytime agents - towards intelligent agents with real-time performance. In *IJCAI'95 Workshop on Anytime Algorithms and Deliberation Scheduling*, 1995.

[2] K. A. Arisha, F. Ozcan, R. Ross, V. S. Subrahmanian, T. Eiter, and S. Kraus. IMPACT: A platform for collaborating agents. *IEEE Intelligent Systems*, 14(2):64–72, 1999.

[3] B. Botella and P. Taillibert. Interlog : Constraint logic programming on numeric intervals. *3rd Int. Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics*, 1993.

[4] C. Dinont, E. Druon, P. Mathieu, and P. Taillibert. CPU sharing for autonomous time-aware agents. In *COGIS'06*, 2006.

[5] J. Dix, S. Kraus, and V. S. Subrahmanian. Temporal agent programs. *Artificial Intelligence*, 127(1):87–135, 2001.

[6] M. Fisher. Concurrent METATEM - a language for modelling reactive systems. In *Parallel Architectures and Languages Europe*, pages 185–196, 1993.

[7] M. Fisher, D. Gabbay, and L. Vila, editors. *Handbook of Temporal Reasoning in Artificial Intelligence*. Elsevier, 2005.

[8] E. Horvitz and G. Rutledge. Time-dependent utility and action under uncertainty. In *Proceedings of the 7th conference on Uncertainty in artificial intelligence*, 1991.

[9] P. Lalanda and B. Hayes-Roth. Deadline management in intelligent agents. Technical report, Knowledge Systems Lab, 1994.

[10] O. Lhomme. Consistency techniques for numeric CSPs. In *IJCAI'93*, 1993.

[11] O. Lhomme, A. Gotlieb, and M. Rueher. Dynamic optimization of interval narrowing algorithms. *Journal of Logic Programming*, 19-20, 1994.

[12] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[13] J. Muller and M. Pischel. The agent architecture InteRRaP: Concept and application, 1993.

[14] N. Muscettola. Incremental maximum flows for fast envelope computation. In *ICAPS*, pages 260–269, 2004.

[15] A. Omicini, A. Ricci, and M. Viroli. Time-aware coordination in ReSpecT. In J.-M. Jacquet and G. P. Picco, editors, *Proc. of COORDINATION 2005*, volume 3454 of *Lecture Notes in Computer Science*, pages 268–282. Springer, 2005.

[16] A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini. Coordination artifacts: Environment-based coordination for intelligent agents. In *AAMAS'04*, 2004.

[17] A. Ricci, M. Viroli, and A. Omicini. Programming mas with artifacts. In *ProMAS'05*, 2005.

[18] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[19] R. Vincent, B. Horling, V. Lesser, and T. Wagner. Implementing soft real-time agent control. Technical report, 2000.

[20] M. Viroli and A. Ricci. Instruction-based semantics of agent mediated interaction. In *AAMAS'04*, 2004.

[21] T. Wagner. *Toward Quantified, Organizationally Centered, Decision Making and Coordination*. PhD thesis, University of Massachusetts, 2000.

[22] S. Zilberstein and A. I. Mouaddib. Reactive control of dynamic progressive processing. In *Proceedings of the 16th IJCAI*, 1999.

Total Performance by Local Agent Selection Strategies in Multi-Agent Systems

Toshiharu Sugawara[1] Satoshi Kurihara[2] Toshio Hirotsu[3] Kensuke Fukuda[5]
Shinya Sato[4] and Osamu Akashi[4]

[1]NTT Communication Science Laboratories
2–4 Keihanna City,Soraku-gun
Kyoto 619-0237, Japan
sugawara@t.ecl.net

[2]Inst. of Scientific and Industrial Research
Osaka University
kurihara@ist.osaka-u.ac.jp

[3]Dept. of Information and Computer Sciences
Toyohashi University of Technology
hirotsu@entia.org

[4]NTT Network Innovation Laboratories
3–9–11 Midori-cho, Musashio
Tokyo 180-8585 Japan
{sato,akashi}@t.ecl.net

[5]National Information Institute
2–1–2 Hitotsubashi, Chiyoda-ku
Tokyo 101-8430, Japan
kensuke@nii.ac.jp

ABSTRACT

In order to achieve efficient progress in activities such as e-commerce and e-transactions in an open environment like the Internet, an agent must choose appropriate partner agents for collaboration. However, agents have no global information about the whole multi-agent system (MAS) and the state of the Internet; therefore, they must select the appropriate partners based on local knowledge and local observations. In this paper, using a multi-agent simulation, we discuss how total MAS performances are affected by local decisions when agents select partners to collaborate with. We also investigate how MAS performances change and how network structures between agents shift according to the progress of agents' local learning and observations. We then discuss the relationship between task load and agent network structure. This relates to estabilishing the optimum time when agents should learn about appropriate partners in an actual environment.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Distributed Artificial IntelligenceMultiagent systems

General Terms

Theory

Keywords

Collaboration, Coordination, Organization, Multi-Agent Simulation, Load-Balancing

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

Each agent in a multi-agent systems (MAS) must select the appropriate collaborative partner agent when assigning a task. The choice of an appropriate agent depends on the types of task and the agent's abilities and specialties. However, if there are still multiple candidate agents, the most efficient agent is preferable. Furthermore if deadlines have been set, the issue of performance/efficiency becomes dominant. For e-commerce on the Internet, for example, the selection of efficient agents (in this case, Web and database servers) provides smooth transactions for all customers. This also means that the computational resources provided by the e-commerce company are well-utilized.

The agents selected for collaborations are often memorized by a local agent for a certain period. This stored agent structure constitutes a kind of agent organization. Such an organization can avoid repeated searches for collaborating agents, retains the quality of the results, and enables agents to accurately estimate the finish time of given tasks for real-time requirements[3]. This suggests that the process by which agent teams are organized and the way appropriate partner agents are selected are important issues in designing MAS applications.

However, in an open environment like the Internet, complete information is not available; it is not possible to grasp the global states of the Internet and the complete states of the entire MAS working there. The efficiency of task execution by other agents depends upon their processing speed and load, and the communication bandwidth, traffic, and latency between agents. Hence, agents have to select appropriate (that is, efficient) partners based only on locally available information, which often contains uncertainties. This kind of local decision-making has already been used in Internet operations[2, 8, 9]. For example, Tenbin DNS[9] is an alternative DNS which, if it knows a number of servers that can provide the same content or functions, returns the IP address of the appropriate server for load-balancing and

for selecting the most efficient servers. Tenbin DNS selects servers based on local observations such as RTT (round trip time) using 'ping' or the average throughput between local hosts and target servers. A similar methodology is also used for Web server selection[2] and for route selection in multi-homed network environments[5, 6].

Here, another issue involved with scalability arises: when all agents select partner agents and form organizations based on local data, is the overall performance of the total MAS maximal or reasonable? ('Total' means not the performance of individual agents but the average performance of all agents.). This issue is important because the result is influenced by factors including task load, the agent's (CPU and processing) performance, communication costs, the agent's distribution across the Internet and the strategies of the agent's partner selections. For example, a high-performance agent a is identified as the best partner by many agents, and thus, is likely to be overloaded. This concentration worsens its observed performance data, so many agents will switch to other lower performance agents. Later, a will again be observed as a high-performance agent so they will switch to it again. Of course, these situations may depend on communication costs and the agent's *partner selection strategies (PSS)*. A number of studies have investigated how much the performance of each agent can be improved[2, 9]. From other global view points, for example, Gaston and DesJardins [4] propose an agent organizational network, and investigate what features are required to effectively form teams. To our knowledge however, there has been little research about how agents' local strategies and task loads influence the total performance of MAS and the structure of networks according to learning by individual agents.

The purpose of this paper is to show, using multi-agent simulation, how the total performance of a MAS improves or degrades when all agents statically, adaptively, or collaboratively select partner agents based on their local information. We also illustrate how the network of inter-agent structures (this network is simply called the agent network or hereafter, AN) changes and converges, and how the total performance changes according to the observed data, where an AN node corresponds to an certain agent and the edge between nodes a_1 and a_2 means that a_1 recognizes that a_2 is the appropriate agent to collaborate with. In this paper, we will first show that agents' PSSs affect the total performance and the structure of the agent network and that simple adaptation and collaboration with neighboring agents can effectively improve the total performance. Furthermore, the AN structures are heavily dependent on the frequency of the tasks occurring as well as the PSS. This result suggests that for the best performance the appropriate PSSs depend on the overall task loads.

This paper is organized as follows: In the next section, the objective and settings of our multi-agent simulation are explained. Then, as a reference for subsequent experiments we will show the results of the first simple experiment. Next, we try to improve the number of dropped tasks and response time by changing some performance parameters. We also examine the effects of fluctuation in selecting the best server. Finally, we will discuss some implications for the control and design of MAS derived from these experiments.

2. AGENT SIMULATION

2.1 Objective

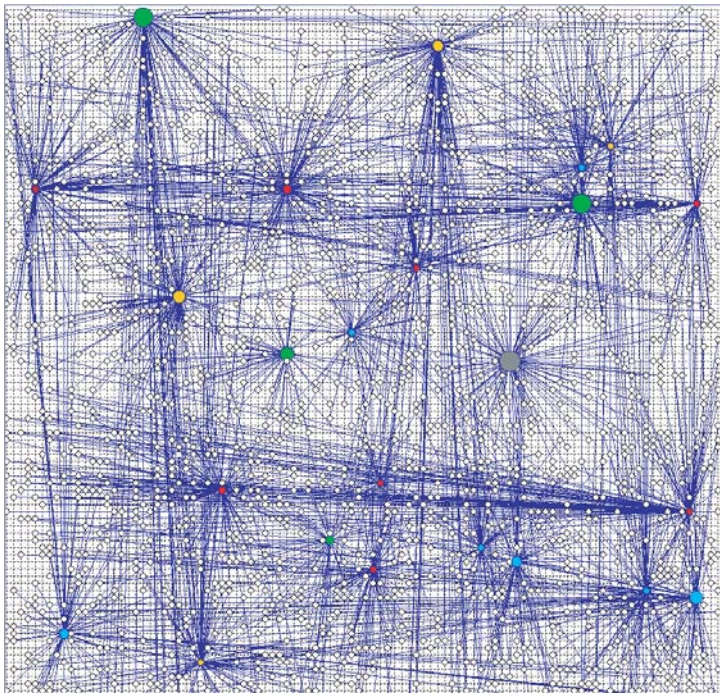
The objective of our simulation is to understand the structural changes of the agent network and to investigate the improvements and limitations of the total performance of the whole MAS when all agents make rational decisions about partner selection. Our simulation model consists of a set of agents $A = \{a_i\}$, and server agents $S = \{s_j\}(\subset A)$, which are able to execute a specific task T. When agent a_i has T, it assigns it to one of the servers that it knows, $S_i(\subset S)$. A PSS corresponds to the method whereby a_i selects s_j from S_i. To clearly understand the relationship between total performance and PSSs, we make the other parameters simple. For this purpose, we assume that all tasks are only of a single type. It is important to note that a server in this paper means an agent that can execute a specific task. Because we only consider a single type of task, the set of servers is static. In general, among the agents in a MAS, multiple tasks request each other, and different agents become servers for other tasks. We do not consider a simple client-server model but rather more flexible multi-agent models in which each agent selects appropriate partner agents depending on the task types from a local viewpoint.

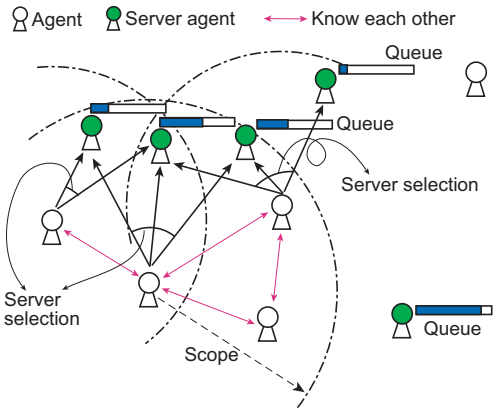
On the actual Internet, one of the observable parameters concerning agents' performance is the response time (rt), which is the length of time between the agent requesting a task and the return of the result. We assume that smaller rt is better. Another parameter for evaluating agents is the number of dropped tasks, which indicates that the requested task is dropped (silently discarded) or refused (with notice), which also is better if it is smaller. Other parameters such as type of CPU and the number of queuing tasks in servers can also been used to evaluate agents, but these parameter values are not usually locally available, so they are not used for the evaluation of servers by agents in our experiments.

Hence, using the observed data, each agent learns which server's rt is expected to be the smallest. In our experiments, agent a_i calculates the average values of the observed response time of the known server s_j (let this value be h_i^j) and (usually) selects the best server, $arg\ \min_{s_j \in S_i} h_i^j$.

To evaluate the total performance of a MAS, we adopt the average value of the observed response time in all agents. This value is denoted by RT. Thus, our experiments demonstrate how RT changes when all agents try to make rt smaller. Another parameter for understanding the total MAS is the drop number, D, which is the total number of dropped tasks. Note that a smaller drop number is better but a drop is observed only when the MAS is overloaded. Therefore, the use of this value is restrictive.

To investigate the structure of relationships between an agent and its selecting server, we introduce the *agent network (AN)* whose nodes are agents or servers and whose links indicate that an agent has determines that a server is the best according to its PSS and local data. For a positive integer γ (= 1 for the default value), called the *selection number*, agent a_i selects γ servers as appropriate servers from S_i, then assigns the given task to one of the selected servers. We look into the *degree* of the server's node in the AN because the degree denotes the level of concentration of the task assignment, that is, how many agents identify that server as appropriate for assigning tasks. The distribution of the servers' degrees expresses the balance or deviation of the overall (and potential) task load. We also investigate how the distribution changes as the agents' observation and





Above: Simulation Environment.
This figure shows the settings of our simulation environment.

Left: A Screen Snapshot of our Simulation.
A small outlined circle corresponds to an agent and a filled circle to a server agent. The sizes of the filled circle express their queue length, so the larger circles are busier. A line between agent and server expresses that when the agent is given a task, the agent assigns it to the connected server with some probability. This snapshot of the simulation is scaled down: gridsize is 100x100 with torus topology, the number of agents is 2000, and the number of servers is 24.

Figure 1: Simulation Environment and Screen Snapshot of our Simulation.

learning progress.

Our simulation can acquire other parameters for understanding the whole MAS performance, such as average values of (1) *communication cost* C (the average RTT between agents and servers), (2) the *queuing time* Q (the average duration that the incoming task is queued for before execution), and (3) *CPU time* E (the average time for executing a task). Although these parameter values are not directly observable on the Internet, they can help to estimate internal AN changes. We assume that $RT = C + Q + E$ holds.

2.2 Simulation Environment

Our simulation (Fig. 1) generates 10000 agents that are randomly placed on points of a 150 x 150 grid plane with a torus topology. Of these, 120 agents are randomly selected to be servers (so the remaining 9880 agents will assign tasks to these servers). We also introduce the Manhattan distance to this grid.

For every tick, tl tasks are generated and given to tl randomly selected agents, where tl is a positive integer. tl is called the *task load*. We can naturally extend these settings for a positive non-integer tl, which means that the average number of statistically generated tasks in every tick is tl. This is denoted by tl task/tick or simply tl T/t, where we assume that 1 tick is 10 msec. Each agent a_i given the task immediately selects a server $s_j \in S_i$ using its PSS and sends the task to s_j. Server s_j processes the committed task and returns the result to a_i. Agent a_i can observe the response time for every task and calculate the average response time h_i^j that will be used in the next PSS.

All servers are assumed to have their own CPU capabilities: each server can process a task in 100 to 500 msec. All of the servers have different capabilities that are randomly assigned *a priori*. Once assigned, these capabilities are invariant. When a task arrives at server s_j, it is immediately executed if s_j has no other tasks. The result is then returned. If s_j has other tasks, the received task is stored in its local queue and the queued tasks are processed in turn. An agent can store 20 tasks in its queue. If a_i already has 20 tasks in its queue, the new task is dropped or refused.

The communication cost (time required to send a task) is assumed to be proportional to the distance between the agent and server and ranges from 10 to 120 msec. Of course, this maximal cost is also a parameter of the simulation environment, although it is not altered in any experiment in this paper, so that we might easily understand the effect of PPS based on local observation. Each agent has its own scope depending on distance (less than 14 in our experiments), so it initially knows and can communicate with all agents and servers within the scope. For all agents to compare servers' response times and to select the best (or better) one, they have to know at least two servers. If an agent knows none or only one server, it asks all known agents for known servers; by doing this, agents can initially know two to fifteen servers.

The experimental results of all our simulations are the average value of three independent experiments derived from a random number using three different seeds. The figures of degree distribution shown in this paper are only for one of the three experiments, but their diagrams are almost identical to the those of the other cases. In these three experiments, the total sum of the capabilities all agents express is that they can theoretically process 4.7 to 5.0 tasks every tick; of course, the actual total capabilities are influenced by the communication cost, the deviation of task allocation, and server distribution in the grid plane.

3. PRELIMINARY EXPERIMENT (EXP. 1)

In the first experiment (Exp. 1), agent a_i selects γ (= 1) servers from S_i using the following PSS.

(P1) a_i selects the server $arg\ \min_{s_j \in S_i} h_i^j$ with a probability of 0.9[1]. If there are multiple servers that have the best h_i^j, one of them is randomly selected.

(P2) Otherwise, a_i selects the server with the probabilistic distribution $Pr(s_j)$,

$$Pr(s_j) = (h_i^j)^{-l} / \sum_{s_k \in S_i} (h_i^k)^{-l} \qquad (E1)^1,$$

where, $l = 3$ in this experiment[1]. In the beginning, the agents have no observed data of known servers. Hence, agent a_i initially sets $h_i^k = 0$ for the known server s_k so that it first selects s_k with no observed data. Note that (P2) introduces *fluctuation* to some extent into the PSS, since an agent may select a server whose h_i^j is not the smallest.

In the cases of tl=1T/t and tl=4T/t, we measure the performance data, RT, C, Q and E, every 20,000 ticks (20K ticks). Fig. 2 shows how these values change over time. The data in Table 1 are the average values of these parameters during a 600K to 800K tick. Fig. 3 (ii) and (iii) illustrate the cumulative degree distribution (CDD) of AN at an 800K tick, and Fig. 3 (i) is the CDD of the initial state. Thus, the CDD gradually changes from (i) to (ii) (the case of 1T/T) or from (i) to (iii) (the case of 4T/t) in Fig. 3. The last column of Table 1 shows the performance data when only (P1) is adopted (so there is no fluctuation in server selection).

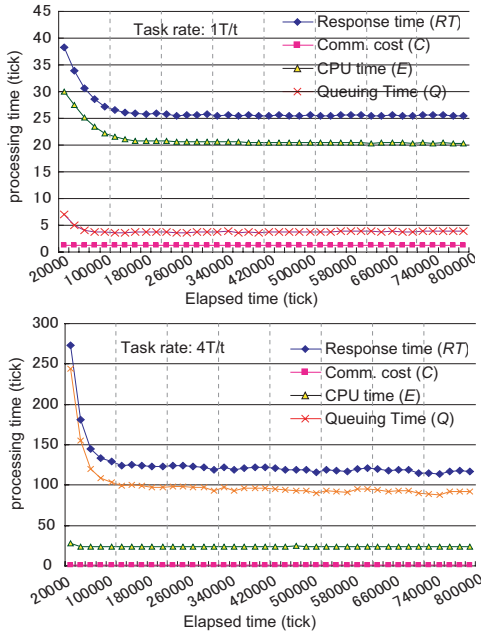


Figure 2: Total performance values (Exp. 1).

[1]We define probability 0.9 and $l = 3$ so that the server selection adequately fluctuates. Although there are a number of other ways to introduce fluctuation, we used (E1) for the first experiment. These values and expression are defined based on some experiments. We will show experimental results elsewhere when these values are changed.

Table 1: Observed performance values in Exp. 1.

Param.	0K (1T/t)	0K (4T/t)	800K (1T/t)	800K (4T/t)	4T/t (no fluct.)
RT	38.29	272.89	25.52	117.11	125.70
C	1.233	1.269	1.295	1.309	1.312
E	30.05	27.41	20.38	24.27	24.01
Q	7.01	244.20	3.85	91.52	100.38
D	0	11180.0	0	622.8	1780.1

The unit of value is a tick except D (number of tasks). 0K means the 'initial state' whose values are observed from 0 to 20K. The value of the columns at 800K are the average values of the parameters observed at every 20K ticks from 600K ticks to 800K ticks.

As indicated in Fig. 2, in this experiment the servers' performance became stable at around 120K ticks because the agents only know 2 to 15 servers, so all of them finished learning their performance. Agents select only local servers (since C hardly changed over time), so the global performance plateaus at an earlier stage. Though the theoretical capability of all servers is higher than 4T/t, approximately 600 tasks are dropped after learning. This implies that the servers' capabilities are underutilized.

The performance data are different depending on whether or not there is fluctuation as shown in Table 1. The fluctuation introduced in these experiments given by (P2) can improve average response time (RT) by 8% when tl = 4T/t. However when tl = 1T/t, this kind of improvement cannot be observed. The details are described later. Note that this experiment is similar to the one of the fixed-load case in [7], although there are many different settings; for example, the number of agents is quite large and communication costs are introduced in ours. The conflict vector in [7] corresponds to the degree of AN in ours; our experiment has more agents so it can show the structure of AN after learning.

In this experiment, the final total performance could have been further improved if all agents had known more servers. However, learning efficiency would have drastically dropped. For example, if all agents knew all 120 servers, they would require at least 1186600 (= $120 * 9880$) tasks to get performance data for all the servers. So in the case of 1T/t, it would take about 1200K ticks. In actuality, however, it would take more than 2400K ticks, because tasks are randomly assigned to agents. In practice, it is not feasible to know and attempt to use all servers on the Internet if the agents are in a massive MAS. This means that a more effective strategy for better knowing servers from a non-local viewpoint is required.

4. PERFORMANCE IMPROVEMENT AND STRUCTURAL CHANGES

We can consider a number of performance measures, but we first focus on the total drop number D in a MAS and try to reduce it. Using local information, we examine the following strategies to reduce D.

(S1) **Best two selection:** Set $\gamma = 2$. All agents select the best two servers every 500 ticks. When a task is given, the agent selects one of two servers using probability (E1) and sends it the task.

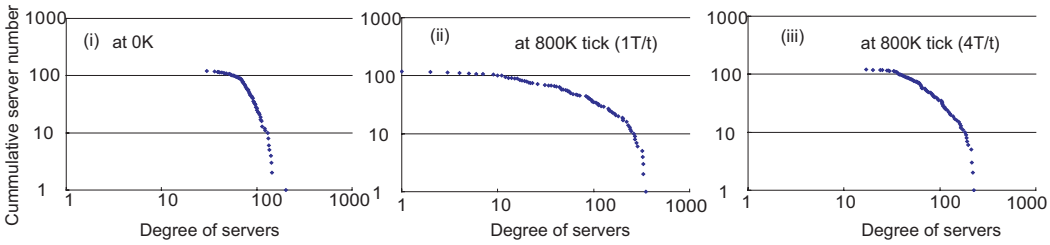


Figure 3: Cumulative degree distribution of servers.
Each point (x, y) indicates that there are y servers whose degrees are more than or equal x.

Table 2: Observed performance values in Exp. 2.

	(S1)	(S2)	(S3)	(S4)
RT	134.08	118.22	153.21	117.00
C	1.305	1.309	1.306	1.307
E	24.81	24.28	24.89	24.36
Q	107.96	92.63	127. 02	91.33
D	729.1	594.5	180.3	420.1

The unit of value is a tick except D. All data are the average values of the corresponding parameters observed every 20K ticks from 600K ticks to 800K ticks. $tl = 4$ T/t.

(S2) **Retransmission:** If their queues are full, the servers do not drop but rather refuse the requested task. The requesting agent resends the task to the server again. Agents will repeat this process five times, but if all five requests are refused, the agent gives up (drops) the task.

(S3) **Alternate selection:** Agents adopt both (S1) and (S2). The main difference is that agents have the top two servers, so after a task is refused, it may be sent to another server.

(S4) **Penalty:** This strategy is the same as (S2) except for introducing the penalty of refusal which is calculated as follows: Add double the RTT to the observed response time. This worsens the rt of the refusing server, so the agent may select another server the next time.

Table 2 shows the results of the second experiment, Exp. 2, whose tl is 4T/t. Comparing Tables 1 and 2, all performance values show only a small change in (S1) and (S2), and especially RT and D get worse in (S1). Even if an agent knows the second best server in (S1), it is probable that the server is the best for a number of other agents, so the drop rate does not decrease. Additionally, the agent requests the second best server, so the value of E (and of Q) also gets worse. Strategy (S2) is not a useful tactic for reducing D; although, after a task is refused, an agent requests it of the same server, there is a high probability that it will be refused again because with 4T/t the server's queue is likely to be still full. Note that if the agent can wait for a longer time D may become smaller, but there is a trade-off between the probability of refusal and the waiting time.

However, strategy (S3) can effectively reduce D. Alternately requesting a task of two servers simply halves the probability of refusal. Instead, RT gets larger; this implies that more tasks are stored in agents' queues, so Q also increases. Finally, strategy (S4) can only improve RT and D a little bit. We expect that this kind of penalty works well when servers are placed non-uniformly and the agents in the area where servers are sparsely settled observe more refusals so they then try to access more distant servers. The figures of cumulative degree distribution for (S1) – (S4) are not shown in this paper, but their characteristics are the same as those in Fig. 3 (iii).

4.1 Collaboration with neighbor agents

The experiments in the previous section aimed at reducing D, but the total response time RT and queuing time Q increased. These strategies kept all agents waiting for a longer time. In this section, we aim to reduce RT and Q.

The performance of the simple server selection strategies used in Exps. 1 and 2 depends on the size of the known servers. Agents that know more servers can choose more capable ones. However, the more servers the agent know, the more inefficient their learning performances are. So, a more effective and efficient strategy for agents to acquire good servers is required.

A practical strategy in the network environment is the recommendation of good servers by neighboring agents. Each agent learns the expected servers' capabilities from their local viewpoint so they can recommend to other agents the best or better server based on their observations. The basic idea is that it is possible for the better servers of neighboring agents to also be good servers for other agents. This strategy works as *collaborative filtering* using server evaluations.

To achieve this collaboration, we add the following PSS, (P3) between strategies (P1) and (P2).

(P3) With a probability of 0.01, an agent a_i randomly selects a known agent, and asks it to recommend a 'good server' s_n. If a_i already knows s_n, this recommendation is ignored. If not, a_i adds s_n to its set of known agents (so $h_i^n = 0$).

An agent that is asked to recommend a server selects a 'good server' using probability distribution (E1). The experimental results using (P1)–(P3) for 1T/t and 4T/t are shown in Table 3 and Fig. 5 (this experiment is referred to as Exp. 3). The cumulative degree distributions of Exp. 3 are illustrated in Fig. 4.

Table 3 shows that both RT and D are improved. This implies that server recommendation is an effective strategy for total MAS performance. Compared with Table 1, Exp. 3's C becomes larger – this indicates that agents assign their tasks to more distant but more efficient servers (E becomes

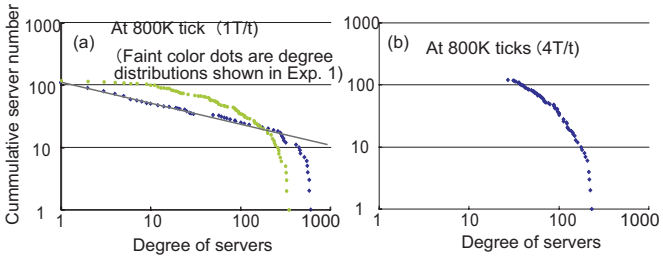


Figure 4: Cumulative degree distribution at 800K (Exp. 3).

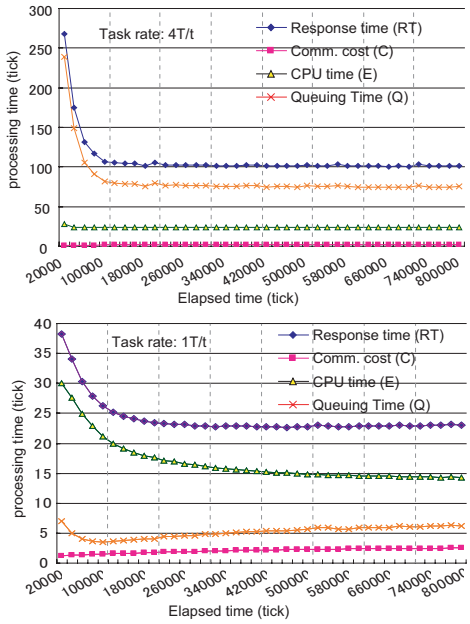


Figure 5: Total performance values (Exp. 3).

slightly smaller). The biggest difference between Exp. 1 and 3 is that the cumulative degree distribution for 1T/t follows the power distribution in Exp. 3, though it follows an exponential distribution in Exp. 1. It seems that the recommendation of good servers works like the preferential attachment in the Barabasi-Albert model[1]. However, there are some differences. For example, in our experiments, (1) all agents re-evaluate the recommended servers from their viewpoints, (2) the server becomes more inefficient if more agents assign their tasks to it, and (3) the communication costs prevent agents from accessing distant servers. So we argue that the server recommendation by agent collaboration (that is, by collaborative filtering) with local evaluation may be another mechanism for expressing the power low in the servers' degree distribution when the workload is low. Note that the cut-off in Fig. 4 (a) appears because of the limited numbers of agents in our simulation setting.

Table 3: Observed performance values in Exp. 3.

	1T/t	4T/t	4T/t with (S3)
RT	22.99	101.33	98.91
C	2.449	2.420	2.420
E	14.43	24.00	24.05
Q	6.11	74.91	72.45
D	0	309.6	0.17

The unit of value is a tick except D (number of tasks). All data are the average values of the corresponding parameters observed every 20K ticks from 600K ticks to 800K ticks.

At 800K ticks, the number of the agents' known servers ranges from 3 to 20 and increases by about 5.5 on average. This number is much smaller than 120, which is the total number of servers in this simulation. Therefore, we can say that the collaborative filtering described here is an effective learning method in our simulation setting.

Additional observations of Fig. 5 and Table 3 are:

- When tl is 1T/t, collaborative filtering produces a task concentration at higher capability servers and thus the number of servers whose degrees are small also increases. This concentration results in highly efficient total performance because the task load is low, which means that Q is still small.

- When tl is 4T/t, there is little difference between the degree distributions of Exp.1 and Exp. 3. However, Tables 1 and 3 express a difference in RT values. Comparing E values, agents did not find high capability servers, but we can understand from the C values that agents allocate their tasks to distant servers. A decrease of Q and D indicates that agents achieve load-balancing across wider regions.

Fig. 5 (b) demonstrates that RT becomes stable at an earlier time, but E, Q, and C still change around 800K ticks when $tl = 1$. This implies that the AN structure constantly changes; the degree distribution gradually changes to express power laws through this structural change.

Finally we want to point out that the collaborative filtering and strategy (S2) in Exp. 2 improves performance for different reasons if we compare Tables 2 and 3. Therefore, we additionally experimented with total performance and strategy by combining them. These results are shown in the final column in Table 3, which expresses the best performance in our experiments.

4.2 Fluctuations and Performance

The probabilistic distribution (E1) in Exp. 1 introduces a small fluctuation into the server selection decision because there is the small possibility of selecting a known server that is not the best. The last column in Table 1 shows the performance data if this fluctuation is excluded (that is, only (P1) is used) when tl is 4T/t. This data show that the fluctuation can reduce RT by 7.3%, Q by 9.7%, and D by 186%. We think that these differences are considerable.

However, this improvement by fluctuation appears only when tl is near four. The data (a) in the two graphs of Fig. 6 illustrate the distributions of $RT' - RT$ and $D' - D$in Exp. 1 over various tl ranging 0.5 to 5.2 T/t, where RT' and D' are the average of the whole response time and the number of total dropped tasks when no fluctuation was introduced (that is, (P2) was not used). The left graph in Fig. 6 shows that when $tl \leq 2.5$ the difference of RT and

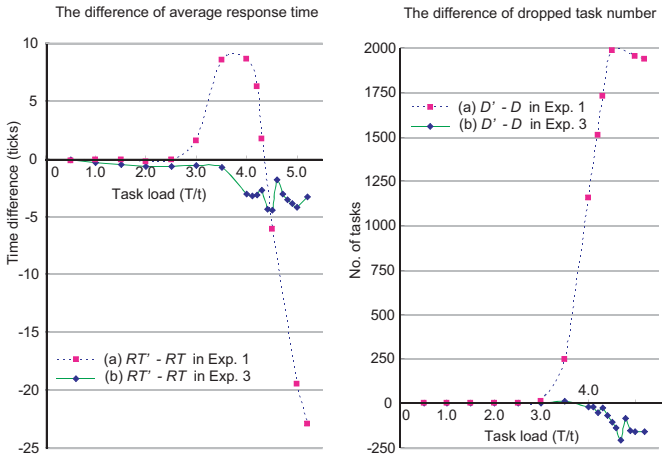


Figure 6: Performance differences (Exp. 4).

These graphs show the difference in the average response times and the number of dropped tasks when fluctuation is introduced and when it is not. Both RT and D are better if they are smaller, so if $RT' - RT$ $(D' - D)$ is positive, the fluctuation positively affects the average of the total response time (the number of dropped tasks).

RT' is almost zero (or more precisely, slightly negative). Then, fluctuation (by P2) can improve the average response time between 2.5 and 4.4, but worsens it when $tl \geq 4.4$. However, D is always better if fluctuation is introduced. We must note again that the theoretical total performance of all servers in our experiments varies from 4.7 to 5.0 and the average value is 4.86.

These figures show that the agents' best selection is not accurate when $2.5 \leq tl \leq 4.4$ in this experiment although agents select, in both cases, the best server within a restricted scope, which is the set of initially given servers. This implies that there is more uncertainty in server selection, in this range of tl. When the total load is low, the agent's PSS without collaborative filtering is relatively accurate, because all servers' queues have enough vacancies so that their observed capabilities are mainly affected by the servers' capabilities. A similar situation also appears when $tl \geq 4.4$, because when all servers' queues are almost full the observed data simply reflects their capabilities. When $2.5 \leq tl \leq 4.4$ however, the observed data are mixed with the capabilities and the queue lengths, which float over time. Therefore, the server observed to be the best may not be the best with higher probability in the next timing of the task assignment. Note that regardless of whether or not fluctuation is introduced, an agent selects servers only from the set of the initially known servers.

This uncertainty is eliminated to some extent when agents employ PSS with collaborative filtering. The graphs (b) in Fig. 6 show $RT' - RT$ and $D' - D$ in Exp. 3. As opposed to the previous case, the fluctuation worsens the total performance and the number of dropped tasks when $tl \geq 4$; this intuitively feels natural. This result implies that the collaborative filtering offers agents a more accurate and scalable decision-making process for selecting appropriate partner agents.

Table 4: Change task rate at 800K tick (Exp. 5).

	1→ 4T/t w/o CF	4→ 1T/t w/o. CF	1→ 4T/t w/ CF	4→ 1T/t w/ CF
Th	180.71	33.99	213.14	32.60
C	1.287	1.312	2.460	2.514
E	22.33	24.30	17.42	24.27
Q	158.09	8.38	193.25	5.81
D	6989	0	23790.7	0

(CF stands for collaborative filtering)

5. DISCUSSION

In real applications, selecting the appropriate agents for task allocation is quite important. We discuss this issue by taking into account our experimental results.

Effect of prior learning

Load-balancing should be achieved when the total load of the MAS is heavy. However, although agents' prior learning to identify the best way to assign specific tasks when the global MAS state is normal (that is, when the load is not heavy), it has little or no effect on performance improvement. One of our findings in both Exp. 1 and 3 was that the AN structures are quite different depending on the task load tl. Especially, Fig. 4 (Exp. 3) shows that the features and their degree distributions are different in the cases of 4T/t and 1T/t; one is exponential and another is a power-law. This implies that the agents' local decision about which server is the best is also affected by the total task load.

In Exp. 5, we switched the task load 1T/t to 4T/t at the 800K-th tick and vice versa; these results are shown in Table 4. Comparing this table and Table 1, the performance values are almost back to the values found in early stages in all cases. So, we can conclude that agents should learn the best partner agent when the MAS is totally overloaded.

Compatibility with conventional programs

Our experiments show that observation and learning by each agent, especially collaborative filtering, play an important role in achieving efficient and effective partner selection. Actually, some systems described in Section 1 such as TenbinDNS and IPv6 route selection decide their preferences based on these kinds of observations without collaborative filtering. This learning-based load-balancing is more effective than static, random, and round-robin-based methods.

Furthermore, strategy (S3) in Exp. 2 can realize more efficient load-balancing with collaborative filtering as shown in Table 3. Although strategy (S3) in Exp. 2 seems to be a natural extension of (S1) and (S2), it is not easy to deploy this function for a number of reasons. First, in real applications, modifying legacy and existing programs is avoided; for example, the previously mentioned approach of load balancing by DNSs and proxies was chosen for this reason. Normal software such as a Web browser makes a query to the local DNS to resolve the IP address from the server's name. The result is then cached for a while (because this function is provided by the OS). Therefore, agents do not change their accessing servers.

The second reason is that agents have to understand the context of the (distributed) processes running on them in order to flexibly change their accessing partners. For exam-

ple, when buying goods using an e-commerce application, if a Web browser were to reconnects to another Web server that provides the same functions during the writing of an address or credit card number, it could cause some inconsistencies. More importantly, it could create an opportunity for user information to be stolen or illegally accessed. Thus, agents cannot change the server even if the server becomes overloaded during an interaction. We assume that (S1) and (S2) can be implemented in actual environments, but that it can only slightly improve the total MAS performance. Therefore, such a strategy (S3) should be implemented in future agent programs.

Rational decision and fluctuation

In general, an agent is required to make rational decisions according to its local knowledge which may contain uncertainty. However, this decision should be made with the consideration of accurate knowledge. Our experimental results regarding the effects of fluctuations indicate that the best selection does not always lead to the best performance. When all agents do not adopt collaborative filtering, it is better that agents sometimes select the second or third best server. However, when they adopt collaborative filtering, their rational decisions usually lead to a better performance.

Our experiments and discussion suggest that collaborative filtering can provide a more accurate view for rational decision-making. Collaboration is indispensable because it can eliminate some of the uncertainties contained in local knowledge. Rational decisions are required in multi-agent design but they must be based on accurate information about non-local information, otherwise the rationality is uncertain.

For more accurate rational decision-making, information about network and other agents' states is important. As shown in Exp. 3, collaborative filtering can achieve effective PSS. This suggests that MAS design should take into account the structure and (dynamic) performance of the Internet. Looking again at the example of DNS-based load-balancing, collaborations with other DNSes in the same autonomous system (AS) or neighboring AS's must be useful. However, Exp. 5 implies that this kind of learning through collaborative filtering in mismatched situations makes the performance worse. In this case, it is better that agents know network traffic and other agents' loads. To form appropriate MAS organizations for problem solving, we believe that MAS infrastructure that has the view of Inter- and Intra-AS structures (that is, routing) and (predicted) traffic, is necessary for future MAS design[10, 11].

6. CONCLUSION

In this paper, we investigated whether agents' local strategies for selecting partner agents that are committed a specific task affect the total performance of an entire MAS. In particular, we focused on the structure of agent networks that vary according to the task load. Then, we discussed a number of findings from the experimental results: (1) learning partners by observation must be derived from the observed data gathered in the same/similar situation, (2) for more precise learning, each agent has to take into account the states of network and other agents, and (3) rational decisions by agents do not provide the best performance when the task load is at or near the critical point for total MAS capabilities.

Only a few years ago, the communications bandwidth of the Internet was narrow. Hence, to provide effective and efficient access to the (Web) servers, it was important to place them at the sites near customers or at major network ISPs. However, recent communication technologies can provide a 1M to 100M line to each home, and the communication bottleneck tends to move to the server side, because they must process a large number of requested tasks. In such an environment, it is necessary for an agent to select high-performance partners rather than closer partners. In designing a MAS, we must consider how each agent finds other agents to assign tasks. Each agent's decision, based on local information, can be improved by careful local observation and learning. This can realize more efficient MAS operation. However, our experiments show that the simple local observation and learning may not be scalable when all agents do the same thing. Collaboration, such as collaborative filtering, plays an important role in achieving more scalable load-balancing.

7. REFERENCES

[1] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[2] V. Cardellini, M. Colajanni, and P. S. Yu. Dynamic load balancing on web-server systems. *IEEE Internet Computing*, 3(3):28–39, 1999.

[3] K. M. Carley and L. Gasser. Computational organization theory. In G. Weiss, editor, *Multiagent Systems*, pages 229–330. MIT press, 1999.

[4] M. E. Gaston and M. desJardins. Agent-organized networks for dynamic team formation. In *Proceedings of AAMAS2005*, pages 230–237, 2005.

[5] C. Launois, O. Bonaventure, and M. Lobelle. The NAROS approach for IPv6 multihoming with traffic engineering. In *Proc. of Int. Workshop on Quality of Future Internet Services (QoFIS2003)*, 2003.

[6] S. Lee, Z.-L. Zhang, and S. Nelakuditi. Exploiting as hierarchy for scalable route selection in multi-homed stub networks. In *Proc. of Internet Management Conference (IMC'04)*, pages 294–299, 2004.

[7] A. Schaerf, Y. Shoham, and M. Tennenholtz. Adaptive load balancing: A study in multi-agent learning. *Journal of Artificial Intelligence Research*, 2:475–500, 1995.

[8] R. Schemer. lbnamed: A load-balancing nameserver written in perl. In *Proc. of Large Installation System Administration Conference*, pages 294–299, 1995.

[9] T. Shimokawa, N. Yoshida, and K. Ushijima. Flexible server selection using DNS. In *Proc. of Int. Workshop on Internet 2000 (held at DCS2000)*, pages A76–A81, 2000.

[10] T. Sugawara, O. Akashi, S. Kurihara, and S. Sato. An Organization-Related Information Maintenance Component. In *Proceedings of ICMAS2000*, pages 443–444, July 2000.

[11] A. Terauchi, K. Fukuda, O. Akashi, M. Maruyama, T. Hirotsu, T. Sugawara, and S. Kurihara. ARTISTE: Agent Organization Management System for Multi-Agent Systems. In *Proc. of PRIMA2005*, 2005.

Effect of Joining Decisions on Peer Clusters

Stéphane Airiau
Mathematical & Computer Sciences Department
600 South College avenue
Tulsa, OK 74104, USA
stephane@utulsa.edu

Sandip Sen
Mathematical & Computer Sciences Department
600 South College avenue
Tulsa, OK 74104, USA
sandip@utulsa.edu

Prithviraj Dasgupta
Computer Science Department University of Nebraska Omaha, NE 68182, USA
pdasgupta@mail.unomaha.edu

ABSTRACT

Super-peer networks have been proposed to address the issue of search latency and scalability in traditional peer-to-peer (P2P) networks. In a super-peer network, instead of having a fully distributed systems of peer nodes with similar or comparable capabilities, some nodes that possess considerable computing power and resources are designated as super-peers. Each super-peer acts as a server for multiple client peers under it. This hierarchical structure of a super-peer network improves the performance of a super-peer network over traditional P2P networks by handling most search queries between the few super-peer nodes, thereby reducing overall network traffic and improving search latency. In this paper, we address the problem of mutual selection by super-peers and client peers. In particular, we evaluate alternative decision functions used by super-peers to allow new client peers to join the cluster of clients under it. We experiment with peers with known resources and demands. By formally representing and reasoning with capability and query distributions, we develop peer-selection functions that either promote concentration or diversification of capabilities within a cluster. We evaluate the effectiveness of these different selection functions for different environments where peer capabilities are aligned or are independent of their queries. We offer insight and analysis on the effects on inter and intra-peer bandwidth consumption which will allow a super-peer to adopt appropriate peer-selection functions given their expectations about the environment.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Design

Keywords

Super-peer, Peer-to-peer, Formation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

Current research in peer-to-peer (P2P) networks mainly focuses on the capacity of the network to provide efficient search capabilities, to treat queries in a timely and accurate fashion, and to maximize the bandwidth to transfer data between peers. Super-peer networks [3, 5, 13] have been proposed to improve the search latency and scalability of traditional P2P systems. Instead of having a fully distributed system of peer nodes with commensurate computing capabilities and resources, a super-peer network comprises of a hierarchical arrangement of nodes with different computational capabilities. A super-peer network is a two-tiered structure: one level consists of super-peer nodes that have considerable computing capability and resources, with each super-peer managing operations such as searching and query forwarding for multiple client peers under it. The benefits of this architecture, compared to a fully decentralized approach, is an improvement in search latency and scalability as most queries are routed between the few super-peers or between client peers under one super-peer. In particular, a peer no longer needs to handle the network traffic associated with every search query. A super-peer, however, can become a bottleneck for its clients: it handles incoming and outgoing queries on their behalf, which requires maintenance of an accurate description of all clients.

Under the assumption that all peers are contributing to the community, [13] gives advice for building efficient super-peer networks. An open problem here is the choice of a super-peer when a new peer joins the network. A super-peer can accept or decline a join request from a new peer. In [5], the design of the network corresponds to ontologies. The idea is to cluster peers falling in the same category so as to make search more efficient in a semantic peer-to-peer network. As a result, super-peers will have similar clients. For example, it is possible to create a community of super-peers hosting computer science related information. Each super-peer could correspond to an area of computer science, say language, operating system, artificial intelligence (AI), etc. and the peers of the super-peer corresponding to AI could host information about agents, planning, Bayesian inference, etc. Depending on the need or the knowledge of the new peer, a similarity measure or the use of an ontology can dictate which super-peer the new peer should register with. [5] prescribes a structured network organization such as a hypercube that determines the topology of client peers under a super-peer. However, the problem of network organization becomes challenging and difficult to design a-priori if we consider unstructured P2P networks. Hence, we are interested in studying decision functions that super-peers and peers can use to mutually select each other and the effects of such decision functions on peer cluster composition.

Consider the case where a super-peer can answer a query from one of its clients by using its other clients. There may not be any

need to send a query to other super-peers. Hence, in addition to not having to coordinate search with other super-peers, answering a query by another peer in the cluster decreases the traffic between super-peers. This phenomenon can be highlighted by considering super-peer networks where each cluster is quasi-self-sufficient: each query being almost always answered by peers from the same cluster. The research question is how to form clusters of clients to significantly decrease the traffic between clusters. Should a super-peer seek heterogeneity of interest and capabilities of its peers, or should it attempt to build a community of peers having similar interests and capabilities.

In this paper, we want to dynamically build the network of super-peers from a fully distributed network. We want to ensure that peers are contributing to the community. To this end, we use a reciprocity based mechanism to promote collaboration between the super-peers. We use a cost function to capture the preference of intra-cluster data transfer over inter-cluster data transfer. We consider that peers are defined both by their interests, which are used to generate queries, and their knowledge or competencies, that are used to answer queries. We believe that in the real world, knowledge and competencies are often aligned: interest in a subject leads to an increase knowledge in that topic. In such situations, peers with similar interests should be able to sustain self-sufficient communities, needing the help of other clusters only rarely. On the other hand, if interest and knowledge are not aligned, it might be preferable to create heterogeneous groups where peers have complementary knowledge and interests. We want to study how the joining decisions affects cluster compositions and the total cost of answering queries. In particular, we consider both environments where agents have similar interest and knowledge, and environments where interest and knowledge are not correlated.

2. RELATED WORK

P2P systems have become an area of active research and development because of the popularity of online resource sharing services such as Freenet [1], Gnutella [2], Napster [4] and SETI@home [9]. Resource management in P2P networks has been an important and challenging issue for researchers. The most common techniques for P2P resource management include structured P2P networks that employ distributed hash tables (DHT). In DHTs, resources are strategically placed on nodes to improve resource availability and enable rapid lookup [7, 10, 14]. In DHTs each node and resource is associated with a key computed from a hash function. A mathematical function [6, 11] is then used to strategically place resources in different nodes to preserve the network topology and balance the network load throughout the system. However, DHTs require additional overhead in the form of updates to local hash tables within a node when nodes and resources join or leave the network, and, forwarding the updates to neighboring nodes.

Yet another mechanism for P2P resource management is super-peer networks [12, 13] that uses a tiered network structure within an unstructured P2P network. In a super-peer network, some nodes act as super-peers or managers that supervise and coordinate the operation of several peers or clients under them. Super-peers maintain meta-information about peers supervised by them. A super-peer can interact with the peers its supervises and with other super-peers to route search queries and implement load balancing algorithms. In super-peer networks, a peer wishing to search for a resource contacts the super-peer supervising it with the search request. The super-peer first searches for the resource within other peers supervised by itself. If the resource is not found within its supervised peers, the super-peer directs the search query to other super-peers. Each super-peer that receives a search request from another super-peer searches for the resources within its supervised peers and responds to the super-peer that initiated the query at the super-peer level if the resource is found. The super-peer that initiated the super-peer level query then forwards the resource-found response to the peer under it that originated the query. In [5], the super-peer framework has been implemented within a structured P2P network. DHT-based algorithms are used to determine the network topology and resource placement within the network. Database schema based techniques are also used to organize the content on different peers. In [3], a protocol for dynamically updating the topology of a super-peer network is described. Super-peers exchange meta-information about peers with each other to reconfigure the network and achieve load balancing. In contrast to these research, our paper describes mechanisms that can be used by peers and super-peers for mutual selection within an unstructured P2P network.

3. SUPER-PEER NETWORK MODEL

In this section, we describe our model of the peer-to-peer system and the query protocol. We assume a fully connected network of super-peers, which allows a super-peer to be contacted directly, i.e., a query can be targeted to a precise set of peers. Each super-peer manages a cluster of client peers.

Each peer is defined by its interests and its competence in providing information to other peers. The interests are used to generate queries, and the competencies are used to answer queries. We will investigate two kinds of environments: one where competencies and interests are the same, the other where they are not correlated. Each resource is indexed by a set of interests or keywords and peers can query for a resource based on its associated interest set. Interests are represented as words (character strings). We have used a hash function to map the different interests to a natural number in the range $\mathcal{S} = [0..dim]$. Every peer uses the same hash function to ensure uniformity in the interpretation of interests across the network. This allows us to model the competence and interest on a one-dimensional line. The interests of a peer are represented by a probability distribution function $\mathcal{I}$ over $\mathcal{S}$. Peaks on $\mathcal{I}$ represents the main interests of a peer. The competence of the peer is represented by a function $\mathcal{C}$ which, given a point q in $\mathcal{S}$, output the probability $\mathcal{S}(q)$ of answering a query for that input. We have chosen this approach to produce a static description of the peers' interests and capabilities that facilitates analysis and experimentation. This would not be possible if peer capabilities were represented for example by a dynamic database of resources.

A super-peer has the responsibility to handle queries from its clients and other super-peers. It maintains a balance of help over the past interactions with other super-peers, and with its own peers. Each time a peer answers a query from another peer, it incurs a cost proportional to the amount of data transferred. We assume that transfer of data within a cluster is cheaper than transferring data between cluster. In order to promote contribution of other super-peers, the balance of cost is used in a probability-based mechanism to make the decision [8] of answering or not answering queries from other super-peers. A super-peer wants to minimize the cost incurred by its own clients by promoting intra-cluster communication. If a super-peer or one of its peers does not collaborate sufficiently, the super-peer can decide to block their future queries until the balance improves.

We address the dynamic formation and update of peer clusters by evaluating alternative joining decision mechanisms that are function of the interest and competence of the new peer and the existing clusters. We require a new peer to reveal an estimate of its interests and competencies. Because of the trust mechanism used by the super-peer to track its actual performance, there is a disincentive to

misrepresenting one's competence or interest. Though we assume that peers can generate their competence and interest vector, we note that this may not be a simple task. In addition, we assume that the super-peer is also a peer, and it can ask as well as answer queries.

3.1 Cost and Reciprocity Framework

The cost metric is a function of the volume of data transferred and whether the communication is intra or inter clusters. We use two rates: c_l for intra-cluster communication and c_w for inter-cluster communication. The difference in cost models a search cost incurred by the super-peer when it needs to deal with the other super-peers.

Each super-peer records the past interactions of help with the other super-peers. For a super-peer i, a super-peer j maintains:

- credit $c(i,j)$, i.e., cost of the help received by j from i.
- debit $d(i,j)$, i.e., cost of the help provided by j to i.
- balance $b(i,j) = d(i,j) - c(i,j)$

To determine whether to answer a query q from super-peer i, super-peer j peer first obtain the cost c_q of answering q using its knowledge about the capabilities of its client peers. It then calculates the probability of responding to the query as

$$P(i,j,q) = \frac{1}{1 + e^{\frac{c_q - b(i,j) - c_0}{\tau}}},$$

where c_0 is the initial inclination to help and τ controls the shape of the probability function. The super-peer will sample this probability to decide whether to help or not the other peer. If c_0 is large, the super peer is more inclined to help other super-peers. In Figure 1(a) and 1(b), we present the probability to answer a query with respect to its cost. In Figure 1(a), we study the effect of c_0 on the probability function. The higher the cost, the less likely it is to provide an answer. In addition, the higher the value of c_0, the more likely it is to answer a query. In Figure 1(b) we vary the value of τ for a fixed value of c_0: large value of τ make the probability function to answer a query quasi-linear in the cost, when small values of τ gives a logistic shape to the function: high probability of answering a query for low cost, low probability for high cost.

3.2 Query generation and Response Protocol

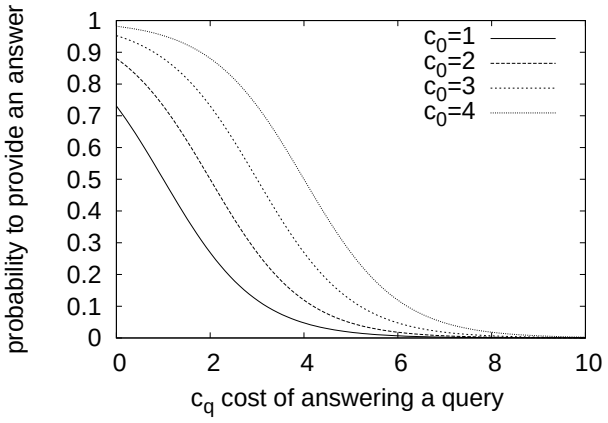
The following describes the different aspects of the query generation and response process:

Query generation: A query corresponds to one point q in $\mathcal{S}$, the interest space. A peer generates a query by sampling its probability distribution $\mathcal{I}$. The message containing the query also contains the address of the requesting peer and its super-peer.

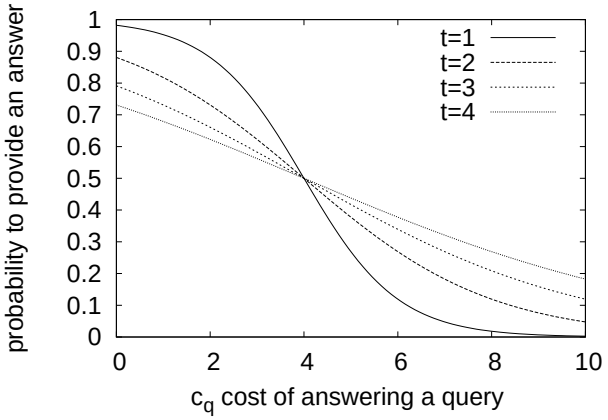
Query-answering capability: The client has an answer to q with a probability $\mathcal{C}(q)$, and do not have an answer with probability $1 - \mathcal{C}(q)$.

Replying to a query q: A super-peer answers a query by sending a message, r_a containing its address, the address of one of its clients, a, that will answer q, and the associated cost. The clients are then responsible for opening a connection between themselves and transfer the data.

Protocol: The protocol for the interaction of the source of a query and the responder is as follows:



(a) Variation of c_0.



(b) Variation of τ.

Figure 1: Influence of c_0 and τ on the probability to help.

1. an initiator peer P_q generates a query q and sends it to its super-peer $SP(P_q)$.
2. $SP(P_q)$ checks its database to find if one of its clients can answer q. We consider that a super-peer has accurate knowledge of the capabilities of its client peers. In our model, this is implemented by $SP(P_q)$ sampling the probability $\mathcal{C}_i(q)$ to determine whether the peer i can answer the query q or not. If one of the clients under $SP(P_q)$ can answer the query, the query is forwarded to that client. If more than one client can answer q, the super-peer chooses the one with the highest balance (download minus upload volume). This allows the super-peer to distribute the load of query answering between its client peers and maintain the satisfaction level of peers.
3. If the clients of $SP(P_q)$ cannot answer the query, $SP(P_q)$ broadcasts the query to other super-peers. When another super-peer SP_a receives a query from $SP(P_q)$ it performs the following steps:
   (a) SP_a determine whether one of its client peers can an-

swer the query. Again, this is implemented by sampling the probability $C_i(q)$ for each of its peer i. If some peers can answer the query, SP_a picks the peer P_a with the highest balance as described above. If no client can answer the query, then SP_a cannot answer the query.

(b) If some client can respond to the query, SP_a uses the reciprocity framework (Section 3.1) to decide whether or not to provide the answer.

(c) If SP_a decides not to answer, an empty message is sent.

(d) If SP_a decides to help, it forwards the query to P_a, and sends a reply r_a to $SP(P_q)$. The reply message contains the address of the peer that will answer the query and the associated cost. At this time, SP_q may or may not use this answer, hence SP_a does not update its balance of yet at this time, it will wait a notification that its peer has provided an answer (see step 4)

$SP(P_q)$ gets all the reply messages, r_a, from all the super-peers. It chooses to use the answer provided by the super-peer SP_a with the highest balance, i.e., it picks the super-peers that owes the most. $SP(P_q)$ updates its balance with SP_a and forwards the reply message to P_q.

4. When P_q receives the reply, it directly contacts the answering peer P_a and transfers the data. At the end of the transfer, P_a notifies its super-peer about the transaction so that SP_a can update its balance with $SP(P_q)$.

3.3 Mechanisms for cluster formation

We assume the existence of a network of super-peers. The peer nodes enter the system one at a time and are assigned to one of the super-peers by the process described below. Once assigned, a peer-node does not change its super-peer.

3.3.1 Random peer assignment

As a baseline scheme for comparison we consider a random assignment of new peers to any of the super-peers.

3.3.2 Negotiated peer assignment

We consider three negotiated mechanisms. In each of these methods new peers negotiate with existing super-peers and final assignment is made by mutual selection.

When a peer seeks to join a cluster, it sends its capability and interest information to all super-peers. The super-peers estimate the "usefulness" of having this new peer in their cluster and reply with the estimate. The peer greedily chooses to join the super-peer that provided the best estimate. To measure this "usefulness", we consider three metrics:

Competence alignment: By using this metric, a super-peer seeks peers that have similar competence, which improves the probability of answering queries on particular topics. The super-peer builds the aggregate competence vector of its cluster of client peer nodes where the aggregate competence of a cluster for a given domain is the probability that at least one peer answers the query. Consider that a super-peer has n client peers, and let $c_{i,j}$ denote the probability of the ith peer to answer a query for the jth domain in the competence space $\mathcal{S}$. The probability for at least one peer to answer the query is

$$c_j = 1 - \Pi_{i=1}^{n}(1 - c_{i,j}), j \in [1..dim].$$

The metric returned is the Euclidean distance between the aggregate competence of the cluster and the competence of the entering peer. The smaller the distance, the better the offer made to the peer.

Competence diversity: With this metric, a super-peer seeks to form a heterogeneous group able to answer the most diverse range of queries. For example, a peer who does not bring any new expertise to a cluster is of little interest. The super-peer computes the probability of answering a query about each domain in $\mathcal{S}$ in the absence or presence of the requesting peer. The difference is the measure of the impact of the peer on the group. The "usefulness" metric sent to the requesting peer is:

$$\Delta = \frac{1}{dim}\sum_{j=1}^{dim}\left(c_j^+ - c_j\right),$$

where dim is the dimension of $\mathcal{S}$, c_j (resp c_j^+) is the aggregate probability of the cluster (resp the cluster and the peer) to answer a query about the jth domain in $\mathcal{S}$ (we assume a uniform distribution of queries over $\mathcal{S}$). A super-peer seeks peers with higher Δ value. The peer chooses the super-peer with the largest Δ.

Competence and interest complementarity: By using this metric, a super-peer seeks peers that can answer queries from the current cluster and vice versa. When this metric is used, peers and super-peers use a two-step interaction protocol: the super-peer will first evaluate the ability of the peer to answer queries from the cluster's member. If this likelihood is above a threshold, the super-peer will send the likelihood for cluster's members to answer a query from the new peer. We now present more details of this process. First the peer sends its competence vector so that the super-peer can estimate the potential of this peer to answer a query issued by current cluster members. The super-peer invites a peer only if it can bring sufficient new expertise to the cluster. If the probability of answering queries from its cluster exceeds a fixed threshold ϵ_u, the super-peer responds favorably to the peer. The peer replies to interested super-peers and sends them its interest vector. The super-peers return the probability of the cluster to answer a query issued by the new peer. The entering peer chooses the super-peer with the highest likelihood of answering its query. Let q denote the interest probability distribution and let c denote a competence vector. The probability that a peer or cluster with a competence c can answer a query from a peer or cluster of interest q is $\sum_{i=1}^{d} q_i c_i$. In a nutshell, the super-peer accepts beneficial peers, and the peer joins the most promising super-peer, i.e. one that is most likely to able to answer its queries.

4. EXPERIMENTAL RESULTS

To evaluate the relative effectiveness of above mentioned cluster formation mechanisms, we simulated different environmental configurations. We generated the interest and the competence of the peers assuming the existence of a preponderant domain of competence and a preponderant domain of interest, which need not be identical. When they are identical , i.e., competence and interest are aligned, we have peers knowledgeable about a particular topic and wanting to know more about that topic. When the preponderant topics are different, i.e., competence and interest are not aligned, we have peers that have knowledge in a domain and wanting to learn about a different domain. In Figure 2, we present a typical competence vector of a peer. The algorithm for generating a competence vector for a peer is presented in Algorithm 1. The interest vector is generated in a similar way, and is normalized to meet the requirements of a probability distribution. We have experimented with two types of peers: when interest and competence are aligned and when they are not correlated. In all experiments we ensure that the preponderant competence and interest are drawn from a uniform distribution.

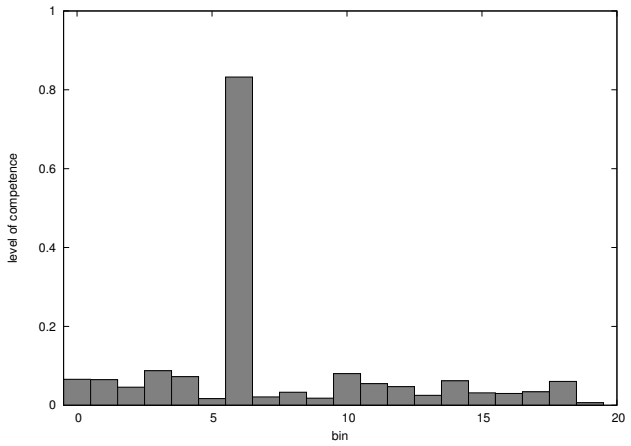


Figure 2: Example of the competence vector of one peer.

Algorithm 1 Form competence vector c for peer p.

p_{high}, p_{low} and v are given
for $i = 1$ to d **do**
  $\epsilon \leftarrow$ normal(0,1)
  if main interest of p is i **then**
    $c_i \leftarrow p_{high} + \epsilon v$
  else
    $c_i \leftarrow p_{low} + \epsilon v$

4.1 Settings

The results presented in this section are averaged over 20 different assignments of the peers to the clusters (the order of introduction of the peers is different). For each assignment, we ran the simulation 5 times, varying the order of the peers asking queries. Each simulation consists of generating 10,000 queries. For each query, a peer is chosen randomly, and its interest distribution is sampled to generate the query type. There is a fixed number of super-peers, one for each domain, and the number of peers varies between 20 and 300.

4.2 Influence of the cost difference between intra and inter cluster communication

We study first the influence of the cost difference between intra and inter cluster communication on the environment where interest and competence are aligned. For these experiments, the reciprocity framework is initialized with $\tau = 1$ and $c_0 = c_{inter}$, i.e. initially, the super-peer is inclined to answer one query issued from another super-peer with a probability of 0.5. In Figure 3, we present results from experiments with a system of 500 peers, 50 super-peers and 50 different domains. The cost for communication within the group is fixed to 1.0 per query.

Since for a given peer, the preponderant competence and interest domains are the same, the optimal assignment occurs when peers with the same preponderant domain are grouped in the same cluster. They are likely to answer most queries of other members of the cluster, thus minimizing total cost of answering queries. Requesting the help of peers from other clusters is needed when a query is outside the preponderant domain, which is much less frequent. Under these conditions, the join function promoting complementarity between peers and the join method based on the alignment of the competence produce best performance. We noticed that as the size of a cluster increases, the cost increases slightly. Even if all the cluster's peers have the same main expertise, the probability of answering other types of queries also increases, which may "mislead" the assignment of some peers entering later. These peers cause the performance to drop, i.e. the cost to increase; since the query of these late entering peers is less likely to be answered by the cluster's member. Surprisingly, promoting the diversity of competence in the cluster performs better than random assignment. The use of a diversity based mechanism should be costly since it is unlikely to have two peers with the same interest/competence coexisting in the same cluster. The relative better performance compared to random is due to the existence of few larger clusters that are more capable of being self-sufficient.

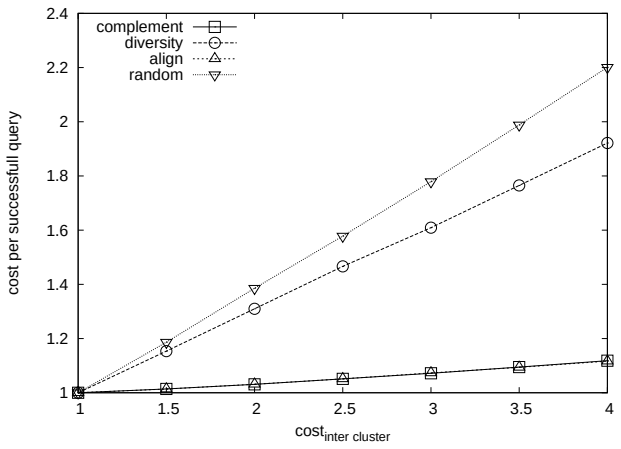


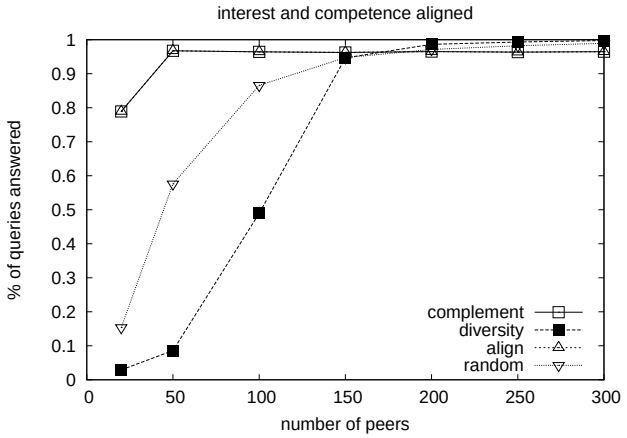
Figure 3: Influence of the cost ratio between inter and intra cluster communication on average cost of answering a query with different cluster formation mechanisms.

4.3 Influence of the number of peers

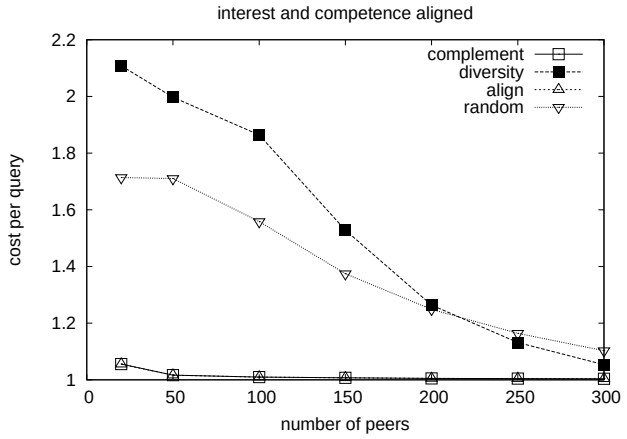
For the rest of the study, we fixed the ratio of the cost of inter and intra-cluster communication to 2.5 where the cost to communicate one answer within a cluster is set to 1.0. In the next set of experiments, we fixed the dimension of the interest space and the number of super-peers to 10. We study the influence of the number of peers in the system both in environment where competence and interest are aligned and when these are not correlated. Our first metric is the average cost of a query. If it is 1.0, all the queries are answered within the cluster. The second metric is the percentage of queries answered (see Section 3.2). A query may not be answered for multiple reasons. First, when a peer p answers a query about domain $i \in \mathcal{S}$, the distribution $\mathcal{C}(i)$ is sampled. Even if i is the domain of expertise of p, it may not get the answer and no one else may be able to answer. In addition, a peer might be capable of answering the query, but the super-peer may block the query because the requesting peer has a low balance.

The results when the competence and interest of a peer are aligned are presented in Figure 4. The complement-based and the alignment-based join functions produce clusters of peers with similar expertise and are performing the best in this environment (the corresponding curves are overlapping). As peers generate queries in their area of interest, other peers in the same group can answer these queries with a high probability and hence they do not require help from other super-peers, keeping the cost low. For a small number of peers in the system, the join function based on diversity is performing worse than random assignment. This is because all the peers in the cluster have different competence/interest, and they have a

small probability of answering queries from the other peers of the cluster. When the number of peers is high, each group contains sufficient number of experts of each type in each cluster to answer queries issued by any peer and hence the performance of clusters generated by all mechanisms become equivalent. This is shown both by an increase in the success rate of answering queries and a corresponding decrease in the cost per query.



(a) Success Rate

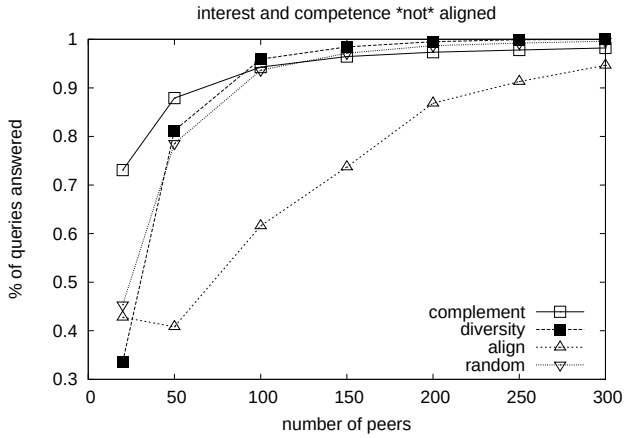


(b) Cost

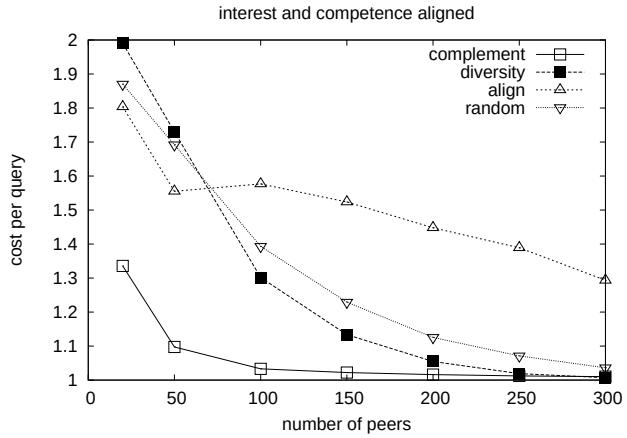
Figure 4: Influence of the number of peers when interest and competence are aligned.

When the preponderant competence and interest of a peer are not correlated, the join function based on complementarity is still performing much better with a small number of peers in the system (see Figure 5). The first significant difference with the previous scenario is that the alignment-based join method is performing much poorer: as peers with the same main expertise do not necessarily have the same main interest, a cluster made of peers with the same main expertise will require the help from many other clusters. The second main difference is that the diversity-based join function takes less number of peers in the system to perform as well as any other scheme. The promotion of diversity in the cluster formation increases the probability of answering any kind of query by another member of the same cluster, which is the key requirement here.

From the above scenarios, the main conclusion to be drawn is that the join method based on complementarity is performing as well as any other selection method over all scenarios. This mechanism appears robust to scale and to competency-interest alignment of peers. The inherent reciprocal benefit consideration underlying this cluster formation scheme enables such clusters to be largely "self-sufficient" in most environments, thus increasing query answering rate and reducing the cost for answering queries.



(a) Success Rate



(b) Cost

Figure 5: Influence in the number of peers when interest and competence are not aligned.

5. CONCLUSIONS

We have investigated the effects of different join-decision-functions on the performance of super-peer networks. Super-peers are responsible to find other peers which can provide an answer to a query, either by using peers from its pool of clients, or by requesting help from other super-peers. Super-peers use a reciprocity mechanism to ensure that there are no free-riders in the system. Each super-peer also ensures that all its client peers are contributing by enforcing load balancing within its cluster of client peers.

We use a probability function to model the competence and probability distributions to model the interest of a peer. Under these conditions, we found out that forming peers based on complementarity between a new peer and a cluster reduces cost of answering and increases success. However, this join method, unlike the others, is a two-step process. Looking for diversity to form a cluster can also be beneficial when there is no correlation between the competence and the interest of a peer. However, when interest and query is aligned, alignment of competence in a cluster is preferable to competence diversity. As the number of peers in each cluster increases, however, the performance difference between the different cluster formation mechanisms monotonically decreases.

We plan to study the effect of the relative ratio of the number of super-peers to the number of capability and interest types. In this paper, agents do not change clusters; we plan to investigate dynamic cluster dissolution and reorganization schemes. Another limitation in the current work is the static interest and capability vectors for peers. We believe that in a number of domains both capabilities and interests of peers can vary over time. That dynamism presents a significant challenge to developing adaptive schemes that will continue to maintain the performance of peer clusters. We will investigate predictive cluster selection schemes to address this critical issue.

Acknowledgment: US National Science Foundation award IIS-0209208 partially supported this work.

6. REFERENCES

[1] Freenet. URL http://www.freeproject.org.

[2] Gnutella. URL http://www.gnutella.com.

[3] A. Montresor. A robust protocol for building superpeer overlay topologies. In *Proceedings of the 4th International Conference on Peer-to-Peer Computing*, August 2004.

[4] Napster. URL http://www.napster.com.

[5] W. Nejdl, M. Wolpers, W. Siberski, C. Schmidt, M. Schlosser, I. Brunkhorst, and A. Löser. Super-peer-based routing and clustering strategies for rdf-based peer-to-peer networks. In *Proceedings of WWW2003*, 2003.

[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, pages 161–172, 2001.

[7] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings ofthe IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.

[8] S. Sen. Believing others: Pros and cons. *Artificial Intelligence*, 142(2):179–203, December 2002.

[9] Seti. URL http:// setiathome.ssl.berkeley.edu.

[10] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM Conference*, 2001.

[11] C. Tang, Z. Xu, and M. Mahalingam. Peersearch: Efficient information retrieval in peer-peer networks. Technical Report HPL-2002-198, Hewlett-Packard Labs, 2002.

[12] B. Yang and H. Garcia-Molina. Improving search inpeer-to-peer networks. In *Proceedings of the 22ndInternational Conference on Distributed Computing Systems (ICDCS'02)*, pages 5–14, July 2002.

[13] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proceedings of the Nineteenth International Conference on Data Engineering*, pages 49–62, 2003.

[14] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault resilient wide-area location and routing. Technical Report UCB CSD-01-1141, University of California, Berkeley, 2001.

History-based Traffic Control

Gabriel Balan
George Mason University
Fairfax, VA (USA)
gbalan@cs.gmu.edu

Sean Luke
George Mason University
Fairfax, VA (USA)
sean@cs.gmu.edu

ABSTRACT

What if traffic lights gave you a break after you've spent a long time waiting in traffic elsewhere? In this paper we examine a variety of multi-agent traffic light controllers which consider vehicles' past stopped-at-red histories. For example, a controller might distribute credits to cars as they wait and award the green light to lanes with the most credits, allowing cars to keep the credits they accumulate during travel. Such history-based controllers are intended to provide a kind of global fairness, reducing the variance in mean time spent waiting at lights during trips. We compare these controllers against other multi-agent controllers which only consider present information, and discover, among other things, that while the history-based controllers are among the most robust, they often unexpectedly provide more efficiency than fairness.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Miscellaneous

General Terms

Algorithms

Keywords

fairness, efficiency-fairness tradeoff, urban traffic control, traffic light controller.

1. INTRODUCTION

You have just spent a long time downtown stopped in gridlocked traffic. As you travel home, shouldn't traffic light controllers recognize this fact and award you the green light, to spread the pain more evenly? This is the idea behind history-based traffic control. Such a traffic controller would take into consideration a vehicle's past stopped-at-red history elsewhere, increasing the fairness with which vehicles are dealt globally on a per-trip basis.

Fairness can be a highly desirable trait: citizens of a town may desire a system which attempts to give every driver approximately the same percentage of waiting-time. But with a few exceptions [13, 15, 18], the literature has instead largely focused on some notion of system-wide *efficiency*, which might translate into lower emission levels, higher average speed, lower probability of stopping, etc. We imagine this lack of study is because fairness is not an easy thing to implement with existing hard-wired traffic controllers. But if one instead views these controllers in a multi-agent context, with vehicles (and their histories) traveling from controller-agent to controller-agent, then a variety of history-based methods become evident which we will argue notionally provide a kind of fairness. In this paper we compare various multi-agent traffic controllers, including history-based ones, and examine the degree to which they are fair and/or efficient.

The idea behind multi-agent traffic control is to develop a good emergent macro-level traffic-control behavior out of the micro-behaviors of the individual traffic lights. But traffic lights typically "live in the now": they know little of where specific vehicles have been and where they plan to go. At best, such controllers may use past historical information about vehicles in the aggregate, and perhaps heuristic information about likely future traffic patterns. Some past work [21, 22] has explored what happens when traffic light controllers are instead told of trip plans by the drivers: but we think it may not be reasonable to demand such information of the drivers. However, it *is* plausible to collect information about vehicles' immediate past histories to help inform the controllers, given reasonable privacy restrictions. This allows traffic controllers to communicate with one another, in some sense, through the immediate histories of cars as they travel from intersection to intersection.

We introduce this notion of historical fairness by allowing cars to store "credits" they receive when waiting at red lights, and cash the credits in when passing through intersections. Traffic lights base their decisions on the credits of various cars at the intersection. We also examine history methods which consider the mean waiting time the vehicles have spent at intersections so far along their respective trips. We compare these various methods against two non-history-based methods from the literature: one awards the green light to the lanes with more cars, and the other one prefers the lanes where cars have spent the longest time waiting.

In focusing on the effect of using "history" from a multi-agent perspective, our simplified experimental setup ignores a variety of important game theoretical and modeling issues which may be useful to study in the future. For example, we treat drivers as non-adaptive entities, while real drivers have varying risk attitudes and exhibit adaptive behavior, all of which present complex, difficult-to-model issues [14, 16, 19]. Even so, we believe that the concept of credits as history may be useful in the abstract in various other multi-agent scenarios involving, among other things, network resource allocation [23], package delivery, etc.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

As we discuss later in the paper, our history-based traffic control methods perform well, but often in surprising ways. Notably, though they were designed to provide fairness, several unexpectedly reside in the *efficiency* side of the efficiency-fairness spectrum!

The paper is organized as follows: Section 2 discusses related work. Section 3 introduces our simulation model. Section 4 describes the various traffic-control micro-behaviors in the study. The experiments, comparison metrics, and experimental results are given in Section 5, followed by conclusions in Section 6.

2. RELATED WORK

In this paper we will refer to the traffic controller of an entire intersection as a *traffic light*. A traffic light *pattern* is a combination of red and green lights for each incoming lane that prevents cars from crashing into one another. The interval of time allotted to a pattern is called a *phase*.

The most common metrics used in the traffic light literature are *waiting time* [1, 8, 9, 12, 18, 22], *delay* (the difference between the minimum possible travel time and the actual travel time, including acceleration and deceleration delays) [5, 15, 20], and *travel time* [1, 8, 9, 15]. Other popular metrics are *percentage of stopped cars* [11, 20], *density of cars* [3, 7, 6, 10] and *global flow* [2] (global density times average speed).

One may view efficiency and fairness as the mean and variance, respectively, of these various metrics. Nearly all the literature has concentrated on efficiency (the mean); and fairness has been only lightly studied. Exceptions include: Liu et al [15] analyzed their controllers' efficiency and fairness separately; and Krogh et al [13] and Montana and Czerwinski [18] combined the two into a single measure.

The notion of "fairness" used in Liu et al deals with cars waiting at a single isolated intersection. The cars' delays are averaged per lane[1] and the fairness is the standard deviation of the lanes' expected delays. We instead are interested in fairness (and efficiency) spread throughout a multi-traffic-light system. Thus our efficiency and fairness measures will be largely based on the mean of waiting times at the intersections along a vehicle's trip. This is the "trip waiting time" metric used in [21, 22], but normalized by the number of intersections in the trip.

It is worthwhile mentioning a very popular paradigm in urban traffic control, the "green wave," which coordinates traffic lights so that cars traveling at a certain speed go across those intersections without stopping. The green-wave phenomena cannot be achieved along all streets simultaneously, so traditionally it is used for heavy-flow arteries with light flow intersecting streets. A more general approach is to dynamically decide the sets of intersections that should coordinate. For example in [6] the traffic lights run a distributed constraint satisfaction algorithm to get ad-hoc green-waves.

There have been some efforts to achieve green-waves with implicit communication through density or flow. The idea is to provide an emergent green wave through coordinated behaviors. Bazzan's [3] traffic lights are self-interested agents that learn to coordinate with their neighboring traffic lights by playing coordination games. Gershenson's [11] basic control method changes to green when the total time spent by cars approaching or waiting at the red light exceeds a given threshold. Large enough "platoons" of cars move from light to light, causing it to turn green by their sheer mass, resulting in emergent green waves. This notion is closely related to our new history-based methods presented later.

[1]The authors actually used the concept of a *movement*, which is an incoming lane or a group of incoming lanes sending cars in the same direction.

3. THE URBAN TRAFFIC MODEL

We use a network model to simulate traffic flow. Nodes in the network are intersections, each one manned by a traffic light. Edges in the network are one-way road segments, each with a specified length, connecting intersections. Two-way road segments are modeled by using two edges, one in each direction.

On this network travel cars. When a car starts along an edge, it waits for a period of time equal to how long it would take to drive along the road to the next intersection. At that point it emerges and is placed into one of several queues depending on its desired action at that intersection. More formally, for each pair of edges $\langle a, b \rangle$ at an intersection, there is a queue $a \rightarrow b$ into which cars emerging from the edge a are placed while they wait for permission to start traveling on b. Queues may be thought of as the intersections' various turn and go-straight lanes of movement.

Our motivation for using a course-grained network of queues, rather than a fine-grained cellular-automaton-style grid of cells, is twofold. First and foremost, it permits the simulation to run much faster. Second, it allows us to choose whether or not to model individual road segments' capacities (we have declined to do so[2]). We also believe that a similar network structure may be applicable to other relatively unrelated domains, such as entry/exit policies for internet routers [23].

This means that the simulation is neither micro-level nor macro-level. Unlike in [2, 11, 21, 22], we do not explicitly model the movement of cars along road segments. We also make many simplifying assumptions a micro-level model would not make: for example, our cars all run at the same speed regardless of traffic density or road capacity. Further, as we do not consider road capacity, we only examine traffic flow difficulties due to long waits at stoplights. However, we do model cars and traffic lights explicitly, rather than treating them as flow in a macro-level dynamical systems model.

There is a constant number of cars traveling at any time along the streets, each following a predetermined *trip* from an origin intersection to a destination intersection. At the start of the simulation, cars' starting points are randomly chosen from the set of intersections. As a car's trip begins, the car is simply placed on its first edge, and when it completes its last edge, its trip is over (thus the car is not involved in queues at its start or destination). After a car has completed its trip, its former destination becomes the origin of the next trip, and a new destination is chosen at random from among all intersections (excluding the origin). After a trip destination is selected, drivers choose from among two ways to plan the trip route. With 40% probability, drivers will build a trip by incrementally adding to the path a randomly-selected intersection from the set of neighbors that minimize the remaining distance to the destination. With 60% probability, the driver will choose a shortest path with the fewest turns (at most one turn for a two way orthogonal street grid); when there are two such solutions, the driver selects the one whose turn takes place furthest from the city center. The intuition here is that the center of the city is so crowded (mostly with drivers using the first behavior), that there are some drivers who (modeled with the second behavior) prefer to avoid it.[3]

Intersections. Every twenty seconds (in simulation time), each traffic light individually considers whether to change its pattern. A traffic light may choose between every possible pattern (meaning

[2]Specifying capacities is more realistic, but unless the drivers are capable of re-routing, capacities can produce artificial deadlocks.

[3]We recognize that in a more sophisticated scenario, drivers would base routes on previous experience and common places to travel [21, 22].

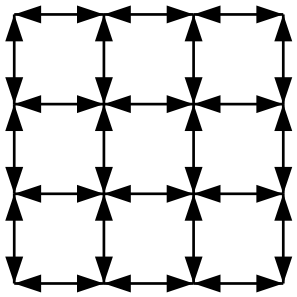


Figure 1: Example (4x4) Street Layout.

maximal set of *non-conflicting* queues). Two queues are said to conflict if cars leaving those queues might collide, either because cars in each queue are entering the same edge, or because the car trajectories intersect with one another. In the intersection $\begin{smallmatrix} & a & \\ c & & d \\ & b & \end{smallmatrix}$ between road segments a, b, c, and d, the queue $a \rightarrow b$ conflicts with, among others, the queues $c \rightarrow d$ (obviously), $c \rightarrow b$ (because they both enter b), and $b \rightarrow c$ (because, assuming drivers drive on the right-hand side, $b \rightarrow c$ cars turn into the $a \rightarrow b$ traffic). On the other hand, $a \rightarrow b$ does not conflict with $a \rightarrow c$, $a \rightarrow d$, $b \rightarrow a$, $d \rightarrow a$, or $b \rightarrow d$. It also does not conflict with $c \rightarrow c$, $d \rightarrow d$, or $a \rightarrow a$ either, but in this paper paths do not make U-turns, so these queues are always empty.

If a queue awarded the green light was not awarded one in the previous phase (its light was red then), it must wait for 5 seconds while conflicting queues clear the intersection (obstensibly during their yellow light period) before the green light may commence. Thus a light changing from red to green has only a 15-second green period, while a light staying green has a 20-second green period. While a light is green, cars waiting in the corresponding queue are removed one by one and permitted to enter the intersection with a three-second "acceleration penalty" corresponding to getting back up to speed after being stopped at the light. Thus in a 15-second green period five cars can traverse the intersection, but during a 20-second green period six cars can do so. If a car is approaching an intersection and the appropriate queue is empty, the car is allowed to go through at full-speed (with no 3-second penalty). The limit of five (six) cars going through per phase and per lane applies to any combination of cars removed from the queue and cars finding the queue empty and going through full speed.

In our model, all roads are two-way streets, laid out along an orthogonal mesh akin to a downtown city grid (see Figure 1). The blocks are squares, each 0.25 miles on a side. The traffic lights have 0.125 mile sensor ranges, inside which they may discern approaching cars and their desired intersection queues (imagine that turn lanes start 0.125 miles out and sensors are positioned where they start). All cars travel at 25 mph, so they need 36 seconds to cover the segment between two intersections. Regardless of whether they go straight or turn, all cars need exactly 5 seconds to traverse the intersection (not including the acceleration penalty).

4. THE TRAFFIC LIGHT CONTROLLERS

We divide the traffic light controller function into two subfunctions. The first subfunction assigns a *score* to each pattern. The second subfunction selects the winning pattern based on the patterns' scores.

The second subfunction uses one of two winner-selection methods: a deterministic method and a stochastic method. In the deterministic method, the pattern with the highest total score is the winner; in case of a tie, the winning pattern is the one with most cars in its queues; further ties are broken at random. The stochastic method uses *roulette selection*: the probability that a pattern will win is proportional to its total score. If all patterns have a score of 0, the pattern with more cars in its queues wins, and further ties are again broken at random. A light only bothers to select a new winner if cars are actually waiting at or approaching the intersection.

The first subfunction is called the *scoring function*, and assigns each pattern a score equal to the total points assigned to cars within the sensor range of the queues receiving the green light in that pattern. We examine the following score functions in combination with the aforementioned winner selection methods:

*Counting Cars (*CC*).* Each car yields one point. As a result, this controller defines scores simply as the total number of cars within the queues' collective sensor range. We include it as a baseline for our comparisons because it is the simplest and most intuitive adaptive traffic controller. Counting Cars was used in [4, 22].

*In-Range Time (*IRT*).* A car's points are equal to the length of time that has transpired since it was picked up by the queue sensors at the current intersection. This is a generalization of Gershenson's method [11], which only worked with two one-way road intersections. In-Range Time is also closely related to a notion of pheromones applied in [7]. There, cars deposit pheromones as they wait, and the traffic light controller makes a decision based on the total level of pheromones in each lane.

*Mean Waiting Time (*MWT*).* This is our first history-based method. The car's points are equal to its mean time spent waiting at intersections in its trip so far, including the current intersection.

*Previous Mean Waiting Time (*PMWT*).* This is our second history-based method. The car's points are equal to its average waiting time over the previous intersections of the trip but *not* including the current intersection. When at its first intersection of the trip, the car's score is the global mean waiting time over all intersections in that simulation run. In some sense, MWT$\approx$ PMWT + IRT.

*Credit Methods (*CREDIT*).* These are our final history-based methods. The car's points are defined as accumulated "credits" which it receives from, and pays back to, various traffic light controllers. There are four credit-disbursement methods, each a combination of the following two parameters:

- *How are cars compensated for receiving the red light?* If using the *general-sum policy*, after every phase (20 seconds), the cars in the lane of a red traffic light each receive one credit, and cars passing through a green light must pay the traffic light 5 credits or the car's accumulated credit wealth, whichever is smaller. If using the *zero-sum policy*, after every phase, cars passing through must again pay up to 5 credits as before, and the total credits paid to the intersection at that phase are in turn distributed evenly among cars in red-light lanes at that intersection. If no car is waiting at red, the zero-sum policy will not require a green-light car to pay anything.

- *How many credits does a car have initially?* At the start of a trip, a car's credit wealth is reset to either *10* or *100* credits.

The values for the parameters of the CREDIT controllers were chosen ad-hoc, and we made no effort to tune them. We note that CREDIT can easily accommodate emergency vehicles and public transportation by endowing special vehicles with unusual numbers of credits, or weighting their credits appropriately.

Grid size	Load	←*More Efficient*				Pareto Front					*More Fair*→	
10×10	31		CR(100, G)									
	62		CR(100, G)									
	125		CR(100, G)									
	250		CR(100, G)				IRT					
	500		CR(100, G)				IRT	MWT				
	1K		CR(100, G)		CR(10, G)		IRT	MWT				
	2K		CR(100, G)	CR(100, Z)	CR(10, G)			MWT				
	4K				CR(10, G)		IRT					
	8K	PMWT				CR(10, Z)			IRT	MWT	CR(10, Z)	
	16K	PMWT				CR(10, Z)	IRT		IRT		CR(10, Z)	CR(10, G)
4×4	31		CR(100, G)		CR(10, G)		IRT					
	62		CR(100, G)		CR(10, G)		IRT					
	125				CR(10, G)		IRT					
	250				CR(10, G)	CR(10, Z)	IRT	MWT				
	500			CR(100, Z)	CR(10, G)	CR(10, Z)	IRT	MWT				
	1K	PMWT						MWT	IRT	MWT		
	2K	PMWT				CR(10, Z)			IRT	MWT		CR(10, G)

Table 1: Pareto fronts for all combinations of city size and workload level. The controllers using the stochastic selection are shown with a gray background, while no background is used for the deterministic selection controllers. The CREDIT variants are depicted as CR(E,P), where E is the initial endowment of credits and P is the policy (either *Z* for zero-sum or *G* for general-sum compensation).

5. EXPERIMENTS

We tested the traffic light controllers under different city grid sizes and demand levels (what we refer to as the *workload*: the number of cars on the streets). We experimented with two scenarios based on the size of the city grid: 4×4 and 10×10. As mentioned in Section 3, we used a closed model to generate the workload, so the number of cars was constant throughout a simulation run. In our experiments we chose the following workload levels: 31, 62, 125, 250, 500, 1000, and 2000. For the 10×10 scenario we also used 4000, 8000, and 16000.

All presented results are based on 100 independent runs for each combination of city grid scenario, workload level, and traffic light controller. We ran each simulation for 1000000 seconds (approximately 11.5 days of simulated time). The reason for such a large simulation length is the stochastic nature of the traffic light controllers and route selection, requiring a large sample to accurately compare the traffic light controllers. In order to eliminate the boundary conditions at the beginning of a run, we discarded the data for those trips finished within the first 500 seconds of each run. Simulations were performed using the MASON multi-agent simulation toolkit [17].

5.1 Comparison methodology

When a car reaches its destination, it reports its average waiting time over all intersections along the trip. This statistic is the normalized version of the average trip waiting time ("ATWT") used in [21, 22].[4] Of interest to us is the distribution of this statistic over all trips: we use the mean and standard deviation of the distribution as our measures of efficiency and fairness, respectively, in the system.

We used a pareto-dominance criterion to compare two methods: method *A* is superior to method *B* if it is at least as good in both metrics and better in at least one metric. By "better" we mean statistically significantly superior to, and by "at least as good" we will mean significantly superior to or not significantly distinguishable from. Since the efficiency and fairness of a controller are distribution samples, we needed a statistical analysis mechanism to assess the Pareto-dominance relationships. We used unequal-variance repeated t-tests with a Bonferroni-corrected significance level $\alpha = 2.31481 \times 10^{-6}$, which exceeded a 99% confidence in the correctness of the entire set of results presented in this paper.

5.2 Results

Table 1 contains the Pareto fronts for all the city size/workload level combinations. We have taken the liberty of spacing the items in the fronts so that occurrences of the same controller are on the same column. There is no special meaning behind this other than our attempt to make the table easier to read. See Figure 2 for representative examples of what the fronts actually look like.

Overall, the results are very promising for the CREDIT controller, as its variants appear in the Pareto fronts in all our scenarios but one (4×4 grid, 1000 load). In general, CREDIT seems to be a versatile controller, adapting well to both high and low workloads.

An analysis of Table 1 reveals some interesting observations. The most interesting one is that although we expected the history-aware controllers to be fair at the cost of efficiency, some variants proved to be more efficient than the history-oblivious controllers. Most unexpected is PMWT being the most efficient controller for the highest workloads. We did not expect this because PMWT ignores the waiting time information about the current intersection, which would provide efficiency, and uses only past information, which in some sense is the source of fairness. Also, we were surprised to find that the deterministic CREDIT variants are more efficient than MWT, even though the information used by the latter is more accurate then that used by the former.[5] Oddly, the situation is reversed in the stochastic versions of the controllers.

Another observation is that for (and only for) high enough workload levels, the stochastic selection controllers are consistently

[4]Using the *total* trip waiting time as our statistic would make drivers on a short trip expect the same total delays as the ones on very long trips, which we felt was unreasonable.

[5]The CREDIT controller cannot tell if a car has waited $k \times 20 + 2$ seconds or $k \times 20 + 18$ seconds. Also, if a queue is long enough, some of the cars waiting in it do not go through in a single 20-second green phase, yet are not compensated for waiting. Therefore a car's compensation is only a rough measure of the time spent waiting in the queue, as it ignores these "green phases."

more fair than the deterministic ones (see Figure 2(c) for a typical example). We observe this behavior starting at 1000 cars for the 4×4 grid and 8000 cars for the 10×10 grid. For these scenarios the drivers' expected waiting times per intersection are 2 and 3 minutes respectively, which we feel is not unrealistically high for a rush-hour traffic-jam situation. Why stochastic controllers would perform in this way is still unknown to us. Figures 2 reveal why stochastic controllers disappear from the front however: as workload drops, the deterministic controllers maintain their efficiency advantage but secure more fairness relative to the stochastic controllers.

There are a few other points worth mentioning. One can make a CREDIT variant more fair or more efficient by simply adjusting the initial endowment of credits. The larger the initial number of credits, the less likely a queue with fewer cars can outscore a queue with more cars, resulting in a less fair controller. At the limit, CREDIT with an endowment of infinity is roughly equivalent to CC, except for the tie situations. But CC is dominated in all the scenarios presented here, which leads us to the conjecture that fair tie breaking should not be ignored when designing a traffic light controller.

6. CONCLUSIONS

We have introduced the concept of trip-based fairness and showed a simple approach to incorporate it into traffic light control. We have implemented several controllers to investigate the effect of awarding the green light based on information regarding the cars' past experiences. The testbed we used was a meso-simulated two-way street grid scenario with uniform demand and a wide range of traffic loads. All controllers investigated here, however, work out-of-the box with any intersection set-up: arbitrary number and layout of one- and two-way streets, with and without shared lanes. Cars are not required to participate in the traffic control system: if a light senses a non-participating car, it might set up a proxy credit accumulator initiated with some mean credit amount, but which does not travel with the car when it leaves the intersection.

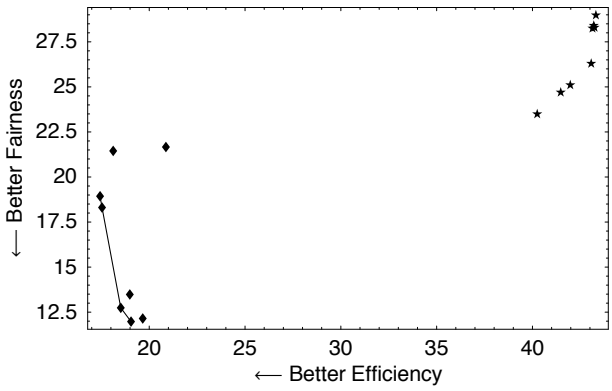
We expected the newly proposed controllers to be fair at the expense of efficiency, but they performed reasonably well in general, and some even turned out unexpectedly efficient (at the expense of fairness). More research is needed to gain a fuller understanding of the phenomena involved.

The data provided interesting (and often unexpected) results when dealing with overcrowded traffic conditions. Here, stochastic selection is a viable solution for improving fairness; and for the history-based methods, the cruder the use of information about the present, the more efficient the controller.

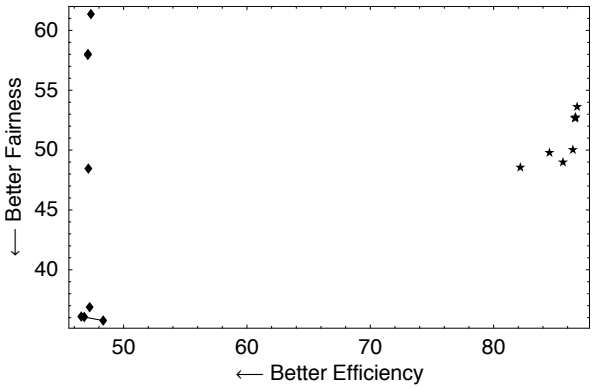
6.1 Future work

Presently our controllers work with a pre-set phase length. While this is not an unusual bias in the traffic light control literature, we think we should try to eliminate the phase length parameter or incorporate phase adaptation into the controllers.

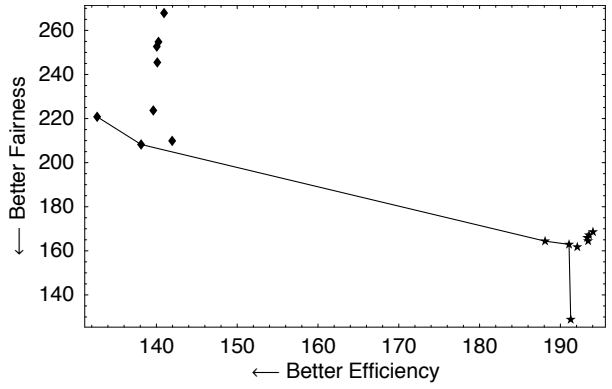
The original inspiration for this paper was to allow the drivers to let the traffic lights know whether the drivers are in a hurry or not. We plan to investigate various auction-style methods for traffic light controllers. In these methods, the drivers are self-interested agents trying to resolve who goes first using a currency accepted at all traffic lights. The lights act as benevolent auctioneers, awarding green lights to those deserving it (able and willing to "pay") the most. When drivers are not in a rush they might choose to bid low and save some credits for the future. The resulting traffic controller system acts as a market, which hopefully requires less tuning because it is self adaptive.



(a) 2000 cars



(b) 4000 cars



(c) 8000 cars

Figure 2: Results for the 10×10 grid with different workloads: 2000, 4000 and 8000 cars. The deterministic controllers are represented as ♦ and the stochastic ones as ★; the lines connect the non-dominated controllers (see Table 1); the units on the plots' axes are seconds.

This is only a sample of future work for this domain. We think distributed traffic control is an attractive problem for multi-agent systems, with a broad, relatively unexplored research domain and a payoff in the form of immediate real-world impact. The area also presents a number of major research challenges: the number of cars and traffic-control devices on the road is huge, and the problem is highly decentralized with regard to vehicles and, more often than not, with regard to traffic-control devices. The humans in the vehicles co-adapt to one another and to traffic controllers. Dense traffic can produce emergent, unexpected phenomena. As such, we imagine this area may prove both fruitful and increasingly important in the future.

7. ACKNOWLEDGMENTS

We are grateful to Aimee Flannery for her invaluable assistance, and to Liviu Panait for his help in organizing this paper.

8. REFERENCES

[1] J. M. Anderson, T. M. Sayers, and M. G. H. Bell. Optimization of a fuzzy logic traffic signal controller by a multiobjective genetic algorithm. In *Ninth International Conference on Road Transport Information and Control*, pages 186–190, April 1998.

[2] R. Barlović, T. Huisinga, A. Schadschneider, and M. Schreckenberg. Adaptive traffic light control in the ChSch model. In P. H. L. Bovy, S. P. Hoogendoorn, M. Schreckenberg, and D. E. Wolf, editors, *Workshop on Traffic and Granular Flow*, 2003.

[3] A. L. C. Bazzan. A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multi-Agent Systems*, 10(2):131–164, March 2005.

[4] C. Bertelle, A. Dutot, S. Lerebourg, and D. Olivier. Road traffic management based on ant system and regulation model. In *The International Workshop on Modeling & Applied Simulation*, pages 35–43, Bergeggi (Italy), October 2003.

[5] E. Bingham. Reinforcement learning in neurofuzzy traffic signal control. *European Journal of Operational Research*, 131(2):232–241, July 2001.

[6] D. de Oliveira, A. L. C. Bazzan, and V. Lesser. Using cooperative mediation to coordinate traffic lights: a case study. In *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 463–470, July 2005.

[7] D. de Oliveira, P. R. Ferreira, Jr., P., A. L. C. Bazzan, and F. Klügl. Reducing traffic jams with a swarm-based approach for selection of signal plans. In *Proceedings of Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence*, Brussels, Belgium, 2004. Poster.

[8] K. Dresner and P. Stone. Multiagent traffic management: An improved intersection control mechanism. In *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 471–477, July 2005.

[9] N. V. J. Findler and J. Stapp. A distributed approach to optimized control of street traffic signals. *Journal of Transportation Engineering*, 118, 1992.

[10] J. France and A. A. Ghorbani. A multiagent system for optimizing urban traffic. In *IAT*, pages 411–414. IEEE Computer Society, 2003.

[11] C. Gershenson. Self-organizing traffic lights. *Complex Systems*, 2005. to appear.

[12] I. Kosonen. Multi-agent fuzzy signal control based on real-time simulation. *Transportation Research Part C*, 11(5):389–403, October 2003.

[13] C. Krogh, M. Irgens, and H. Traetteberg. A novel architecture for traffic control. In L. Olaussen and E. Helli, editors, *The 3rd International Conference on Vehicle Navigation and Information Systems*, pages 75–81, 1992.

[14] D. Levinson. The value of advanced traveler information systems for route choice. *Transportation Research Part C*, 11(1):75–87, February 2003.

[15] H. X. Liu, J.-S. Oh, and W. Recker. Adaptive signal control system with on-line performance measure. *Transportation Research Record*, 1811:131–138, 2002.

[16] H. X. Liu, W. Recker, and A. Chen. Uncovering the contribution of travel time reliability to dynamic route choice using real-time loop data. *Transportation Research Part A*, 38(6):435–453, July 2004.

[17] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan. Mason: A multi-agent simulation environment. *Simulation*, 81(7):517–527, July 2005.

[18] D. J. Montana and S. Czerwinski. Evolving control laws for a network of traffic signals. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 333–338, Stanford University, CA, USA, 1996. MIT Press.

[19] J. Wahle, A. L. C. Bazzan, F. Klgl, and M. Schreckenberg. The impact of real-time information in a two-route scenario using agent-based simulation. *Transportation Research Part C*, 10(5–6):399–417, December 2002.

[20] W. Wei and M. Wang. Traffic signal control using fuzzy and neural network. In *8th International Conference on Neural Information Processing*, Shanghai, China, November 2001.

[21] M. Wiering, J. van Veenen, J. Vreeken, and A. Koopman. Intelligent traffic light control. Technical Report UU-CS-TR-2004-029, Institute of Information and Computing Sciences, Utrecht University, 2004.

[22] M. Wiering, J. Vreeken, J. van Veenen, and A. Koopman. Simulation and optimization of traffic in a city. In *IEEE Intelligent Vehicles Symposium (IV'04)*, 2004.

[23] Y. Zhou. *Resource Allocation in Computer Networks: Fundamental Principles and Practical Strategies*. PhD thesis, Drexel University, 2003.

Communication Management Using Abstraction in Distributed Bayesian Networks

Jiaying Shen and Victor Lesser
Department of Computer Science
University of Massachusetts
Amherst, MA 01003-4610
{jyshen, lesser}@cs.umass.edu

ABSTRACT

Techniques were developed in previous work for managing communication in a controlled satisficing manner in two layer distributed Bayesian Networks. DEC-MDPs were used to sequence the information transferred in order to guarantee the required confidence level. In this paper, we introduce multiple abstraction layers into the Distributed Bayesian Network as a way of carrying more useful information in transmitted data to further reduce the number of messages that need to be sent. An algorithm is developed to automatically generate appropriate abstraction data. Techniques are introduced to effectively incorporate this abstraction data set into the DEC-MDP framework. We show that the appropriate addition of abstraction data actions simplifies the DEC-MDP while reducing the expected communication cost. This work provides us with a formal view of the use of abstraction in agent cooperation and begins to give us an understanding of when the less abstract data needs to be transmitted.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

DESIGN, PERFORMANCE

Keywords

Abstraction, Communication, Decentralized MDP

1. INTRODUCTION

In complex distributed applications, such as distributed interpretation (DI), a problem is often decomposed into a set of subproblems and each subproblem is distributed to an agent who will be responsible for solving it. The existence of interactions between subproblems means that the agents cannot simply solve the subproblems individually and then combine local solutions together. In such systems, the amount of communication among agents may be very significant in order to guarantee global optimality or even global consistency. Thus, "satisficing" approaches have been developed that trade off optimality for reduced communication [4]. One approach is for agents to generate local solutions based on their own data and then transmit these high level solutions to other agents. Based on the consistency and credibility of these local solutions, new local solutions may be generated or more detailed data sent until a sufficient level of consistency and credibility has been achieved among the agents. An important characterization of such distributed protocols is how much communication is required and the likelihood that the solution will be the same as that generated by an optimal centralized algorithm which uses all available information.

Most approaches to managing communication trade off solution quality for reduced communication, but only from a statistical point of view. The behavior of the algorithms are often analyzed over a collection of problems to say that p percent of the time they will get the required solution quality q with an average amount of communication c [4].

Shen et al. [11] took this satisficing approach to the next step by designing a parameterized algorithm where one can predict, for a desired confidence level in the final solution, the expected amount of communication the agents need. They studied these issues in terms of a two layered Distributed Bayesian Network (DBN), as shown in Figure 1. A decision-theoretic framework was proposed to model this multi-agent coordination decision problem. A decentralized Markov Decision Process (DEC-MDP) [1] can be constructed from the Bayesian Network structure to find a joint communication policy that minimizes the expected communication cost.
Each agent needs to decide its communication actions based only on its local observations. The framework takes the amount of communication into account, as well as the quality of the final solution. The agents use only the necessary amount of communication to achieve the required level of solution quality.

In this paper, we introduce abstraction layers into the Distributed Bayesian Network as a way of carrying more useful information in one piece of transmitted data to further reduce the number of messages that need to be sent. In order to effectively use the abstraction technique, there are two questions that need to be answered: How to generate the abstract information to be transferred, and When to transfer the abstract information. To answer the first question, we present an algorithm that automatically generates

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

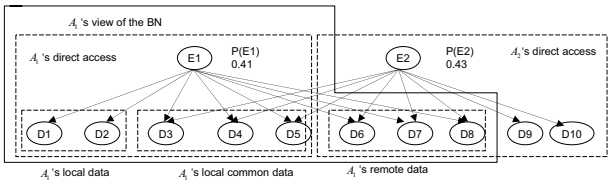


Figure 1: There are two events E_1 and E_2. Data $D_1, D_2, ...D_{10}$ are distributed between two agents. A_1 has access to $D_1, ...D_5$ and is responsible for solving E_1, while A_2 can see only $D_6, ...D_{10}$ and is responsible for solving E_2. The value of E_1 is dependent on A_1's data and vice versa. The objective is for A_1 and A_2 to decide what E_1 and E_2 are with required confidence while minimizing the expected communication cost.

appropriate abstraction data, which reduces the expected communication cost necessary to achieve the required confidence level. This is a greedy approach since the abstraction data enables the agent to immediately reach the desired confidence level. We chose this type of abstraction data because it is intuitive and straightforward to generate, yet it enables us to study the concept of abstraction and its use in reducing communication cost from a formal perspective. There are other forms of abstraction that can be potentially useful. For example, some abstraction data may contain useful information but is not sufficient to reach the confidence by itself. Other abstraction may be an intermediate interpretation result that can be incorporated into the local solution. We plan to look into these different types of abstraction in future work.

To answer the second question, we examine three different approaches to incorporate the new abstract communication actions to the existing DEC-MDP. The *all data action selection* approach allows abstract communication actions as well as the normal communication actions in all the states. The *abstraction data action selection* approach allows only the abstract communication actions in all the states. In the *hierarchical action selection* approach, before an agent has acquired all of the abstraction data, the agents are not allowed to transfer any of the information available at the next more detailed level. Experimental work demonstrates that the hierarchical approach simplifies the DEC-MDP while still reducing the expected communication cost. Furthermore, experiments on larger networks with multiple abstraction layers show that the hierarchical approach is able to solve more and larger networks than the other approaches and therefore is more suitable for scaling up the approach.

Most work studying communication in DEC-MDP and its related models studies synchronizing communication, where every communication action synchronizes the agents' views of the world state [8, 10]. In contrast, this work studies the problem of what to communicate, where only part of the local observation is transferred between the agents. Abstraction has often been used in hierarchical planning community to represent abstract plans that can be further refined into specific actions. In particular, Clement and Durfee [6] used summary conditions, a form of abstraction, to coordinate the concurrent interactions of plans at different levels of abstraction between agents. Abstraction has also been used in distributed diagnosis research to identify the sources of problem. Chung and Barrett [5] demonstrated how Boolean expressions can facilitate finding minimal cost diagnoses in linear time. However, neither work studies the tradeoff between abstraction and communication. In the original Distributed Problem Solving work [7], abstraction was used as a mechanism for controlling the information needing to be communicated. However, the use of abstraction was not formalized. Nor was there a clear understanding of when the lower level data needed to be transmitted. Carver and Lesser [3] studied the use of multiple levels of abstraction to reduce the necessary communication. Like [7], the abstraction layers were predefined and the use of abstraction was not formalized. In contrast, the algorithm we introduce allows the system to generate appropriate abstraction data automatically without predefinition. Our study of the addition of abstraction layers into the Bayesian networks and transferring abstraction data as actions in the DEC-MDP provides us with a formal view of the use of abstraction in the management of communication cost in a distributed problem solving system.

The next section formally defines the problem we study and summarizes the DEC-MDP model we developed to generate the communication strategy. Section 3 introduces the algorithm that automatically generates the abstraction data that when transmitted can greatly facilitate the achievement of the confidence level. Section 4 discusses how the abstraction data can be incorporated to improve the system performance. Section 5 extends the different approaches discussed so far to larger networks with multiple abstraction layers. We conclude with a discussion of future work.

2. PROBLEM DEFINITION AND DEC-MDP MODEL

In an interpretation system there is a set of observable data that is caused by some possible events, which agents do not have the means to directly observe. The agents need to observe the data and identify the events that are most likely the cause of them. In many environments, the problem is inherently distributed and the network is fairly large. In those cases, it is common to distribute not only the data but also the interpretation task among several agents. Inevitably, there is a close interaction between the agents since the local sub-problems an agent is responsible for are often dependent on some of the data collected by other agents. The existence of subproblem interaction means that the agents will need to communicate during problem solving. Therefore, agent communication strategies can have a major effect on the cost of problem solving.

We use a two-layer Bayesian Network to represent the problem structure (Figure 1). The top level nodes are the events that are the possible causes of the observed data, while the leaves are the raw data gathered by various agents. There are two agents, each of whom has direct access to only a part of the observable data. The interpretation task is distributed to the two agents as well with each agent responsible only for part of the overall problem. Additionally, an agent only has knowledge of the network structure that is directly relevant to its interpretation task. The agents can either request or send data. The objective is to identify the most likely interpretation of the causal events with a target level of confidence using as little communication as possible. For example, in Figure 1, there are two agents A_1 and A_2. A_1 is responsible for interpreting the event E_1 and therefore

knows about the part of the causal network that is relevant to E_1 . Out of the relevant data $D_1, ..., D_8$, it can directly observe only D_1 through D_5. D_6, D_7 and D_8 are *remote data* and may need to be transferred in order for A_1 to solve its local problem with confidence.

The **evidence** ϵ_{A_i} of an agent A_i are the values of the data that the agent has collected so far. They can be the values that are observed directly by that agent or acquired from the remote agent. An agent A_i is only able to find the most likely interpretation of the local events given its current evidence set, i.e., $MAPI(\epsilon_{A_i}) = argmax_h P(E_{A_i} = h|\epsilon_{A_i})$ [4]. Ideally the decentralized system should generate the interpretation that a centralized system will generate given the complete data values $MAPI(\epsilon^*_{A_i})$, where $\epsilon^*_{A_i}$ is the complete evidence set relevant to A_i. Unfortunately, with only partial knowledge of the relevant data values, it is not always guaranteed that the current local MAPI is the global MAPI. On the other hand, with the conditional probability table given by the BN, an agent can predict the probability of different relevant data configurations given its current evidence $P(\epsilon^*_{A_i}|\epsilon_{A_i})$. Hence we have the following definition.

Definition 1. **Confidence** *is the likelihood of the local MAPI being the global MAPI of E_A given the current known evidence of A.*

$$\begin{aligned} C(MAPI(\epsilon_A)) &= P(MAPI(\epsilon_A) = MAPI(\epsilon^*_A)) \\ &= \sum_{e \in \{\epsilon^*_A | MAPI(\epsilon_A) = MAPI(\epsilon^*_A)\}} P(e|\epsilon_A) \end{aligned}$$

What is important about the solution quality of a distributed system is not the likelihood of an event being the local MAPI, but whether the local MAPI agrees with the global MAPI, i.e., whether the decentralized system will generate the same result that a centralized system would. That is exactly what this definition of confidence achieves. Using confidence as a measurement of the solution quality of a decentralized system as compared to that of a centralized one, we can make a tradeoff between the quality of the solution and the communication cost. Another interesting and useful property of our definition of confidence is that it is guaranteed to reach 100% when the values of all the relevant data are known. As a result, a given confidence level can always be satisfied.

Given a problem structure in a two level BN such as the one in Figure 1 and a specified confidence threshold, we need to find a communication strategy. Such a communication strategy should specify what communication action each agent should take based on their current knowledge at each stage. In this paper we are considering only the case of synchronous communication. In other words, the two agents take turns to decide their actions in the sequence of $A_1, A_2, A_1, A_2, ...$ until the confidence threshold is reached.

Definition 2. *A* **communication strategy** $\pi_{\langle A_1, A_2 \rangle}$ *is a pair of policies π_{A_1} and π_{A_2}. Each policy π_{A_i} is a mapping from local evidence and communication history to a local action: $\pi_{A_i} : \epsilon_{A_i} \times H \rightarrow a_i$.*

There are three classes of actions for each agent: SEND, REQUEST and NULL. The content of the SEND and REQUEST actions at each stage are restricted by the communication history so far. An agent can send only the part of the local common data that the other agent has not yet acquired, and can request only a part of the remote data that it has no knowledge of. When the communication history indicates that it is not A_i's turn to act, the policy of that agent will always be NULL. A REQUEST action and its response by the other agent occur in the same stage. For example, in Figure 1, let us assume that initially $\epsilon_{A_1} = \{D_1 = 1, D_2 = 0, D_3 = 0, D_4 = 1, D_5 = 1\}$ and $H = \emptyset$. A valid action for A_1 at this step would be "SEND D_3 and D_4". It could also "REQUEST D_8 and D_9". On the other hand, since it is not A_2's turn, the only valid action for A_2 is NULL. The cost model for this framework is very general. Each communication has its own cost, based on its action class and the particular data sent. The only restriction is that the cost of a REQUEST must also include the cost of the immediate reply by the other agent. For example, in our experiments SEND costs 1 per data sent. A REQUEST action costs 1 for the request plus a reply cost equal to a SEND action of the same data. "REQUEST D_8 and D_9" would cost 3.

To summarize, the problem can be defined by a two level BN and a required confidence threshold t. The goal is to find a communication strategy $\pi_{\langle A_1, A_2 \rangle}$ that allows the agents to interpret their local events with a confidence greater than, or equal to t, i.e., $C(MAPI(\epsilon_{A_1})) \geq t$ and $C(MAPI(\epsilon_{A_2})) \geq t$, while minimizing the expected communication cost.

To solve this problem, we model it as a DEC-MDP. Unlike a centralized MDP, in a DEC-MDP the process is controlled by multiple agents, each with a different view of the current global state. At each time step the agents' observations together uniquely determine the global state, though frequently no one agent has this complete view. A local policy for an agent is a mapping from its local histories of observations to an action. The goal of a decentralized MDP is to find an optimal joint policy for all of the agents that maximizes the expected total reward.

We model our problem in the framework of a decentralized MDP as follows.

- Every global state is in the form of $s = \langle \epsilon^*, i \rangle$, where ϵ^* is all of the low level data values observed by the system, and i is an external feature that indicates it is A_i's turn to communicate. s_0 is a dummy start state $\langle \emptyset, 0 \rangle$. When all the sensors collect their data, the start state transitions to one of the possible real global states $\langle \epsilon^*, 1 \rangle$ before any communication takes place.
- Ac_1 and Ac_2 are action sets of the two agents. $Ac_i \in \{$ NULL, SEND x, REQUEST $y\}$, where x is a subset of the local common data of A_i and y is a subset of the remote data. When it is not A_i's turn to communicate, its action is the NULL action. A joint action $\langle Ac_1 \neq NULL, NULL \rangle$ transitions a state $\langle \epsilon^*, 1 \rangle$ to $\langle \epsilon^*, 2 \rangle$, and a joint action $\langle NULL, Ac_2 \neq NULL \rangle$ transitions a state $\langle \epsilon^*, 2 \rangle$ to $\langle \epsilon^*, 1 \rangle$. If at state s, both agents have reached confidence level t for its local interpretation task, i.e., when $C(MAPI(\epsilon_{A_i})) \geq t$, $i = 1, 2$, then s is a final state.
- P is a transition probability table. $P(s'|s, a_1, a_2) = P(\langle \epsilon^*, i \rangle | \langle \epsilon^*, j \rangle, a_1, a_2) = 1$ iff $i \neq j$. Otherwise, $P(s'|s, a_1, a_2) = 0$. As a special case, the transition probability from s_0 to $s = \langle \epsilon^*, 1 \rangle$ after both agents execute NULL action is $P(s|s_0, NULL, NULL) = P(\epsilon^*)$.

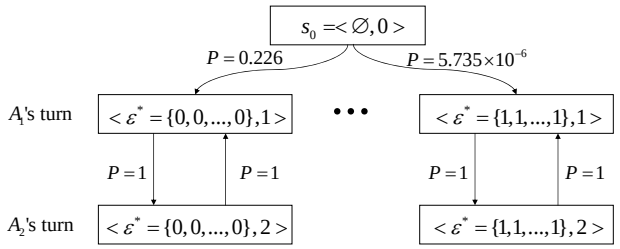


Figure 2: A portion of a typical decentralized MDP.

It can be calculated given the BN structure.

- R is a reward function. $R(s, a_1, a_2, s') = R(a_1, a_2) = -c(a_1) - c(a_2)$, where $c(a_i)$ is the cost of the communication action a_i, $i = 1, 2$.
- Ω_1 and Ω_2 are the sets of observations for the two agents. An observation $o_i \in \Omega_i$ is the data value just sent or received by A_i. As a special case, after s_0, each agent observes the values of its local data. An observation sequence $\overline{o}_i = \langle \epsilon^0_{A_i}, H \rangle$, where $\epsilon^0_{A_i}$ is the initial observation of A_i's local data set before any communication occurred in the system, and H is the communication history of the system.
- $O(s, a_1, a_2, s', o_1, o_2) = 1$ iff o_1 and o_2 are the new data sent or received by the agents after them taking actions a_1 and a_2 at state s. Otherwise it equals to 0.
- A local policy π_i for agent A_i is a mapping from the local observation sequence $\overline{o}_i$ to an action $a_i \in Ac_i$. A joint policy $\pi = \langle \pi_i, \pi_j \rangle$ is a pair of local policies, one for each agent.

For example, Figure 2 is the DEC-MDP built for the problem in Figure 1. A possible global state s when both agents just observed their local sensor data might be $\langle \{D_1 = 1, D_2 = 0, D_3 = 0, D_4 = 1, D_5 = 1, D_6 = 0, D_7 = 1, D_8 = 1, D_9 = 0, D_{10} = 0\}, 1\rangle$. The possible actions for A_1 are NULL, SEND x, and REQUEST y, where $x \subseteq \{D_3 = 0, D_4 = 1, D_5 = 1\}$ and $y \subseteq \{D_6, D_7, D_8\}$. Since it is not A_2's turn to communicate, its only action is NULL. If A_1 chooses REQUEST $\{D_7, D_8\}$, then the next state is $\langle \{D_1 = 1, D_2 = 0, D_3 = 0, D_4 = 1, D_5 = 1, D_6 = 0, D_7 = 1, D_8 = 1, D_9 = 0, D_{10} = 0\}, 2\rangle$ and both agents observes $\{D_7 = 1, D_8 = 1\}$. The observation sequence for A_1 is $\overline{o}_1 = \{\langle D_1 = 1, D_2 = 0, \ldots, D_5 = 1\rangle, \langle D_7 = 1, D_8 = 1\rangle\}$, and the observation sequence for A_2 is $\overline{o}_2 = \{\langle D_6 = 0, D_7 = 1, \ldots, D_{10} = 0\rangle, \langle D_7 = 1, D_8 = 1\rangle\}$. The policy π_i decides which action to take for each agent based on its current $\overline{o}_i$. Remembering the entire communication history H for both agents have two purposes. First, it includes the newly acquired evidence received from the other agent. Second, even for the agent who just SEND part of its own data without receiving any new evidence, it is important to remember this so that in the future it does not send the same information unnecessarily.

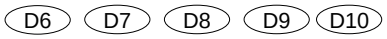
3. GENERATING THE ABSTRACTION LAYER

So far we have discussed two layer networks, which means that there are no intermediate abstraction data. Many domains, however, have structures that can be exploited to improve the performance of both local processing and inter-agent merging of evidence. Domain structure can result from a number of factors such as independencies among subsets of events or between events and certain data subsets and cases where only a fraction of the combinations of some data are informative. Many of these situations are best captured by using BNs that include intermediate nodes that lie between the event and data levels.

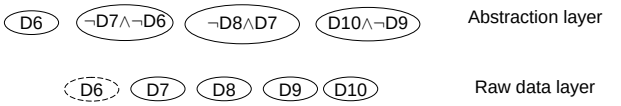
```
1.1 Input: agent A's view of BN, the values of
    the local data, the required confidence
1.2 Output: a set of logic expressions of the
    remote data that, if true, put the local
    confidence above the required confidence
1.3 Set DataList = all the remote data of A,
    potential = null
1.4 Initialize LogicTree
1.5 while DataList.next is not null do
        currentLeaf = DataList.next
        add currentLeaf and ¬currentLeaf as new
        children to LogicTree.root
        if confidence given currentLeaf ≥ required
        confidence then
            mark currentLeaf
            continue
        if confidence given ¬currentLeaf ≥
        required confidence then
            mark ¬currentLeaf
            continue
        if potential == null then
            potential.add(currentLeaf)
            potential.add(¬currentLeaf)
            continue
        for each p in potential do
            potential.add(addChildren
            (currentLeaf, p), addChildren
            (¬currentLeaf, p))
1.6 convert the marked part of LogicTree to a set
    of logic expressions and return it
1.7 addChildren (currentLeaf, p)
    begin
        Add p to LogicTree as a child of
        currentLeaf in a depth first fashion, check
        the confidence at every step. If confidence
        ≥ required confidence, stop going deeper
        into this branch and mark it.
        Output: the unmarked subtree of
        LogicTree starting from currentLeaf
    end
```

Algorithm 1: The algorithm for generating the abstraction layer

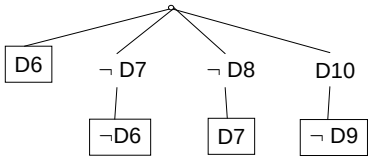
Research on learning Bayesian Networks from data [2, 9] focuses on finding the BN structure that best matches the collected data. These techniques are important in constructing the BN structure that represents the problem we are trying to solve. However, we need other methods to find an appropriate abstraction layer from the existing BN that, when transmitted from the remote agent, more efficiently conveys the necessary information to facilitate the local problem solving. In other words, this abstraction layer, when acquired, should be able to reduce the expected communication necessary to achieve the required confidence level.



(a) The action options for the *raw data action selection* approach



(b) The action options for the *all action selection* approach. The data that can be transferred are divided into two categories.

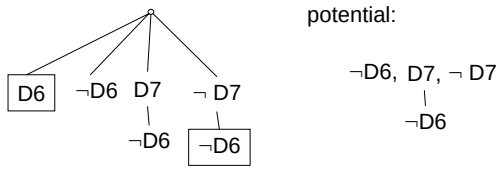


(c) The corresponding **LogicTree** to the abstraction data.

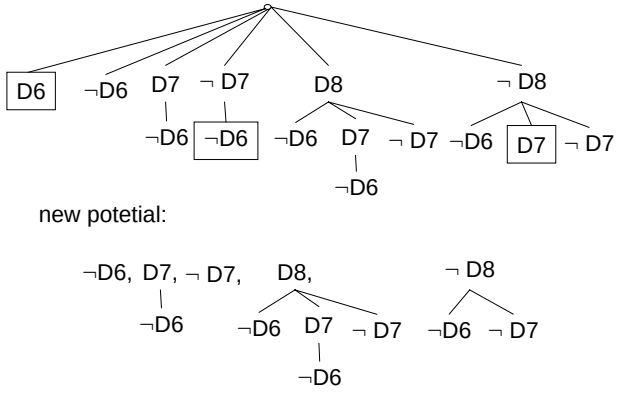
Figure 3: An example of the abstraction layer. Remote data includes $\{D6, \cdots, D10\}$. The required confidence level is 75%

We achieve this goal by developing an algorithm (Algorithm 1) that automatically generates an abstraction layer given a value combination of an agent's local data and the desired confidence level. The basic idea behind the algorithm is to find a set of logic expressions consisting of the remote data such that if at least one of the expressions is true the required confidence is reached. For example, an agent has 5 pieces of remote data $\{D6, \cdots, D10\}$ and needs to achieve a confidence level of 75%. In the original system, the only possible actions are to transfer some or all of these raw data. We will call this approach *raw data action selection*. However, if one or more of $D6$, $\neg D7 \wedge \neg D6$, $\neg D8 \wedge \neg D7$ and $D10 \wedge \neg D9$ are true, then the agent immediately has a local solution with a confidence level of 75%. These data carry more abstract information than the raw data themselves and therefore are more efficient, though the information is not as refined. If these data values can be acquired as well as the raw data, it can potentially yield huge savings on the communication cost. Additionally, the values of the abstraction data also give valuable information about what the values the raw data may have. For example, if the value of $\neg D7 \wedge \neg D6$ is transferred and is true, then the process can immediately terminate because the confidence level of 75% has been reached. If the value is false, even though the desired confidence level is not immediately achieved the agent does now have the knowledge that $D6$ and $D7$ cannot be false at the same time. This information can be retained by updating the BN. Figure 3(a) shows the action options of the *raw data action selection* approach for this example while Figure 3(b) shows the action options if the abstraction data can also be transferred.

The desired set of logic expressions can be generated by simply enumerating all of the possible combinations of the remote data values and selecting those that will enable the agent to reach its desired confidence level. Our algorithm improves efficiency by a logarithmic factor over the exponential brute force enumeration by reusing the subgraphs of logic expressions.



(a) **LogicTree** and **potential** before adding $D8, \neg D8$ to **LogicTree**.



(b) **LogicTree** and **potential** after adding $D8, \neg D8$ to **LogicTree**.

Figure 4: An example of the key step of Algorithm 1.

Figure 4 gives an illustration of an episode showing how the algorithm is run for the example in Figure 3. We use a tree, which we call a **LogicTree**, to keep track of all the data values examined so far. The nodes along the same path are connected by the $\wedge$ operator, and the different branches represent different value assignments. The marked branches are the ones that pass the confidence test, i.e., if true, the desired confidence level is reached. The set **potential** is used to keep track of the subtrees examined so far that did not pass the confidence test. Only the members of **potential** should be examined as a subgraph of longer expressions. Figure 3(c) shows the final **LogicTree** for this example.

If the values of all the remote data are acquired, the local confidence is guaranteed to reach 100%. Therefore, we limit the depth of the **LogicTree** to be less than the number of remote data. When the maximum depth is the number of remote data minus 1, the algorithm generates all of the expressions that have the desired property. Reducing the depth of the **LogicTree** results in a smaller number of abstraction nodes generated and therefore a smaller DEC-MDP. This tradeoff will be further discussed in the next section. In general, the number of abstraction data generated should be no more than the number of raw data. For each path in the **LogicTree**, we record the likelihood of the abstraction data represented by the branch being true, and we choose the n abstraction data with the highest likelihood of being true to incorporate into the DEC-MDP, where $n <$ the number of raw data.

When given a BN and a desired confidence level, the agent generates an abstraction layer for each raw data value combination and adds them to its action options for the states that have the corresponding raw data values. The expanded DEC-MDP can be solved to generate a communication strategy. We call this approach the *all data action selection*

approach. Since the raw data communication actions are competing with the abstraction actions, an agent chooses to transfer an abstraction data if and only if it will result in a lower expected communication cost than transferring any of the raw data. Therefore, the new system performance should be no worse than the original system in terms of the expected communication cost.

We compared the performance of the *all action selection* approach and the *raw data action selection* approach. The cost of sending a piece of abstraction data was equal to the cost of sending raw data. We used the Iterative Algorithm introduced in [11] to solve the DEC-MDP. We ran experiments on 100 problem structures with 2 high level events and 10 raw data (5 local to each agent) for different confidence levels. All of the networks were fully connected, which means that for both agents to have the complete evidence, 10 pieces of data needed to be transmitted. The corresponding curves in Figure 5 shows a comparison of the minimum expected communication cost generated by both systems. Column (a) in Table 1 shows the percent improvement in the expected communication cost when transmitting abstraction data in addition to the raw data. As shown, the *all data action selection* has a noticeable improvement over the *raw data action selection* approach. This illustrates that the addition of the abstraction data does help reduce the communication cost. The improvement is most significant when the required confidence is between 70% and 80%. When the confidence level is low, the expected communication cost is already low so that it is difficult to achieve a significant improvement. On the other hand, when the required confidence level is high, there are much fewer abstraction data generated resulting in a smaller improvement.

4. HIERARCHICAL ACTION SELECTION

Introducing the communication actions that transmit the values of the new abstraction data leads to both a larger state space and a larger action space for the generated DEC-MDP. As a result, the time needed to solve the new DEC-MDP can be significantly greater than the original DEC-MDP, which only has the raw data transmission as its actions. Column (c) in Table 1 shows the average time the *all data action selection* approach took to solve the DEC-MDP, where the average time needed for the *raw data action selection* approach equals 1.00. In this section, we introduce techniques that address this problem.

First we examine the case where the agents only transfer the abstraction data between them. We call this approach the *abstraction data action selection* approach. While the size of the DEC-MDP generated is often much smaller than that of the original DEC-MDP, one major drawback of this approach is that we can no longer guarantee that the required confidence level can be reached. Only when at least one of the abstraction data is true will the desired confidence level be reached. The corresponding curve in Figure 5 shows the expected communication cost achieved for the cases where the desired confidence level can be reached. When the required confidence level is high, there are fewer abstraction data generated, and therefore the expected cost is much lower than those of the two approaches we discussed in the previous section. Notice, however, that this is a false savings as the desired confidence level cannot always be reached, which is not reflected in the figure.

We seek to combine the advantages of the *all data action selection* approach and the *raw data action selection* approach so that we can save time on solving the DEC-MDP as well as guarantee the required confidence level. We achieve this by restricting legal actions for different states. Before an agent has acquired all of the abstraction data, the agents are not allowed to transfer any of the raw data. We call this approach the *hierarchical action selection* approach. An agent prefers to transfer abstraction data because it abstracts from multiple pieces of raw data and thus is a more efficient information carrier. Only when the acquisition of all of the abstraction data cannot achieve the desired confidence level will an agent start to acquire the raw data in order to get the necessary information.

Like the *all data action selection* approach, the *hierarchical action selection* approach guarantees that the desired confidence level can be achieved. Figure 5 illustrates the comparison of the performance of the approaches we have discussed. As we expected, the *all data action selection* approach performs the best. In fact, it generates the optimal solution for any DEC-MDP whose action space in-

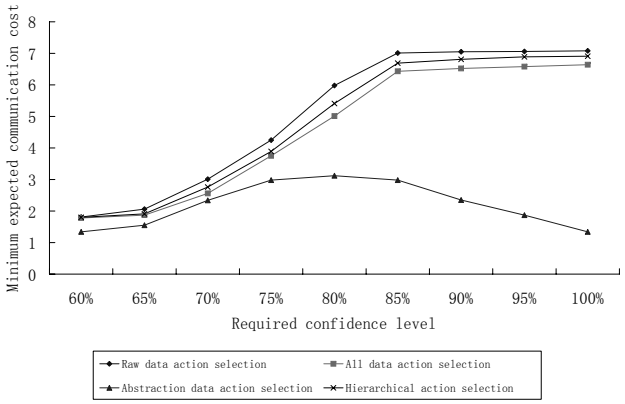


Figure 5: A comparison of the minimum expected communication cost given different action selections

required confidence	(a)	(b)	(c)	(d)
60%	1.66%	0.55%	1.25	0.89
65%	9.22%	7.28%	1.53	0.77
70%	14.95%	8.31%	1.49	0.69
75%	11.76%	8.47%	1.52	0.63
80%	16.22%	9.53%	1.61	0.59
85%	8.27%	4.56%	1.41	0.65
90%	7.52%	3.40%	1.21	0.77
95%	6.80%	2.41%	1.10	0.82
100%	6.21%	2.40%	1.09	0.87

Table 1: Performance compared to raw data action selection approach for single abstraction layer in a 2-10 network. (a) Expected communication cost improvement of *all action selection*. **(b) Expected communication cost improvement of** *hierarchical action selection*. **(c) Time needed to solve the DEC-MDP for** *all action selection* **normalized by that for raw data action selection. (d) Time needed to solve the DEC-MDP for** *hierarchical action selection* **normalized by that for raw data action selection.**

cludes transferring both the abstraction data and the raw data. On average, the *hierarchical action selection* approach outperforms the *raw data action selection* approach. However, there are cases where the *hierarchical action selection* approach requires more communication than the *raw data action selection* does. In those BNs, it is often the case that there is a low likelihood of any of the abstraction data being true. Column (b) in Table 1 shows the amount of improvement in the minimum communication cost the *hierarchical action selection* approach gains over the *raw data action selection* approach. The pattern is similar to that of column (a).

Column (d) in Table 1 shows the average time needed to solve the DEC-MDP for the *hierarchical action selection* approach normalized by the average time needed for the *raw data action selection* approach. It achieves substantial savings. Even though the *hierarchical action selection* approach does not reduce the size of the action space compared to the *all action selection* approach, it does reduce the number of legal actions available to any given state. This also decreases the size of the state space because H, the communication history, has fewer possibilities. These two factors combined together contribute to the time savings, and the larger the network is, the more substantial the savings should be. We plan to run more experiments on larger networks to verify this observation.

All of the experiments we have shown so far set the depth of the **LogicTree** in Algorithm 1 to be the number of remote data minus 1, i.e., 4. A shallower **LogicTree** will generate fewer abstraction data and result in a higher expected communication cost to reach the same confidence level. However, generating the abstraction layer and finding the communication strategy will take less time. Our future experiments will explore this tradeoff between computation time and communication cost as the depth is varied.

5. MULTIPLE LEVELS OF ABSTRACTION

The idea of varying the depth of the **LogicTree** leads to an interesting way of generating a hierarchy of abstraction data. We can modify Algorithm 1 to generate a set of abstraction data of different lengths. The more literals in the abstraction data, the more abstract it is, the more efficiently it carries information, and the less refined the information is.

Just like in the single abstraction layer case, there are different ways of incorporating multiple layers of abstraction data into the action space of the DEC-MDP. We did experiments on the all data action selection approach and the hierarchical action selection approach, and compared their performance to that of the raw data action selection approach. In the all data action selection approach, all the abstraction data of various length are added to the action space to compete with the raw data. In the hierarchical action selection approach, the agents transfer the most abstract data first. If the confidence level is not achieved, then the agents start transferring the next level of abstract data. This process goes on until the confidence level is reached.

We tested these approaches on networks of different sizes: 4 high level events and 20 low level data (4-20 networks); 6 high level events and 30 low level data (6-30 networks); 8 high level events and 40 low level data (8-40 networks). We generated two abstraction layers for each of the networks. Figure 6 shows the percentage of the networks we tested on

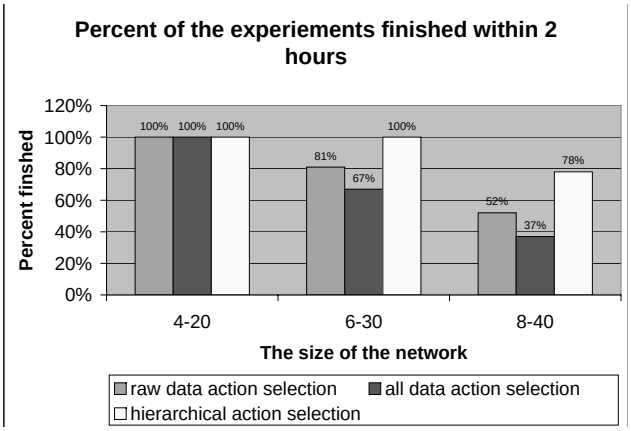


Figure 6: The percentage of the problems tested that each approach was able to finish solving within 2 hours.

required confidence	(a)	(b)	(c)	(d)
60%	3.87%	2.14%	1.33	0.91
65%	10.42%	8.19%	1.49	0.79
70%	16.01%	12.34%	1.52	0.62
75%	19.84%	15.62%	1.59	0.56
80%	18.26%	13.43%	1.71	0.49
85%	12.22%	9.53%	1.59	0.58
90%	8.49%	6.72%	1.39	0.68
95%	5.71%	4.43%	1.19	0.76
100%	4.23%	3.12%	1.10	0.78

Table 2: Performance compared to the raw data action selection approach for multiple abstraction layers in a 6-30 network. (a) Expected communication cost improvement of *all action selection*. (b) Expected communication cost improvement of *hierarchical action selection*. (c) Time needed to solve the DEC-MDP for *all action selection* normalized by that for raw data action selection. (d) Time needed to solve the DEC-MDP for *hierarchical action selection* normalized by that for raw data action selection.

that each of the three approaches was able to finish within two hours. It shows that hierarchical action selection approach was able to finish the policy search better than either of the two other approaches. The larger the network, the bigger an advantage the hierarchical approach has over the other two. Table 2 shows the performance comparison of the all data action selection approach and hierarchical action selection approach over the raw data action selection approach. The data is shown only for the 6-30 networks for which all of the three approaches finished searching for the optimal policy. The first abstraction layer is composed of the abstraction data with no more than 5 literals, and the second layer is composed of the abstraction data with no more than 10 literals and greater than 5 literals. The results show a similar pattern as those of the single abstraction layer experiments. The all data approach is not practical for large networks or more than a few abstraction layers, since it substantially expands both of the state space and action space of the DEC-MDP. However, it does give a

lower expected communication cost than the raw data action selection approach when it can solve the problem. On the other hand, the hierarchical approach proves to be more effective than both the raw data action selection approach and the all data action selection approach. It is able to solve more and larger networks, and on average it takes much less time. Furthermore, for the networks that both approaches were able to finish the policy search, the solution generated by the hierarchical action selection approach has a lower expected communication cost on average than that generated by the raw data action selection approach.

6. CONCLUSIONS AND FUTURE WORK

In this paper we investigated the techniques of transferring abstraction data in addition to raw data in Distributed Bayesian Networks to reduce the required communication cost. We introduced an algorithm that automatically generates appropriate abstraction data that facilitates the achievement of the required confidence level and reduces the necessary communication cost. We also discussed approaches to incorporate the new abstraction data into the DEC-MDP framework effectively. Both the improvement in the minimum expected communication cost and the time savings in solving the DEC-MDP make the *hierarchical action selection* an attractive approach, especially for the systems which require a mid-ranged confidence level. We further extended the different action selection approaches to larger networks and multiple abstraction layers. The hierarchical action selection approach was shown to be able to solve problems with larger networks than the other approaches.

This work allows us to look at the use of abstraction to reduce communication cost from a formal perspective. The *hierarchical action selection* approach defines when and how the abstraction data and the raw data should be transferred in the communication process. The addition of transferring abstraction data to the communication actions can be extended to reduce the communication cost of other Cooperative Distributed Problem Solving (CDPS) applications as well. Though the algorithm that we used to generate the abstraction data is specific to the DBN used to represent the DI problems, given a CDPS problem with specific semantics, there often is abstraction data that can be generated and used to transfer information more efficiently. The hierarchical action selection approach then can be used to incorporate this abstraction data to further minimize the communication cost.

We plan to do more experiments with larger networks, and we predict that the savings of the *hierarchical action selection* approach shown in this paper will be more significant for larger networks. The influence of different depths of **LogicTree** in Algorithm 1 on the system performance will also be investigated. In this work, we looked at one particular type of abstraction. There are other forms of abstraction that can be potentially useful. For example, some abstraction data may contain useful information but is not sufficient to reach the confidence by itself. Other abstraction may be an intermediate interpretation result that can be incorporated into the local data. We plan to look into these different types of abstraction in future work. We also intend to investigate other techniques in generating multiple layers of abstraction data, such as different levels of intermediate interpretation results.

7. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant Nos. IIS-0219606 and IIS-0414711. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

8. REFERENCES

[1] D. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, November 2002.

[2] W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210, 1996.

[3] N. Carver and V. Lesser. The DRESUN testbed for research in fa/c distributed situation assessment: Extensions to the model of external evidence. In *Proceedings of ICMAS-95*, pages 33–40, 1995.

[4] N. Carver and V. Lesser. Domain monotonicity and the performance of local solutions strategies for CDPS-based distributed sensor interpretation and distributed diagnosis. *International Journal of Autonomous Agents and Multi-Agent Systems*, 6:35–76., 2003.

[5] S. H. Chung and A. Barrett. Distributed real-time model-based diagnosis. In *Proceedings of the 2003 IEEE Aerospace Conference*, 2003.

[6] B. J. Clement and E. H. Durfee. Theory for coordinating concurrent hierarchical planning agents using summary information. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 495–502, 1999.

[7] L. Erman, F. Hayes-Roth, V. Lesser, and D. Reddy. The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty. *Computing Surveys*, 12(2):213–253, June 1980.

[8] C. V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS03)*, 2003.

[9] D. Heckerman. A tutorial on learning with bayesian networks. In *Learning in Graphical Models*. MIT Press, Cambridge, MA, 1999.

[10] R. Nair, M. Roth, M. Yokoo, and M. Tambe. Communication for improving policy computation in distributed pomdps. In *Proceedings of The Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-04)*, 2004.

[11] J. Shen, V. Lesser, and N. Carver. Minimizing Communication Cost in a Distributed Bayesian Network using a Decentralized MDP. In *Proceedings of Second International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2003)*, volume AAMAS03, pages 678–685, Melbourne, AUS, July 2003. ACM Press.

Toward an Understanding of the Impact of Software Personal Assistants on Human Organizations

Steven Okamoto
Carnegie Mellon University
sokamoto@cs.cmu.edu

Paul Scerri
Carnegie Mellon University
pscerri@cs.cmu.edu

Katia Sycara
Carnegie Mellon University
katia@cs.cmu.edu

ABSTRACT

Intelligent software personal assistants for human organizations are an active research area within the multiagent community. However, while many capabilities for these software personal assistants are imagined or already developed, there has been no quantification of how an organization's performance is improved by software personal assistants. Moreover, while intuitively organizations will adapt to take advantage of the new technology, there has been no work looking at how organizations should or will change in response to the new technology. This paper presents a first step toward addressing this oversight. Specifically, a computational model of the working of an organization and how software personal assistants will affect that organization is developed that allows effects of software personal assistants to be modeled. By varying the potential capabilities of the software personal assistants and the structure of the organization, we can explore the impact of the technology. Our results show that managing task contingencies can greatly improve organizational performance by as much as 45%.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems*

Keywords

Organizational Design, Personal Software Assistants

1. INTRODUCTION

A variety of recent agent research has focused on developing techniques for building intelligent software personal assistants (SPAs) for humans in complex organizations. Each human in the organization will have his or her own SPA which is anticipated to have a wide variety of capabilities from scheduling joint activities[19, 23, 11, 21], monitoring and reminding individuals of key timepoints[7, 4], sharing key information[23, 20], assisting in negotiation decision support[14], and even ordering lunch[7]. Just as previous technological innovations have revolutionized human organizations, SPAs are likely to revolutionize the way organizations ranging from first responders to the military to commercial enterprises organize their joint activities. However, despite much research into enabling technologies, there have been no previous efforts to understand how human organizations might utilize the technology. Moreover, it is not even well understood which SPA capabilities are most useful for improving the effectiveness and efficiency of an organization.

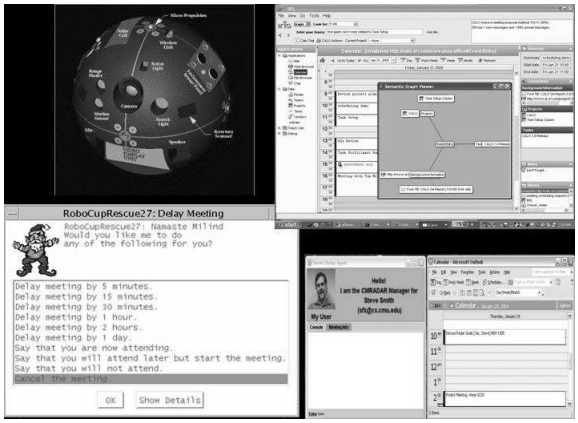


Figure 1: A sampling of SPA technology. Clockwise from upper right: CALO, RADAR, Electric Elves, PSA.

Outside of the multiagent community, organizations and their adaption to technological change has been extensively studied. Some research, particularly within economics, uses abstract analytical models to determine optimal features of an organization given the task for which it is designed[12, 10, 3]. Other work, often in sociology, attempts to develop analytic models to explain observed phenomena[5, 17]. Unfortunately, such analytic models are typically too abstract to model the subtle effects of SPA technology. One sub-field of research focuses on design and adaption of human organizations[5], but has not looked specifically at the impact of new technology on those organizations. Within the multiagent community, researchers such as Dignum[9], Horling[13] and Dastani[8], have developed interesting techniques for modeling organizations. As these languages mature they will play an important, complementary role to this work. Hence, while organizations are extensively studied, no work specifically allows quantification of the impact of SPAs on an organization.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

In this paper, we present our initial efforts at modeling the impact of SPAs on human organizations. The aim is to provide SPA developers with input on which capabilities are most likely to be useful and to provide SPA adopters with input on how to best apply the technology. Specifically, we are interested in organizations configured for a particular large scale task, e.g., response to a major disaster or a military operation, where SPAs such as the DARPA COORDINATORS assistants[23] may be deployed. Since SPA capabilities typically affect detailed work activities and their broader influence is unclear, we must model the organization at a fine grained level of detail, capturing, for example, message paths, decision making, times, etc, in order to see their effects. An abstract simulation environment takes proposed organizations and tasks to be performed by that organization and computes key properties of task execution, including how well and how quickly the organization performed the task and how robustly it handled individual failures. The simulation captures important aspects of the operation of the organization, such as non-determinism and individual cognitive limits, but abstracts away domain level details, making it feasible to evaluate many instances.

In our computational model, SPAs are represented as changing and relaxing constraints on human cognition. With reference to ongoing projects and previously published literature about SPA technology, we identified four key capabilities that are being developed and inferred how these would alter how a person operates in an organization. For example, several key technologies[23, 4] will have the effect of increasing the speed with which an individual in an organization will be able to make decisions. The model of the organization allows the impacts of capabilities to be considered individually or in combination.

To quantify the impact of SPAs on an organization, we generated a large number of different organizational structures and evaluated the performance of each organizational structure with and without combinations of SPA capabilities. This general approach was used for two reasons. First, it enabled us to quantify the impact of SPAs on the organization by measuring the difference in performance between organizations with and without SPAs. Specifically, we found that managing contingencies to prevent actor overload improved performance most significantly, allowing a 45% performance improvement over an organization without SPAs. Second it allowed us to compare the performance of different organizational structures to determine what changes may be needed to leverage SPA technology. We found that the types of hierarchies currently used by many organizations remains effective in the presence of SPAs.

2. RELATED WORK

The effect of information technology on human organizations has been a focus of considerable research, especially in business and organization theory[5, 17, 1]. While it is generally assumed that the deployment of such technology in an organization reduces coordination costs and enhances performance[16], other research[6] indicates that the adoption of information technology can have unexpected and deleterious effects. It is thus essential to consider the interactions between proposed technology and the dynamics of the organization deploying that technology.

Even in cases where the adoption of information technology improves organizational efficiency, the underlying organizational structure must often change in order to realize the full benefits. Two notable and well-studied such technologies are electronic marketplaces in supply chain management[17] and e-mail for communication and coordination[1]. While the organizational adaptations needed to leverage those technologies are now understood, there is to date no research on the organizational changes necessary to most fully exploit the newer technology of SPAs. Research on SPAs has instead focused on issues of actually building and deploying such complex software systems, in domains ranging from military applications[23] to disaster response[20] to office scheduling and collaboration[7, 19] to space environments[4]. Studies involving agents actually assisting human teams perform tasks have been performed[22, 7], but such an approach is impractical for studying the effects of agents on large organizations, and also require the SPA to already be built.

Organizational research in the multiagent community has tended to focus on modeling organizations in order to guarantee logical properties during execution[9]. While important for a complete understanding of organizational behavior and the successful integration of agents into human organizations, this approach does not specifically address the impact of SPAs, and also neglects quantitative evaluation of organizational performance. Recent work[13] addresses the issue of quantitative evaluation, but is focused primarily on organizational modeling as opposed to design, and still does not specifically take into account SPAs.

3. PROBLEM

We represent an organization of human actors that perform tasks and communicate in order to make decisions. Successful execution of sensing tasks yields information that is used as input to making decisions. This abstraction describes a wide variety of real-life organizations, from military units that detect and eliminate enemy threats, to businesses that identify customer needs and reallocate investments.

A number of assumptions govern the behavior of actors in our model:

- An actor that attempts to perform a task may *fail*. The actor cannot try to perform the task again.
- Tasks have fixed execution durations during which time they must be performed, uninterrupted.
- Actors can perform only a limited number of tasks simultaneously. In particular, an actor can only perform a single sensing task at a time, or 3 decision-making tasks at a time. Actors cannot simultaneously perform sensing and decision-making tasks. An actor who attempts to exceed these limits becomes overloaded and cannot perform *any* tasks.
- Actors can only transmit information along predetermined communication links.
- Actors are bounded in the number of communication links they can maintain. Each actors can have at most max_{in} incoming communication links and max_{out} outgoing communication links.
- Communication takes a fixed amount of time.
- Actors can only perform decision tasks if they have all the required information.

We do not explicitly represent SPAs in the organization, but instead model their effects as changes in organizational constraints and parameters. For example, the presence of SPAs may be modeled by increasing the communication speed from a baseline for an organization without SPAs.

Formally, an organization O is a tuple

$$O = \langle A, T, I, Prov, Req, Cap, Fail, G, speed_d, speed_c, M, d \rangle$$

with the following components:

- $A = \{a_1, a_2, \ldots, a_m\}$: the set of human actors in the organization
- T: the set of tasks to be performed by the actors. Tasks are partitioned in two orthogonal ways:
  - $T = T_P \cup T_C$
    - $T_P = \{t_1, t_2, \ldots, t_n\}$: the set of *primary tasks* to be performed.
    - $T_C = \bigcup_{t_i \in T_P} C_i$: the set of *contingencies* to be performed in case of task failures.
  - $T = T_D \cup T_S$
    - T_D: the set of *decision-making tasks* that require information to be performed.
    - T_S: the set of *sensing tasks* that provide information when successfully performed.
- $I = \{I_1, I_2, \ldots, I_l\}$: the set of information produced by sensing tasks and used by decision tasks.
- $Prov : T \rightarrow \mathcal{P}(I)$: the information provision function mapping (sensing) tasks to the set of information provided by successfully completing the (sensing) task.
- $Req : T \rightarrow \mathcal{P}(I)$: the information requirement function mapping (decision-making) tasks to the set of information required to perform the (decision-making) task.
- $Cap : A \times T \rightarrow [0, 1]$: the capability function indicating how well actors perform tasks.
- $Fail : A \times T \rightarrow [0, 1]$: the failure function indicating the probability actors will fail at performing tasks when attempted.
- G: the directed graph with actors as vertices and directed edges indicating possible communication paths between actors.
- $speed_d$: the rate at which an actor can make a decision.
- $speed_c$: the speed of communication.
- M: the assignment matrix of primary tasks and contingencies to actors. Each primary task and contingency must be assigned to exactly one actor.
- d: the deadline by which time all decisions must be made in order for the organization to succeed.

For each primary task $t_i \in T_P$, there exist 0 or more contingencies, $C_i = \{c_i^1, c_i^2, \ldots, c_i^{k_i}\}$. The contingencies are pre-determined backup tasks that are invoked in case of task failure, and are ordered by preference. Hence contingency c_i^1 must be performed if primary task t_i fails, and contingency c_i^2 must be performed if contingency c_i^1 fails, and so on. The set of all contingency tasks is written $T_C = \bigcup_{t_i \in T_P} C_i$.

3.1 An Organization Model

The directed graph G in the specification of the organization describes the structural communication links that exist between members of the organization. The vertices of G are the actors in A. A directed edge from $a_i \in A$ to $a_j \in A$ indicates that a_i can transmit information to a_j. A path through G must exist from one actor to another when the first supplies information to the second, i.e., a path must exist from two actors a_i and a_j if $M(a_i, t) = 1$ and $M(a_j, t') = 1$ and $Prov(t) \cap Req(t') \neq \emptyset$. Communication will actually occur on the edges of G if it falls on a path required for delivering information.

Notice that tasks in T_C may not actually be performed, hence edges in G that only transmit information for these tasks may not actually be used. Moreover, even those edges that are on paths required for tasks in T_P may not be used if the actor performing the task fails, since a contingency would then be invoked. Hence we distinguish between two graphs, G and $\bar{G}$. We refer to G as the potential structure graph, since it must contain all the edges that could be required for any possible combination of tasks and contingencies. $\bar{G}$, on the other hand, is called the *instantiated structure* and only contains those edges that are actually being used given the current set of tasks and contingencies.

Human cognitive limitations restrict the number of contacts a person can effectively handle. We model this as constraints on the potential structure graph G. In particular, we bound the in- and out-degrees of the actors in G: $\forall a \in A$, $indegree(A) \leq max_{in}$ and $outdegree(A) \leq max_{out}$.

3.2 Execution

The actors attempt to perform primary sensing tasks assigned to them in M. For any primary task failure, contingencies are invoked as described in section 3. If an actor a succeeds at a sensing task t_s (whether it is a primary task or contingency), the actor gains information $Prov(t_s) \subseteq I$. This information has an associated quality which we denote by $Qual : I \rightarrow [0, 1]$. For all $I_j \in Prov(t_s)$, we let

$$Qual(I_j) = Cap(a, t_s) \in [0, 1] \tag{1}$$

The actor then transmits the newly learned information to any neighbors as defined in G. These neighbors store the information in their personal knowledge bases and also in turn propagate the information to their neighbors.

When an actor assigned a decision-making task gains all the information required to make the decision, either from other actors or by performing the sensing tasks itself, it attempts to perform the decision (which can fail). If the actor succeeds, the decision is made with a quality that depends both on the capability of the decision-making actor and the quality of the provided information, and the organization gets reward for making the decision. This reward is denoted $Reward : T_P \cap T_D \rightarrow [0, 1]$. For a decision-making task t_d successfully performed by actor a, if $t_d = t_i \in T_P$ or $\exists t_i \in T_D$ with $t_d = c_i^r \in T_C$, reward is given by

$$Reward(t_i) = Cap(a, t_d) \left(\sum_{I_j \in Req(t_d)} Qual(I_j) \right) \tag{2}$$

where $Qual(I_j)$ is defined above in equation 1. The total reward earned by the organization will be one measure of organizational performance, as described next.

3.3 Objective Function

The "performance" of an organization is complex and multi-attributed. In this work, we focus on three specific measures.

- *Success Rate* This is a measure of how well the organization handles individual failure and is calculated as the fraction of the time the organization makes all of the decisions represented in T_D before the deadline d.
- *Reward* This is a measure of how well the organization makes decisions when it succeeds, and is calculated:

$$\sum_{t_d \in T_D \cap T_P} Reward(t_d)$$

  where $Reward(t_d)$ is given by equation 2 in section 3.
- *Speed* This is a measure of the organization's efficiency and is calculated as the difference between the deadline time d and the time at which the last task finishes execution.

The overall objective function used for performance is a weighted sum of these three metrics.

4. EXPERIMENTAL SETUP

We evaluated a large number of organizations with a variety of organizational structures. The basic experimental setup was as follows:

- 20 human actors.
- 44 total primary tasks and contingencies.
  - 4 decision tasks, each requiring information provided by 3 distinct sensing tasks.
  - Each decision task had 1 contingency.
  - Each sensing task had 2 contingencies.
- Performance function: each metric (success rate, reward, and speed) was normalized prior to weighting. The weights were:
  - Success rate weight = 5
  - Reward weight = 1
  - Speed weight = 1

We tested 9 types of organizational structures. These 9 types of structures can be grouped into 4 broad classes:

- Ring. Actors form a a singly linked cycle. If actors can link to more than one actor (i.e., $max_{out} > 1$), actors form at regular intervals across the ring to minimize the average path length to every actor.
- Regular hierarchy. Actors form a full n-ary rooted tree, where n is the span of control. We tested hierarchies with three spans of control:
  - RH3: regular hierarchy with span of control of 3.
  - RH4: regular hierarchy with span of control of 4.
  - RH5: regular hierarchy with span of control of 5.

  Every actor in a regular hierarchy has one outgoing link and a fixed number of incoming links, so this structure is invariant to changes in max_{out} and max_{in}. Note, however, that max_{in} must be greater than or equal to the span of control for a regular hierarchy to be allowed.
- Scale free network. These were generated according to [2]. A small initial set of actors were chosen and the remaining actors were added sequentially, with each adding max_{out} directed links to the actors already in the graph. Link formation was done preferentially to actors with higher connectivity. No actor was allowed to have more than max_{in} incoming links.
  - SF: linking preference was proportional to the in-degree of the target actor.
  - SFpreR1: each actor in the initial set added an edge at uniformly at random to another actor in the initial set before generation began. Preferential linking was by in-degree.
  - SFB: linking preference was proportional to the total degree (in-degree + out-degree) of the target actor
  - SFBpreR1: each actor in the initial set added an edge at uniformly at random to another actor in the initial set before generation began. Preferential linking was by total degree.
- Random graph (Rand). max_{out} links were added randomly from each actor to other actors. No actor was allowed to have more than max_{in} incoming links.

4.1 Baseline Performance

In the baseline configuration, $max_{out} = 1$ and $max_{in} = 4$. Note that this meant that actors in ring structures did not have any links across the ring. In addition RH5 could not be used in the baseline configuration since actors in RH5 must have 5 incoming links, which exceeds the constraint on max_{in}.

Figures 2 – 4 show organizational performance as described in section 3.3. Figure 2 shows the success rate component, Figure 3 shows the reward component, and Figure 4 shows the speed component. Baseline performance is depicted by black columns labeled "No SPAs".

Observe that the regular hierarchies show the highest success rates in Figure 2. This is because there is a high congruence between hierarchical structure and the decision making problem we are considering: actors lower in the hierarchy perform sensing tasks, and they transmit information upward to decision-making actors. The structure of the organization naturally constrains the flow of information to the actors needing that information. The root actor will eventually have all information gained by actors in the organization, and is thus well placed to make decisions or to perform contingencies for decision-makers lower in the hierarchy.

In contrast, ring structures show the lowest success rate. This is because of the lack of congruence between the ring structure and the problem. Because the ring structure is strongly connected, all actors eventually gain any information gained by any actor. However, there is no structural force to encourage the specialization of actor, and so many actors overload by attempting to simultaneously perform sensing contingencies and make decisions.

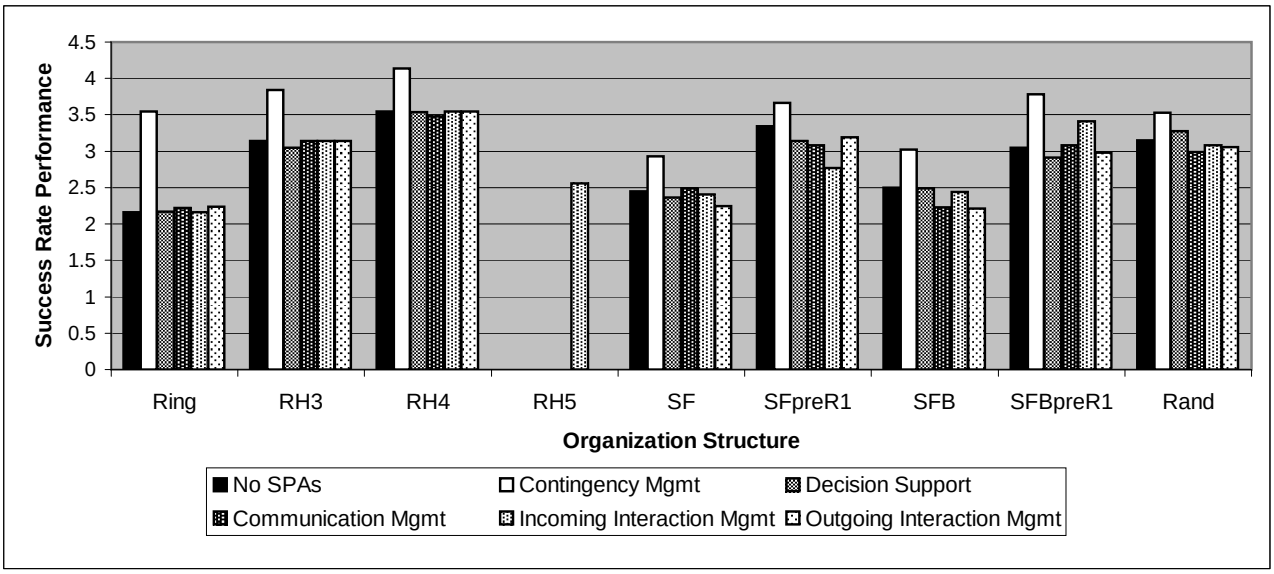


Figure 2: Success rate contribution to performance.

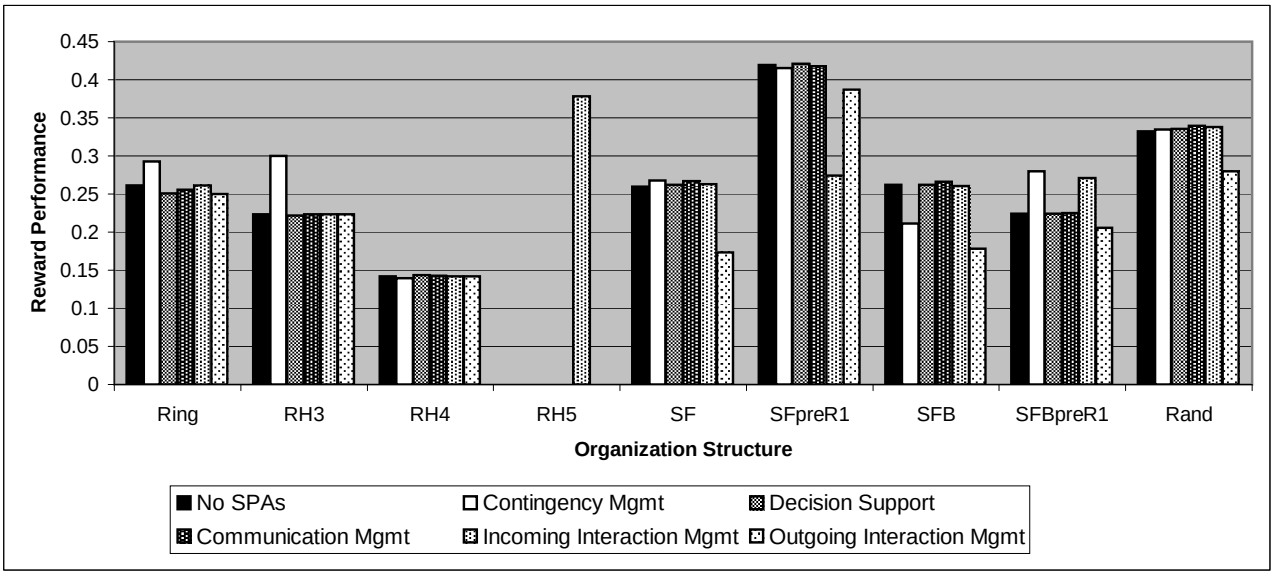


Figure 3: Reward contribution to performance.

Ring structures also show the lowest speed in Figure 4. This is because actors are only allowed a single outgoing link. Thus there are no edges crossing the cycle. As a result, the diameter of the ring structure is linear in the number of actors, while it is logarithm in the number of actors for the other structure types.

Results presented in the next section will detail how performance changes in the presence of SPAs.

5. MODELING PERSONAL ASSISTANTS AND RESULTS

The capabilities of SPAs are generally being designed to overcome perceived cognitive limitations of humans that are thought to limit the performance of an organization[22, 23, 4]. In this section, we describe four such limitations and the models of SPA capabilities — contingency management, decision support, communication management, and interactor interaction management for information sharing — that might overcome these limitations. We also present experimental results showing the effect of these capabilities on organizational performance in figures 2 – 4.

Many other SPA capabilities (and corresponding cognitive limits) might have been modeled. We focused on this initial set because there was active research in the field on developing these capabilities. Moreover, we selected SPA capabilities that appeared most suited to the special purpose organizations that are the focus of this paper. Future work will expand the scope of SPA capabilities, including to those where there is not yet active research.

5.1 Managing Task Contingencies

An SPA may manage contingencies in order to prevent its actor from being overloaded. Instead of invoking contingencies only when a domain-level failure is suffered, the SPA may monitor the actor's current status and workload and automatically invoke contingencies for any tasks that would overload the actor. Similar approaches have been developed or proposed for SPAs in other environments[7, 15]. Intuitively, these SPA capabilities should make the organization more robust to individual failure. Moreover, we hypothesize that it will be possible for the organization to arrange itself such that more capable team members will be involved in more contingencies, since their task performance provides the most reward to the whole organization. Managing contingencies or assigning roles has been a key area of research, but this is the first work that attempts to quantify the benefit of such a capability on an organization and understand how the organization may change to leverage this capability.

The results of adding contingency management capabilities to the simulated organizations are shown in figures 2 – 4 in the white columns labeled "Contingency Mgmt". From Figure 2, it is clear that contingency management greatly improved the success rate for all tested organization structures. It was particularly effective in improving the success rate of the ring structure, which is especially prone to actor overload: the success rate of the ring structure improved by 64%, while the success rate of RH4 improved by only by only 16%. This large gain translated into an overall gain in performance of 45% for the ring structure, against an average performance gain of 17% for all structures.

As shown in Figure 3, the addition of SPA-managed contingencies also improved the reward obtained by some of the organizations, in accordance with our hypothesis. For example, reward increased by 24% for SFBpreR1 and 34% for RH3. The only structure that lost reward was SFB, which saw reward decrease by 20%.

5.2 Decision Support

As introduced above, a key cognitive constraint on humans in an organization is the number of decisions that can be effectively made in a limited amount of time. SPAs could relax this constraint in a variety of ways, e.g., taking over routine tasks to provide more time for "decisions"[18, 20] or collating and presenting information in a way that speeds up the actual decision-making process[15]. Without reference to specifically how it is done, in the computational model of the organization, we simply allow a person to effectively make more decisions in the same period of time, if they have access to SPA technology, by doubling $speed_d$, the speed of making decisions.

The performance of organizations with this capability is shown in figures 2 – 4 in the columns labeled "Decision Support". We expected that this capability would impact the success rate and the speed of execution, It would affect the success rate because each decision would have execute more quickly, thereby reducing the amount of time that the actor may be overloaded. It would also increase the speed of execution directly. Both these factors turned out not to have noticeable impacts of performance because the decision duration was already quite low, especially in comparison to the deadline (which was used to normalize the speed metric). When modeling other problems this capability may yet prove to be significant and merits further study.

5.3 Communication Management

Wired and wireless networks upon which much communication in an organization travels is amazingly fast in contrast to human communication speeds. However, in most cases, people are still required to input information and subsequently process received information. The input and output time is subject to human limitations and is fast becoming (if it is not already) the bottleneck in intra-organization communication, and could be alleviated by SPAs.

We modeled this capability by doubling the communication speed in the organization. The effects on performance can be seen in figures 2 – 4 in the columns labeled "Communication Mgmt". This capability had little effect on either success rate or reward. As shown in Figure 4, it also had only small but noticeable effect on the speed of most organizations, increasing average speed by 8%, because they already displayed efficient communication. As expected, however, this capability had a substantial impact on ring-structured organizations, with a speedup of 29%.

5.4 Managing Interactions for Information Sharing

The rate at which decision-makers in an organization can effectively make decisions is limited by two key factors: (i) their ability to get the appropriate information to make required decisions and (ii) their innate processing speed for making those decisions (see section 5.2). The cognitive load of managing incoming information from others in the organization and directing outgoing information to others is a constraint that may be relaxed by SPA technology[15, 17]. Specifically, a decision-maker may be able to handle input

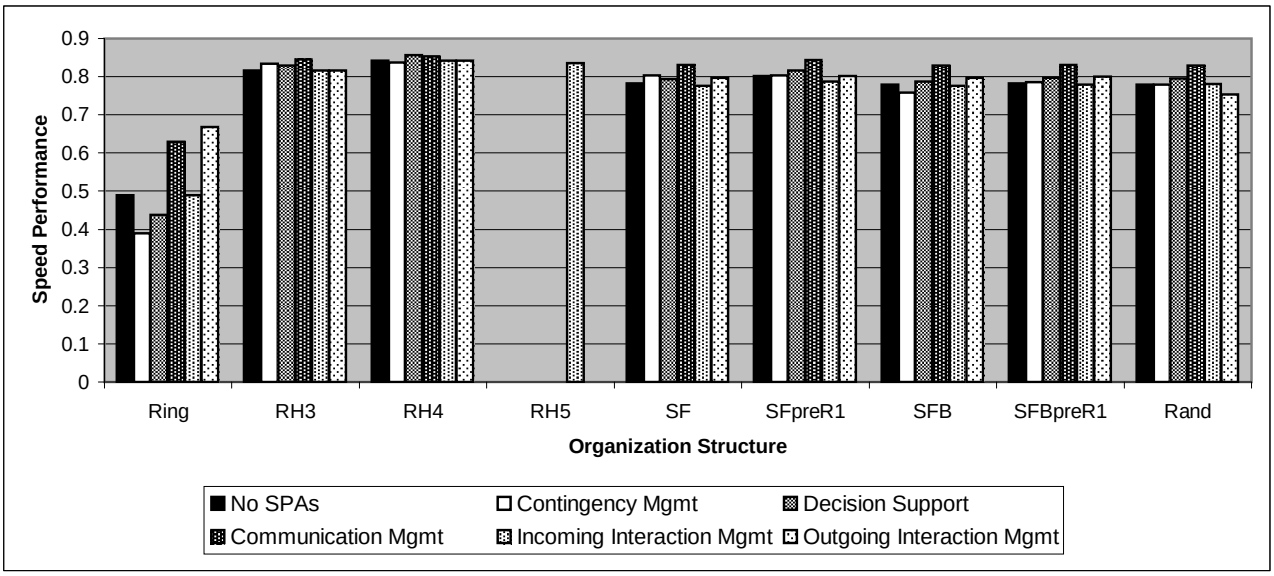


Figure 4: Speed contribution to performance.

from a greater number of members of an organization and hence, either get more direct access to that information or make more decisions. For example, the CEO of an organization needs information from all over parts of the organization to make key strategic decisions. However, it is cognitively and organizationally infeasible for a person to directly receive input from a large number of people[5]. Hence, in the case of the CEO, that information must pass through a small number of department heads, both delaying the information and potentially distorting it. Conversely, suppliers of information are limited in how many others they can provide information to, requiring organizational structures that channel information and introduce delays and distortion. SPA technology could allow information suppliers to more directly communicate with the decision-makers requiring their information.

The effect of managing interactions was modeled by changing the values of max_{in} and max_{out}. max_{in} was increased from 1 to 2, and max_{out} was increased from 4 to 5. We considered the two effects separately, with the resulting performance shown in figures 2 – 4 by the columns marked "Incoming Interaction Mgmt" and "Outgoing Interaction Mgmt", respectively.

It is important to note that unlike the other capabilities tested, interaction management related directly to the organizational structure by allowing actors to form more links. Some structural classes were completely unaffected by such changes. For example, in a regular hierarchy, each actor has a single outgoing link, and a fixed number of incoming links (bounded above by max_{out}), and so changing max_{out} had no effect on regular hierarchy performance. Similarly, increasing max_{in} had no effect on the ring structure performance. However, increasing max_{out} allowed actors in the ring structure to add links crossing the cycle, which reduced the diameter of the graph by half. This translated into impressive gains in speed of 36%, as shown in Figure 4.

The increase of max_{in} from 4 to 5 allowed us to test a new regular hierarchy, RH5, in which actors had a span of control of 5. This structure had a poor success rate, lower than that of both RH3 and RH4, as shown in Figure 2. This is because the hierarchy of RH5 was too shallow; there were too few interior actors to make all the decisions and so they are being overloaded.

5.5 Multiple Capabilities

We also tested the organizations with different combinations of SPA capabilities. Because there are four capabilities (those described in subsections 5.1 – 5.4), with information sharing including two distinct effects, there are a total of 32 distinct combinations of capabilities. Figure 5 shows the performance breakdowns of the SFpreR1 scale free structure for all combinations of our SPA capabilities; the number of capabilities in the combination increases along the x-axis, from no capabilities on the left to all capabilities on the right. It is clear that not all combinations of capabilities have equal effects on performance, and many (including the combination of all capabilities) even negatively impact performance. This emphasizes the need to carefully select SPA capabilities to fit the organization.

Several of the effects seen in the individual capability experiments were evident in this experiment as well. For instance, managing contingencies to prevent actor overload provided substantial gains in organizational performance. Also, managing interactions frequently decreased performance by encouraging flatter, bushier structures. The resulting decrease in levels of redundancy that can be exploited through the assignment of contingencies leads to higher chances of actor overload.

Managing contingencies can offset this increase, as shown by the best performing organization using contingency management, communication management, and incoming interaction management. In this case, the increased number of incoming links allow a more capable actors to make decisions, yielding higher rewards. Speed is also increased both directly by communication management, and indirectly because of the bushier structure. This sort of interplay between capabilities is important to take into account when evaluating and designing SPAs.

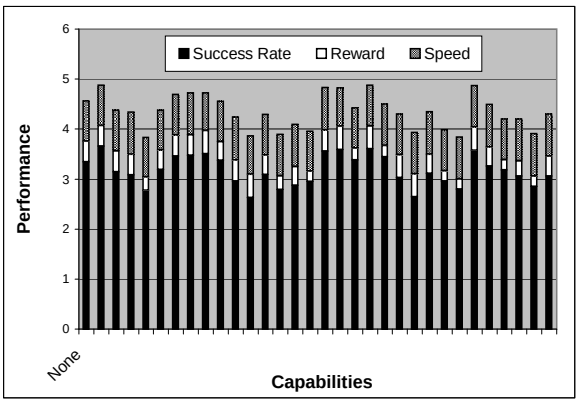


Figure 5: Performance breakdown of organization with scale free structure for all combinations of SPA capabilities.

6. CONCLUSIONS

We have taken an initial step toward identifying the ways in which intelligent personal assistants will impact the performance of human organizations. We have hypothesized a number of possible capabilities of SPAs, evaluated their effect on performance, and shown that the structure of the underlying organization has a substantial effect on the effectiveness of the SPA in improving performance. Our results indicate that SPAs can substantially improve organizational performance by flexibly managing pre-planned contingencies in order to reduce actor overload. However, the effects of the SPAs are far from dramatic and do not impact all structures evenly. Significantly, more work will be required to understand precisely when SPAs are useful. More importantly more analysis should be performed to determine what SPA capabilities might make a dramatic impact.

7. REFERENCES

[1] Manju Ahuja and Kathleen M. Carley. Network structure in virtual organizations. *J. Computer-Mediated Communication*, 3(4), 1998.

[2] A.-L. Barabasi, R. Albert, and H. Jeong. Mean-field theory for scale-free random networks. *Physica A*, 272:173–187, 1999.

[3] A. W. Beggs. Queues and hierarchies. *Review of Economic Studies*, 68:297–222, 2001.

[4] J.M. Bradshaw, M. Sierhuis, Y. Gawdiak, R. Jeffers, N. Suri, and M. Greaves. Adjustable autonomy and teamwork for the personal satellite assistant. In *IJCAI-01 Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents*, 2001.

[5] K. Carley. Computational and mathematical organization theory: perspective and directions. *Comput. Math. Organ. Theory*, 1(1):39–56, 1995.

[6] K. Carley. Computational organization science: A new frontier. *Proceedings of the National Academy of Sciences*, 99:7257–7262, 2002.

[7] H. Chalupsky, Y. Gil, C. Knoblock, K. Lerman, J. Oh, D. Pynadath, T. Russ, and M. Tambe. Electric Elves: Applying agent technology to support human organizations. In *International Conference on Innovative Applications of AI*, pages 51–58, 2001.

[8] Mehdi Dastani, M. Birna van Riemsdijk, and John-Jules Ch. Meyer. *Multi-agent Programming: Languages, Platforms and Applications*, chapter Programming Multi-Agent Systems in 3APL. Springer Science+Business Media, Inc., 2005.

[9] Virginia Dignum, Javier Vázquez-Salceda, and Frank Dignum. Omni: Introducing social structure, norms and ontologies into agent organizations. In *PROMAS*, pages 181–198, 2004.

[10] Luis Garicano. Hierarchies and the organization of knowledge in production. *Journal of Political Economy*, 108:874–904, 2000.

[11] L. Garrido and K. Sycara. Multi-agent meeting scheduling: Preliminary experimental results. In *ICMAS-96*, 1996.

[12] John Geanakoplos and Paul Milgrom. A theory of hierarchies based on limited managerial attention. *Journal of the Japanese and International Economies*, 5:202–225, 1991.

[13] B. Horling and V. Lesser. Quantitative organizational models for large-scale agent teams. In *Proceedings of the International Workshop on Massively Multi-Agent Systems*, pages 297–312, December 2004.

[14] C. Li, J. Giampapa, and K. Sycara. Bilateral contract negotiation decisions with uncertain dynamic outside options. *IEEE Systems, Man and Cybernetics, Part C.: Special Issue on Game Theoretic Analysis and Stochastic Simulation of Negotiation Agents*, 36(1), 2006.

[15] Pattie Maes. Modeling adaptive autonomous agents. *Artificial Life Journal*, 1(1 & 2):135–162, 1994.

[16] Thomas W. Malone and Kevin Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119, 1994.

[17] Thomas W. Malone, Joanne Yates, and Robert I. Benjamin. Electronic markets and electronic hierarchies. *Communications of the ACM*, 30(6):484–497, 1987.

[18] Tom Mitchell, Rich Caruana, Dayne Freitag, John McDermott, and David Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):81–91, July 1994.

[19] Pragnesh Jay Modi, Manuela Veloso, Stephen F. Smith, and Jean Oh. CMRadar: A personal assistant agent for calendar management. In *LNCS*, volume 3508, pages 169 – 181. Jan 2005.

[20] N. Schurr, J. Marecki, M. Tambe, P. Scerri, J.P. Levis, and N. Kasinadhuni. The future of disaster response: Humans working with multiagent teams using DEFACTO. In *AAAI Spring Symposium on Homeland Security*, 2005.

[21] T. Shintani, I. Takayuki, and K. Sycara. Multiple negotiations among agents for a distributed meeting scheduler. In *ICMAS-00*, 2000.

[22] K. Sycara and M. Lewis. *Team Cognition*, chapter Integrating Agents into Human Teams, pages 203 – 233. Erlbaum Publishers, 2004.

[23] T. Wagner, J. Phelps, V. Guralnik, and Ryan VanRiper. COORDINATORS: Coordination managers for first responders. In *AAMAS'04*, 2004.

Robust Recognition of Physical Team Behaviors Using Spatio-Temporal Models

Gita Sukthankar
Robotics Institute
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA
gitars@cs.cmu.edu

Katia Sycara
Robotics Institute
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA
katia+@cs.cmu.edu

ABSTRACT

This paper presents a framework for robustly recognizing physical team behaviors by exploiting spatio-temporal patterns. Agent team behaviors in athletic and military domains typically exhibit an observable structure characterized by the relative positions of teammates and external landmarks, such as a team of soldiers ambushing an opponent or a soccer player moving to receive a pass. We demonstrate how complex team relationships that are not easily expressed by region-based heuristics can be modeled from data and domain knowledge in a way that is robust to noise and spatial variation. To represent team behaviors in our domain of MOUT (Military Operations in Urban Terrain) planning, we employ two classes of spatial models: 1) team templates that encode static relationships between team members and external landmarks; and 2) spatially-invariant Hidden Markov Models (HMMs) to represent evolving agent team configurations over time. These two classes of models can be combined to improve recognition accuracy, particularly for behaviors that appear similar in static snapshots. We evaluate our modeling techniques on large urban maps and position traces of two-person human teams performing MOUT behaviors in a customized version of Unreal Tournament (a commercially available first-person shooter game).

Categories and Subject Descriptors

I.5 [**Pattern Recognition**]: Miscellaneous; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Algorithms

Keywords

multi-agent plan recognition, RANSAC, HMMs

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

In certain domains tasks are too complicated to be performed by individual agents and must be achieved through the coordinated efforts of a group of agents over a period of time. To analyze performance of these tasks, we need to extend existing behavior recognition formalisms to accommodate group behaviors. Due to the increase in number of actions generated, assuming that each agent is simultaneously executing actions, team behaviors have a more complicated *temporal* structure than single agent behaviors. However team plans involving physical movement also possess a distinctive *spatial* structure, characterized by the relative positions of teammates and external landmarks, that can be exploited to classify team behaviors. This paper presents a framework for constructing spatio-temporal models to robustly recognize physical team behaviors from position traces of the agents' movement.

We demonstrate our method in the domain of MOUT (Military Operations in Urban Terrain) team planning to recognize military team behaviors customarily performed by human soldiers while moving through an urban environment. Examples of team behaviors include building entry, perimeter guarding, opponent flanking, and formation movement, such as stacked and bounding overwatch.[1] Although this paper focuses on MOUT team behaviors, our algorithms are also applicable for recognizing team behaviors in other military, robotic, and athletic domains.

In this paper, we present two classes of spatial models and illustrate how they can be used to robustly recognize team behaviors in the presence of spatial variations, noise, clutter, and human variability in behavior execution.

team templates: models encoding static spatial relationships between team members and external landmarks. These models are composed of the relative positions of map entities and can be generalized to different geographic layouts using similarity transforms. To efficiently identify team templates over large map areas without using exhaustive search, we employ an efficient randomized search technique, RANSAC (Random Sampling and Consensus) [4].

spatially-invariant HMMs: a set of Hidden Markov Model classifiers applied over short overlapping time windows to identify temporal patterns in agent team configura-

[1]See Section 3 for more details on MOUT team behaviors

tions. By representing the agents' positions in a canonical reference frame, the classifier is robust to certain spatial variations.

We evaluate our modeling and recognition techniques on two different types of data: 1) large urban simulation battle maps and 2) position traces of two-person human teams performing designated sequences of MOUT behaviors in a customized version of Unreal Tournament (a commercially available first-person shooter game).

2. RELATED WORK

Previous work on team behavior recognition has been primarily evaluated within athletic domains, including American football [6], basketball [7], and Robocup soccer simulations [14, 15]. Recognition for military air-combat scenarios has been examined in the context of event tracking [18] and teammate monitoring [8], Finally, a general framework for multi-agent plan recognition (Hierarchical Multiagent Markov Processes) [16] has been demonstrated for a single pair of humans moving around a laboratory.

The research described in this paper focuses on the pattern classification aspect of team behavior recognition—how to systematically model and identify complex spatio-temporal patterns that are created by agent team behaviors. Most previous research in the area of team behavior recognition either leverages the existence of domain-specific features to classify spatial patterns [6, 7] or relies on identifying temporal patterns in behavior sequences [18, 8, 16], largely ignoring implicit spatial information. For many athletic behaviors, researchers have been able to exploit simple region-based or distance-based heuristics to build accurate, but domain-specific classifiers. Based on the premise that all behaviors always occur on the same playing field with a known number of entities, it is often possible to discretize the playing field into grids [15] or typed regions (goal, scrimmage line) [6] that can be used to classify player actions. Prior work on spatial representations for the MOUT domain [2, 11] only addresses the problem of behavior generation, not recognition.

3. MOUT DOMAIN

MOUT (Military Operations in Urban Terrain) scenarios involve moving teams of soldiers and vehicles through heavily-cluttered urban areas to accomplish high-level strategic objectives. A common task performed by human soldiers is building clearing, during which a firing-team of soldiers must enter a building and clear it of enemy occupants and hazards; the challenge is to explore unknown areas while constantly remaining in a defensible position against enemy attack. To achieve this, soldiers move through the urban terrain using set procedures that maintain a defensive position while still propelling the team forward.

The cognitive task analysis of building clearing given in [12] emphasizes the importance of spatial cues in the soldiers' decision making process. Features such as proximity to other buildings, opportunities for cover, open spaces, windows in the building, street layout, fortifications, height of buildings, locations of stairways, obstacles, potential booby-traps, and doors, are critical cues that expert human decision-makers must take into account while performing the task. Certain team behaviors are triggered by the spatial characteristics



Figure 1: MOUT scenario in customized Unreal Tournament environment from spectator viewpoint. A pair of human players control soldiers A and B as they execute the procedure for traversing a T-shaped intersection. The bot models and animations were modified to conform to the appearance of real human soldiers rather than the larger-than-life UT fantasy fighter models.

of the terrain; for instance, firing-teams have special procedures for moving through L-shaped and T-shaped street intersections. In these cases, we hypothesize that static spatial configurations, such as the ones described in our team template models, are highly predictive of the teams' actions.

However, there are some cases, particularly for smaller two-soldier subteams, in which static spatial configurations, by themselves, lack predictive power. The second part of the paper focuses on the three such behaviors, stacked movement, bounding overwatch, and buttonhook entry commonly used during the building clearing task. These behaviors are difficult to identify solely on the basis of static snapshots due to their spatial similarities (see Figure 3).

During stacked movement the purpose is to move the team in such a way that their gun angles completely span all possible areas of approach; the team moves slowly and in synchrony. For moving through open areas or intersections, this approach is less feasible since it's difficult to cover all possible threatened areas. In this case, the bounding overwatch behavior is used; one soldier moves forward while the other remains stationary. The buttonhook entry is similar to bounding overwatch; one soldier moves through the doorway hugging the wall while the other soldier waits and guards. After the entry is clear, the second soldier moves through the doorway hugging the opposite wall. To recognize these behaviors, our classifier needs to exploit the temporal information in behavior sequences in conjunction with spatial information on the position and velocities of team members in a way that is robust to geographic variations that modify the execution of the behavior.

4. METHODS

In this section, we present our methodology for developing spatial models and using them to classify instances of team behavior. Section 4.1 discusses the representation and

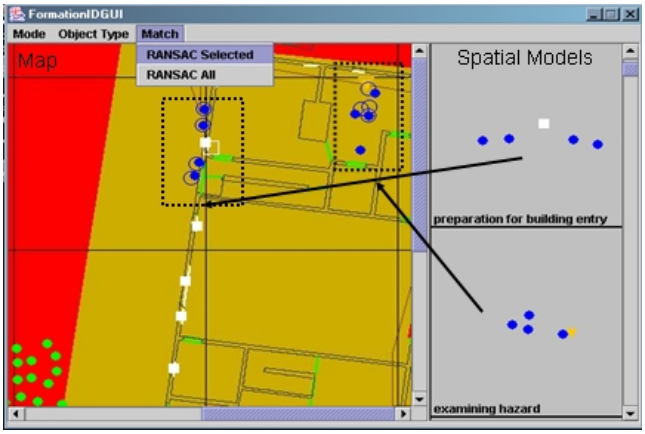


Figure 2: Spatial model authoring and matching system. The library of previously created spatial models are shown to the right of the screen; the left side of the GUI displays an annotated map to be analyzed. A fire team of soldiers (blue circles) examining a hazard (orange inverted triangle) are displayed on the map; a second fire team prepares to enter the building through the door (marked by the white rectangle). Our matching technique successfully associates both groupings of soldiers with the correct spatial models; hollow circles and rectangles show the locations of the entities as predicted by model projection.

authoring of team templates to encode static spatial relationships between team members and external landmarks. Section 4.2 provides an overview of the randomized search technique, RANSAC (Random Sampling and Consensus) that we use to efficiently and robustly identify these templates over large simulation map areas without resorting to exhaustive search. In Section 4.4 we describe the Unreal Tournament simulation environment that we use to collect data from pairs of human subjects performing the three types of statically-similar team behaviors (bounding overwatch, buttonhook entry, stacked movement). Finally, Section 4.5 presents our procedure for converting agent position traces into the canonical representation used by our Hidden Markov Model classifiers (Section 4.6).

4.1 Team Template Representation

To model team behaviors, we developed a tool that enables the author to describe behaviors by designating a set of characteristic spatial relationships that commonly occur during the execution of a behavior. Once a library of spatial models has been constructed, they can be used to classify formations of MOUT entities on a 2D annotated urban map. If labeled training data of team behaviors exists, these team templates could potentially be learned from data using a supervised classifier; however all the experiments described in this paper were created with the authoring interface shown in Figure 2.

Each team template contains the following attributes:

behavior name: Behaviors are represented by collections of spatial models; however no particular temporal structure, or execution order, is attached to the collection. Each spatial model can only belong to a single behavior; we do not include models which appear in a large set of behaviors since they are unlikely to help in discriminating between multiple behaviors.

spatial position of relevant entities: Entities are represented by a single (x,y) coordinate of their centroid; larger entities are represented as groups of points connected by visibility constraints.

entity type: For our library of MOUT behaviors, we designated eleven types of entities including person (unknown), civilian, teammate, opponent, hard cover, soft cover, empty area, windows, intersections, doorways, hazards, and objectives.

One consideration in developing models is generalization—how well do models developed for one scenario match behaviors executed in a different spatial layout? Without generalization it becomes impractical to exhaustively enumerate all possible spatial relationships that can occur across different maps. To solve this problem, we define a set of legal transforms to project models to new spatial layouts and score the quality of the match. We define the set of legal transforms to be the class of similarity transforms (rotation, translation, and scaling); these can be parameterized in homogeneous coordinates as follows:

$$\mathbf{T} = \begin{bmatrix} s\cos(\theta) & s\sin(\theta) & x \\ -s\sin(\theta) & s\cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix}$$

where θ is the angle of rotation, s is a scale factor, x is the x-translation, and y is the y-translation. This formulation can easily be extended to model three dimensional transforms by increasing the matrix to 4×4. The next section describes a robust and efficient technique for searching the space of possible transforms.

4.2 Efficient Template Matching

Given a set of spatial models and valid transforms, the problem of determining which spatial models are applicable to the current map can be solved by searching the space of potential transforms and models to find all the combinations of model plus transform that result in a match of sufficient quality. A commonly-used approach is exhaustive template matching [1]. Each template is applied to all possible locations in the map using a sliding window; the distance function is calculated over the window area and those matches that score under the threshold are retained. This process is exhaustively repeated for a range of scales and rotations, over all models in the library. Unfortunately this process is time consuming, scales poorly to higher dimensional transforms, and is sensitive to noise, occlusion and misalignment.

Instead, we employ a statistically-robust technique, Random Sampling and Consensus (RANSAC) [4], to efficiently sample the space of transforms using hypotheses generated from minimal sample sets of point correspondences. The algorithm can be summarized as follows:

hypothesis generation: entities are drawn uniformly and at random from the annotated map and associated with randomly-selected entities of the same type in the model. Two pairs of corresponding entities are sufficient to uniquely specify a transform hypothesis. This data-driven method of generating hypotheses is much more efficient than uniformly sampling the space

of possible transforms or exhaustively searching a discretization of the transform space.

hypothesis testing: Given a transform hypothesis, we project all of the entities in the model to the coordinate frame of the map and assess the quality of the match based on both spatial similarity and type matching. This gives us the likelihood that the given hypothesis could have generated the observed data in the map.

For each spatial model, we use RANSAC to randomly generate and test a large number of plausible transforms and select those hypotheses (a combination of a model and a valid transform) with match quality better than a specified threshold.

Since our spatial transforms have four degrees of freedom, they can be fully specified by two pairs of point correspondences. First, we randomly select two entities from the model under consideration; then based on the types of the entities (e.g., civilians, hard cover, hazard) we randomly select candidate entities on the map with compatible object types. The positions of these entities is used as the minimal set to generate a transform hypothesis as follows.

Given the minimal set $\{(x_1, y_1), (x_2, y_2)\}$ from the model and the corresponding set of points $\{(X_1, Y_1), (X_2, Y_2)\}$ from the map, we generate a third virtual pair of correspondences $(x_3, y_3) \mapsto (X_3, Y_3)$ where

$$\begin{aligned} x_3 &= x_1 + y_2 - y_1 \\ y_3 &= y_1 + x_1 - x_2 \\ X_3 &= X_1 + Y_2 - Y_1 \\ Y_3 &= Y_1 + X_1 - X_2 \end{aligned}$$

From these three correspondences, we can directly recover $\mathbf{T}$ using matrix inversion.

$$\begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} = \begin{bmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}^{-1}$$

This is a solution to a general affine transform [5] given three pairs of point correspondences, however $\mathbf{T}$ is guaranteed to be a valid, orientation-preserving similarity transform due to our construction of the third point.

To score a hypotheses, we transform the location of every entity in the model to the map using the transform $\mathbf{T}$. Each model entity contributes a positive vote for the given hypothesis if the distance from its predicted location to the closest map entity of compatible type falls below a specified threshold. The quality of a hypothesis is defined as the normalized sum of these individual votes.

Since RANSAC stochastically searches the space of possible transforms it is not guaranteed to find the best match. However the following formula can be used to determine how many iterations are necessary to achieve the best match with a specified probability of success [19]:

$$m = \left\lceil \frac{\log(1-P)}{\log[1-(1-\epsilon)^s]} \right\rceil$$

P is the target probability (e.g., $P = 0.99$ means the best match is found 99% of the time). s is the number of elements required to define the minimal set ($s = 2$ since a similarity transform requires 2 pairs of point correspondences). ϵ is the expected fraction of outliers in the data set. In traditional RANSAC applications ϵ is typically only about 0.1 (10% of the points are expected to be invalid). For our application, the fraction of outliers refers to the number of map annotations that do not match a single model; since each map actually contains multiple models in addition to entities that do not match any model the fraction of expected outliers is approximately 0.95. From the formula above, this indicates that the number of RANSAC iterations required to reliably find the best match is 1840. Note that executing the given number of iterations does not guarantee a particular level of classification accuracy; it merely ensures that there is only a 0.01% probability that a similarity transform exists that would achieve a higher score than the one returned by RANSAC. In Section 5.1, we present classification results for identifying team behaviors from 2D annotated maps of simulated urban layouts using our team template model and RANSAC.

4.3 Spatio-temporal Classification

There are some cases, particularly for smaller two-soldier subteams, in which static spatial configurations, by themselves, lack predictive power. To recognize these behaviors, our classifiers need to exploit the *temporal* information in behavior sequences in conjunction with spatial information on the position and velocities of team members. Since team behaviors can be executed in a variety of terrains, the classifiers must be robust to deviations in behavior execution caused by the team's response to local terrain features. However, arbitrarily introducing similarity transforms in the middle of a behavior sequence can destroy the spatio-temporal pattern created by the team's movements. To address the problem, we developed spatially-invariant classifiers, by transforming our position data into a canonical reference frame defined by the team's motion, and applying a set of Hidden Markov Model classifiers to recognize three statically similar team behaviors (bounding overwatch, buttonhook entry, stacked movement).

4.4 Human Data Collection

To evaluate our spatio-temporal classifiers, we collected data from a pair of human players using our modified Unreal Tournament game interface to manipulate "bots" through a small urban layout while performing a particular sequence of team behaviors (see Figure 1). Note that the subjects were not playing Unreal Tournament, but using Unreal Tournament to execute sequences of commonly used MOUT team maneuvers. To directly monitor the performance of human players, we customized Unreal Tournament (UT) using the game development language *Unrealscript*. Many of the original UT game classes were written in *Unrealscript* and thus can be directly subclassed to produce modified versions of the game (known as mods); for example, Gamebots [9] is an example of a mod that allows external programs to control game characters using network sockets.

We developed our own `TrainingBot` mod that allows us to save the state of all the bots in the scenario; currently we save each player's ID number, position $(x, y.z)$, and rotation (θ,ϕ) every 0.15 seconds. This information is useful for both offline behavior analysis and for a separate replay mode that allows us to create bots that follow the paths recorded by the original players.

4.5 Canonical Representation

Due to the continuous nature of the domain, automatically determining the exact transition points between team behaviors is a difficult problem. While approaching and entering buildings, the players continue moving their bots, changing team behaviors as appropriate for the physical layout. We address this issue by dividing the traces into short, overlapping time windows during which we assume that a single behavior is dominant; these windows are classified independently as described in Section 4.6. To recognize team behaviors performed in different physical layouts, it is important for our classifier to be rotationally- and translationally-invariant; we achieve this by transforming the data in each window into a canonical coordinate frame as described below.

More formally, we define:

- $a \in 1, \ldots, A$ is an index over A agents;
- j is an index over W overlapping windows;
- $t \in 1, \ldots, T$ is an index over the T frames in a given window;
- $\mathbf{x}_{a,j,t}$ is the vector containing the (x, y) position of agent a at frame t in window j.

The centroid of the positions of the agents in any given frame can be calculated as:

$$\mathbf{C}_{j,t} = \frac{1}{A} \sum_{\forall a} \mathbf{x}_{a,j,t}.$$

We describe the configuration of the agent team at any given time relative to this centroid to achieve translation invariance. However, rather than rotating each frame independently we define a shared canonical orientation for all of the frames in a window. This is important because it allows us to distinguish between similar formations moving in different directions (e.g., agents moving line abreast vs. single file). One standard technique for defining a canonical orientation is to use the principal axis of the data points for that window, which can be calculated using principal component analysis (PCA). However for efficiency we have empirically determined that we can achieve similar results by defining the canonical orientation as the displacement of the team centroid over the window: $\mathbf{d}_j = \mathbf{C}_{j,T} - \mathbf{C}_{j,1}$.

We rotate all of the data in each window so as to align its canonical orientation with the x-axis, using the rotation matrix $\mathbf{R}_j$. Thus the canonical coordinates, $\mathbf{x}'$, can be calculated as follows: $\mathbf{x}'_{a,j,t} \equiv \mathbf{R}_j \mathbf{x}_{a,j,t} - \mathbf{c}_{j,t}$. Our current recognition technique (described in Section 4.6) also relies on observations of agents' velocity as a feature which we locally compute as: $\mathbf{v}_{a,j,t} \equiv ||x'_{a,j,t+1} - x'_{a,j,t}||$.

4.6 HMM Classification

For each canonically-transformed window in our trace, our goal is to select the best behavior model. We perform this classification task by developing a set of hidden Markov models (HMMs), one for each behavior b, and selecting the model with the highest log-likelihood of generating the observed data. Our models ($\{\lambda_b\}$) are parameterized by the following:

- N, the number of hidden states for the behavior;
- $\mathbf{A} = \{a_{ij}\}$, the matrix of state transition probabilities, where $a_{ij} = Pr(q_{t+1} = j | q_t = i), \forall i, j$ and q_t denotes the state at frame t;
- $\mathbf{B} = \{b_i(o_t)\}$, where $b_i(o_t) = \mathcal{N}(\mu_i, \Sigma_i)$. The observation space is continuous and approximated by a single multivariate Gaussian distribution with mean, μ_i and a covariance matrix, Σ_i, for each state i;
- $\pi = \{\pi_i\}$, the initial state distribution.

For our problem, given A agents in a team, the observations at time t and window w are the tuple:

$$o_t = (\mathbf{x}'_{1,w,t}, v_{1,w,t}, \ldots, \mathbf{x}'_{A,w,t}, v_{A,w,t}).$$

We determine the structure for each behavior HMM based on our domain knowledge. For instance, the stacked behavior can be described using only two states ($N = 2$), whereas we represent the more complicated bounding overwatch behavior using six states connected in a ring. Each hidden state captures an idealized snapshot of the team formation at some point in time, where the observation tuple (in canonical coordinates) is well modeled by a single Gaussian. Rather than initializing the HMMs with random parameters, we use reasonable starting values. These can be polished using expectation-maximization (EM) [3] on labeled training data.

To determine the probability, $Pr(o_{1\ldots T}|\lambda_b)$, of generating the observed data with the model λ_b, we employ the forward algorithm [13] as implemented in the Hidden Markov Model toolbox [10]. We classify each window segment with the label of the model that generated the highest log-likelihood.

5. RESULTS

We evaluate our methods using two sets of experiments: 1) formation recognition in simulated 2-D overhead maps of urban areas annotated with the location of MOUT entities (see Section 4.1). 2) behavior identification from activity traces of two-person human teams performing sequences of MOUT behaviors (see Section 4.4). The former assesses the accuracy of the RANSAC-based method while the latter examines the performance of spatially-invariant HMMs. The two methods are complementary, and Section 6 discusses strategies for combining them to further improve recognition accuracy.

5.1 Team Template Matching

To test the robustness of our team template matching approach, we add clutter to the maps and distort formations by perturbing the positions of MOUT entities. Figure 5.1 reports the precision (fraction of correctly-classified results) and recall (fraction of formations that were detected) of our classifier under different conditions of clutter and location perturbation. Note that our approach independently matches each template against the data; thus, all matches that exceed the threshold score are reported as detections. The precision/recall curves are generated by varying this threshold parameter. There is no intrinsic restriction against assigning the same map entity to different templates — this enables us to create templates corresponding to a team and its component sub-teams, and to simultaneously recognize both.

In each experiment, we randomly place fifty MOUT formations on the urban map and report the precision and

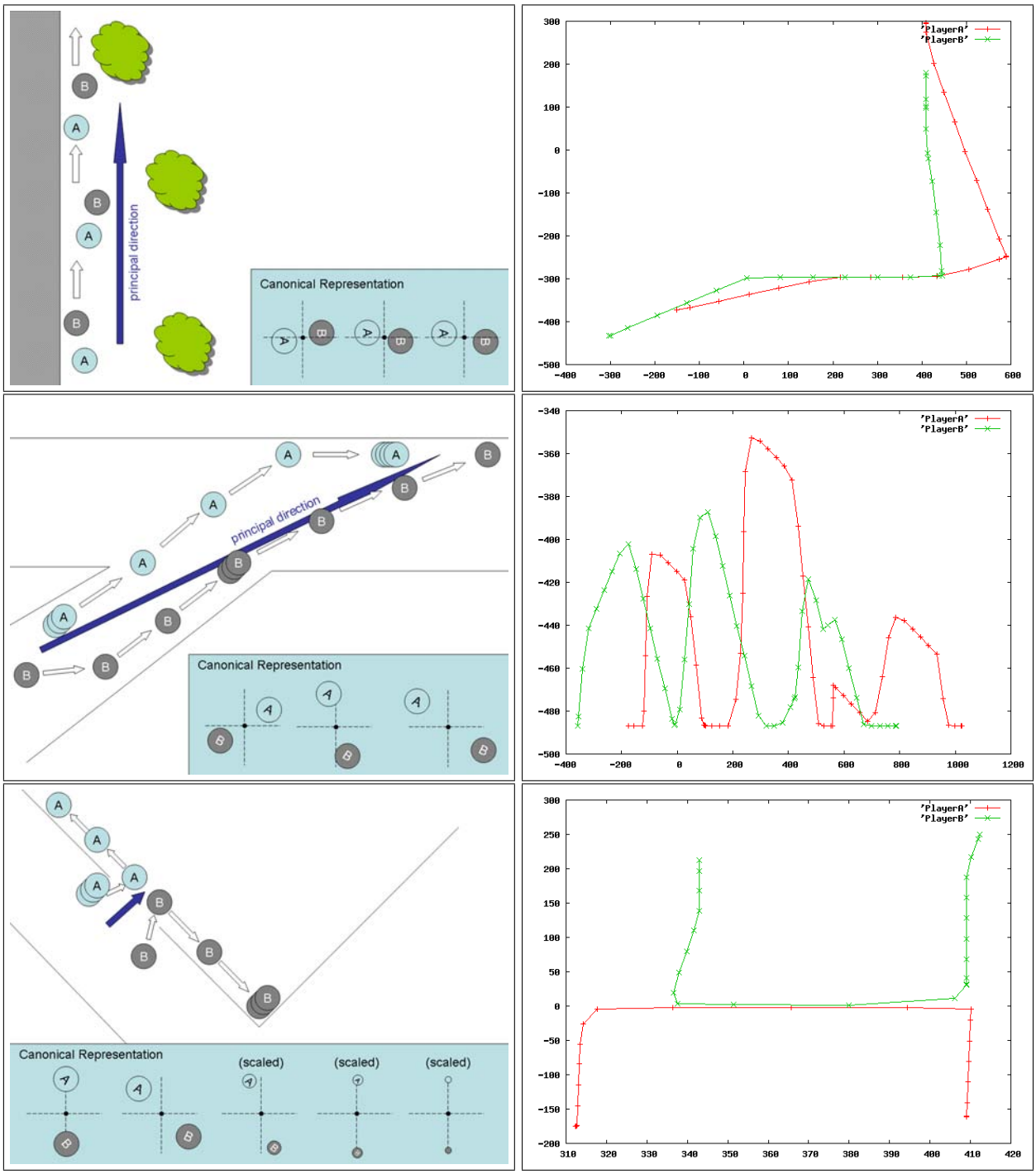


Figure 3: Team Behaviors: Stacked Formation (top), Bounding Overwatch (middle), Buttonhook Entry (bottom). Schematics for each behavior, along with the canonical representation for several frames, are depicted in the left column. A sample raw trace for each behavior is shown in the right column; the coordinates of the axes are in Unreal Tournament length units.

recall averaged over ten RANSAC searches. The left panel of Figure 5.1 shows the effects of adding clutter (spurious MOUT entities of the appropriate type) to the map, without increasing the number of RANSAC iterations. The percentage of clutter is measured against the total number of MOUT entities in the formations on the map (thus 100% clutter denotes a 1:1 ratio between spurious and desired MOUT entities). The results show that, as expected, the RANSAC-based approach is very resistant to the presence of spurious entities on the map, and that precision/recall are both very high even at the extreme clutter levels. The right panel of Figure 5.1 shows precision/recall results for experiments where the locations of each of the MOUT entities were perturbed with iid Gaussian noise. As expected, the performance degrades as noise is added since the spatial configuration of the formation ceases to resemble the formation represented by the idealized model. However, we note that the technique is successful at identifying an acceptable number (80%) of the formations under reasonable noise conditions.

5.2 Spatially-Invariant HMMs

To evaluate our spatio-temporal classification method, we developed HMM models for three behaviors typically employed by two-person firing teams during the building clearing task: stacked movement, bounding overwatch, and buttonhook entry (see Figure 3). Note that these three behaviors look very similar in static snapshots and can only be robustly recognized by observing spatio-temporal traces. Position data was simultaneously recorded from two subjects at 0.15 second intervals using our `TrainingBot` mod (see Section 4.4). Players executed team behaviors in predesignated sequences, transitioning smoothly from one behavior into the next, adapting each behavior as needed to the local physical layout (turning corridors, entering rooms). The traces were divided into overlapping 20 frame (3 second) windows, which were transformed into a canonical coordinate frame as described in Section 4.5 (illustrated in the inset of Figure 3). The window size was empirically selected base on the observed average speed of the MOUT soldiers.

By using real data collected from human players rather than simulated traces, we can evaluate the robustness of our approach to realistic deviations during behavior execution. Figure 3 (right) shows a raw trace for each behavior; note that consecutive executions of the same behavior exhibit significant variation, particularly noticeable in the bounding overwatch behavior. Each behavior was also performed in a variety of local physical layouts. Table 1 presents the classification results (confusion matrix) for the three modeled behaviors; the accuracy of the HMM approach is good, particularly for the stacked formation. Buttonhook entry is sometimes confused with bounding overwatch, as may be expected from similarities in the canonical representation shown in Figure 3.

Table 1: Confusion matrix for HMM behavior classification. The ground truth is given in the left column; the classification result is given in the top row. The spatially-invariant Hidden Markov Model approach achieves good accuracy. Buttonhook entry is often confused with bounding overwatch, as may be expected from similarities in the canonical representation as shown in Figure 3.

	stacked	bounding	buttonhook
stacked	90%	10%	0
bounding	14%	67%	19%
buttonhook	0%	33%	67%

6. DISCUSSION

The two approaches presented in this paper, team templates and spatially-invariant HMMs, are complementary. Team templates are easy to author and can be rapidly executed over large, cluttered maps. Spatially-invariant HMMs are more difficult to design, but are well-suited for classifying statically-indistinguishable behaviors. We believe that combining outputs from both models should lead to improved recognition accuracy. Team templates are effective at determining the spatial context of an observed behavior, enabling improved discrimination between behaviors that display a similar temporal structure. For instance, although a buttonhook entry and a bounding overwatch often appear similar to the spatially-invariant HMM, the presence of context can enable us to disambiguate them since the former behavior is typically performed to enter a door or window, while the latter is used to move the formation down a corridor or street.

We are interested in applying our algorithms to the recognition of large team formations [17]. The efficient team template matching strategy described in the first half of the paper scales well to larger team sizes. Introducing more points into the team template is actually beneficial for two reasons: 1) the minimal set required to calculate the transform is independent of team size; 2) including additional model entities improves the robustness of the matching since a greater number of entities can contribute votes in support of the best hypothesis, and it is less probable that a configuration of spurious entities will be a good match.

In principle, the spatially-invariant HMM approach can also be applied to larger teams; however constructing models for teams with more members can become difficult since having more team members creates many more possible configurations (even in our canonical representation). For our particular domain, separating larger teams (of 8 or 16) into two-person subteams which can be be analyzed separately is a feasible approach. We are investigating strategies for combining multiple classifiers to recognize team behaviors; using repeated executions of static template matching followed by the use of more specialized classifiers designed for detecting particular behaviors. Larger team behaviors would be detected by combining the outputs of specialized classifiers (e.g., detecting invariants) rather than using a single highly-complex classifier.

7. CONCLUSION AND FUTURE WORK

This paper presents two methods for recognizing physical team behaviors from spatio-temporal movement patterns. We demonstrate that our techniques are robust to the effects of local spatial variations, clutter, noise, and human variability during behavior execution. In future work, we are particularly interested in the following problems: 1) incorporating the outputs of our classifiers into a framework for symbolic team plan recognition; 2) evaluating our mod-

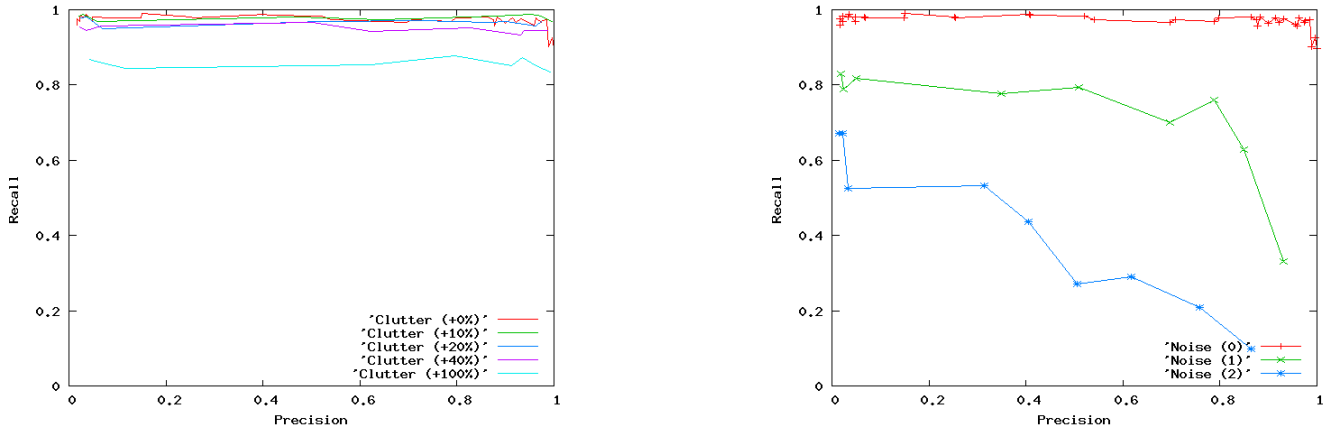


Figure 4: Precision and recall curves for matching team templates using the RANSAC-based method. Fifty formations were placed on an urban map using randomized similarity transforms. Each run employed 100,000 iterations and the precision/recall averaged over 10 trials is shown. The left panel shows the effect of adding spurious MOUT entities (clutter) while maintaining the same number of iterations. The right panel shows the results of distorting observed formations by perturbing the location of each MOUT entity on the map with iid Gaussian noise.

els on MOUT data from human domain experts; and 3) evaluating alternate temporal classifier formalisms.

8. ACKNOWLEDGEMENTS

The authors thank Rahul Sukthankar for proof-reading the document and Reid van Lehn for his work on Unreal Tournament software development. This work has been supported by AFOSR grant F49620-01-1-0542.

9. REFERENCES

[1] D. Ballard and C. Brown. *Computer Vision*. Prentice-Hall, 1982.

[2] B. Best and C. Lebiere. Spatial plans, communication, and teamwork in synthetic MOUT agents. In *Proceedings of Behavior Representation in Modeling and Simulation Conference (BRIMS)*, 2003.

[3] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley & Sons, Inc, 2001.

[4] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, 24(6), 1981.

[5] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[6] S. Intille and A. Bobick. A framework for recognizing multi-agent action from visual evidence. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1999.

[7] M. Jug, J. Pers, B. Dezman, and S. Kovacic. Trajectory based assessment of coordinated human activity. In *Proceedings of the International Conference on Computer Vision Systems (ICVS)*, 2003.

[8] G. Kaminka and M. Tambe. Robust agent teams via socially attentive monitoring. *Journal of Artificial Intelligence Research*, 12:pp.105–147, 2000.

[9] G. Kaminka, M. Veloso, S. Schaffer, C. Sollitto, R. Adobbati, A. Marshall, A. Scholer, and S. Tejada. Gamebots: A flexible test bed for multiagent team research. *Communications of the ACM*, 45(1), 2002.

[10] K. Murphy. The Bayes Net Toolbox for Matlab. *Computing Science and Statistics*, 33, 2001.

[11] D. Pearson and J. Laird. Redux: Example-drive diagrammatic tools for rapid knowledge acquisition. In *Proceedings of Behavior Representation in Modeling and Simulation Conference (BRIMS)*, 2004.

[12] J. Phillips, M. McCloskey, P. McDermott, S. Wiggins, and D. Battaglia. Decision-centered MOUT training for small unit leaders. Technical Report 1776, U.S. Army Research Institute for Behavioral and Social Sciences, 2001.

[13] L. Rabiner. A tutorial on Hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77, 1989.

[14] P. Riley and M. Veloso. On behavior classification in adversarial environments. In L. Parker, G. Bekey, and J. Barhen, editors, *Distributed Autonomous Robotic Systems 4*. Springer-Verlag, 2000.

[15] P. Riley and M. Veloso. Recognizing probabilistic opponent movement models. In A. Birk, S. Coradeschi, and S. Tadorokoro, editors, *RoboCup-2001: Robot Soccer World Cup V*. Springer Verlag, 2002.

[16] S. Saria and S. Mahadevan. Probabilistic plan recognition in multiagent systems. In *Proceedings of the International Conference on AI and Planning Systems (ICAPS)*, 2004.

[17] G. Sukthankar. Thesis proposal: Activity recognition for physically-embodied agent teams. Technical Report CMU-RI-05-44, Robotics Institute, Carnegie Mellon, 2005.

[18] M. Tambe. Tracking dynamic team activity. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1996.

[19] G. Zu and Z. Zhang. *Epipolar Geometry in Stereo, Motion, and Object Recognition*. Kluwer, 1996.

How Autonomy Oriented Computing (AOC) Tackles a Computationally Hard Optimization Problem

Xiao-Feng Xie
Department of Computer Science
Hong Kong Baptist University
Kowloon Tong, Hong Kong
xfxie@comp.hkbu.edu.hk

Jiming Liu
Department of Computer Science
Hong Kong Baptist University
Kowloon Tong, Hong Kong
jiming@comp.hkbu.edu.hk

ABSTRACT
The hard computational problems, such as the traveling salesman problem (TSP), are relevant to many tasks of practical interest, which normally can be well formalized but are difficult to solve. This paper presents an extended multiagent optimization system, called $MAOS_E$, for supporting cooperative problem solving on a virtual landscape and achieving high-quality solution(s) by the self-organization of autonomous entities. The realization of an optimization algorithm then can be described in three parts: a) encode the representation of the problem, which provides the virtual landscape and possible auxiliary knowledge; b) construct the memory elements at the initialization stage; and c) design the *generate-and-test* behavior guided by the law of *socially-biased individual learning*, through tailoring to the domain structure. The implementation is demonstrated on the TSP in details. The extensive experimental results on real-world instances in TSPLIB show its efficiency as comparing to other algorithms.

Categories and Subject Descriptors
I.2.8 [**Problem Solving, Control Methods, and Search**]: heuristic methods; I.2.11 [**Distributed Artificial Intelligence**]: multiagent systems

General Terms
Algorithms, Experimentation, Theory

Keywords
Multiagent system, Autonomy Oriented Computing (AOC), Traveling Salesman Problem (TSP), global optimization, search, cooperative problem solving, emergent and collective behavior.

1. INTRODUCTION
The hard computational problems are ubiquitous in scientific and engineering fields [26], which often are conceptually simple and can be cast as a search through a space of alternatives [24][37].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8–12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

The landscape paradigm originated in theoretical biology is used for search in general [25]. Suppose the *representation space* (S_R) contains all the potential solutions, where each is called a *state* $\vec{s}$, and the solution space (S_O) is a set of states with reasonable quality. Normally, the $|S_O|/|S_R|$ is quite small since that $|S_R|$, i.e. the cardinality of S_R, expands exponentially with the size of problems [24]. For the task to find $\vec{s} \in S_O$ with high probability, typical challenges include: a) little *a priori* knowledge is available; and b) total computational resource and time are bounded.

The traveling salesman problem (TSP) [40] is a classic NP-hard problem. Although the TSP is easily formulated, it exhibits various aspects of hard computational problems and has often served as a touchstone for new problem solving methods [29][54]. The researches [10][17][53] have indicated the phase transitions under certain order parameter(s), however, it seems most real-world instances are still located at the hard computational region.

Autonomy oriented computing (AOC)[31] addresses the modeling of autonomy in the entities of a complex system and the self-organization of them in achieving a specific goal. The complex behaviors can emerge from the interactions of autonomous entities following simple rules. Recently, the AOC based methods have been applied to solve various problems [30][47][51].

In this paper, the compact multiagent optimization system ($MAOS_C$) [51], which is originally proposed for handling with the numerical optimization problem (NOP) only, is extended for supporting the cooperative search on a virtual landscape by a society of compact agents with limited memory capacity and simple behavioral rules under ecological rationality [18]. Each agent only has moderate problem solving capability by comparing with two extremes: a) the reflex agent, such as *ant* [12], which has no declarative memory and can only produce reflex behaviors in the environment; b) the cognitive architecture, such as ACT-R [1], which is rather sophisticated under unbounded rationality. Besides, the cooperation [13] is necessary because no single agent has sufficient knowledge on the virtual landscape. Here the agents collaborate with the others by indirect interactions, which are implemented through the communication medium role of the environment [31], instead of by sophisticated negotiation [27].

The paper is organized as follows. In Section 2, the extended multiagent optimization system ($MAOS_E$) in two levels is introduced in details. In Section 3, the implementation is demonstrated on the TSP, which addresses exploiting the problem features in search. In Section 4, the extensive experimental results by $MOAS_E$, using real-world instances from the TSPLIB, are

compared with those of some existing algorithms [45][54]. In the last section, this paper is concluded.

2. THE MULTIAGENT SYSTEM

The extended multiagent optimization system is built in two levels: a) the symbolic level for supporting basic problem solving; and b) the multiagent framework for specifying the organization structure and operation mechanism based on the symbolic level.

2.1 The symbolic level

The general problem solving capability arises from the consequence of the interaction of declarative and procedural knowledge [1]. As in the information process system [37], the declarative knowledge is represented in *symbol structures*, called T_INFO elements, and the procedural knowledge is represented in *elementary information processes*, called behavioral rules.

Each T_INFO element is described by a tuple <I_RC, I_TYPE, I_CON>, where I_RC is the retrieve cue, I_TYPE indicates a certain search space, and I_CON is its state.

The memory [19][37] is used for storing T_INFO elements. Each element can be retrieved from a memory according to its I_RC.

The memory must be updated if it is to be useful [19]. But updating is not encoding a new T_INFO element. Instead, only the I_CON is subjected to change. Then a T_INFO element can be referred as a trajectory that comprise of many instances. For specified time t, a T_INFO instance is expressed as $\text{I\_TYPE}_{\text{I\_RC}}^{(t)}$.

Each behavioral rule is described by a tuple <I_NAME, I_KEY>, where I_KEY indicates a virtual interface for handling specific T_INFO elements, and I_NAME indicates the real operations, which can be controlled by specifying setting parameters within a rule-parameter space. A behavioral rule is expressed as $R_{\text{I\_KEY}}^{\boxed{\text{I\_NAME}}}$.

2.2 The multiagent framework

As an AOC-based model [31], the framework consists of a society of N agents where they cooperate in a sharing environment (E) to realize a common intention of finding high-quality solution for an optimization task. Since the optimization task can be well-defined, the sophisticated negotiation [27] among agents is not necessarily. Instead, an agent is communicated with the others by indirect interactions, which are implicit implemented through the communication medium role of the environment (E).

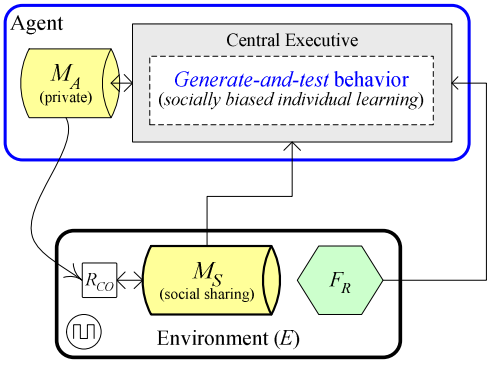


Figure 1. An agent and the environment (E) it roams

Without loss of generality, the agents are homogenous in the sense that they have the same organized structure. Figure 1 shows one of the agents and the environment it roams. All other agents are interacted with the environment in same way.

2.2.1 *Environment*

The environment serves as the domain in which agents roam [23].

Firstly, it contains an internal representation (F_R) [36][41] of the optimization task, which encapsulates accessible rudimental knowledge, includes a virtual landscape [25] and some auxiliary information associated with problem structure.

For the landscape, a quality evaluation rule (R_M) is used for measuring which one has better quality between any two states ($\vec{s}$) in the representation space (S_R). The quality evaluation implies the *intention* to attain the state that is better than another.

For normal problems, there is a cost function $f(\vec{s})$ for each state $\vec{s} \in S_R$. Then the evaluation is realized by a $R_M^{\boxed{O}}(\vec{s}_a, \vec{s}_b)$ rule: for $\forall\ \vec{s}_a, \vec{s}_b \in S_R$, if $f(\vec{s}_a) \le f(\vec{s}_b)$, then the $\vec{s}_a$ is better than the $\vec{s}_b$, and $R_M^{\boxed{O}}$ returns TRUE, else it returns FLASE.

Secondly, the environment holds a socially-shared memory (M_S), called *collective memory* [38], which serves as the blackboard for all agents by creating a shared past. The information flow from the agents to the M_S is supported by the *coordinating* behavior (R_{CO}) as in Fig. 1. In this sense, the environment also serves as an accumulation pool for the emergent collective behavior instead of a pure information provider as in cognitive architectures [1].

Thirdly, the environment keeps a central clock that helps synchronizing the behaviors of the total system, if necessary [31].

In fact, the environment can be seen as a pseudo autonomous entity for supporting the indirect cooperation of agents.

2.2.2 *Agent*

Each agent is a socially situated autonomous entity. Autonomy [31] is an attribute of a self-governed, self-directed entity with respect to its own status, free from the explicit control of another entity. The essence is that each entity is able to make decisions for itself, subject to the limitations of the available information.

The central executive (CE), the most important component of working memory model [3][43] in term of its general impact on cognition, is responsible for manipulating the local behavior and behavioral rules of an agent, which govern how it should act or react to the available information on the representation of problem while determine the next status to which the entity will transit.

The agent has two *belief* sources. Firstly, it possesses a private memory, called M_A, which has limited capacity and can only be modified by agent itself. The M_A is a fundamental mental component for supporting individual learning. Secondly, it can access to the M_S, which is socially available from the environment.

The law of behavior is *socially biased individual learning* (SBIL) as adopted by many species in real world [16], which is a mix of reinforced practice of own experience and socially available information. The SBIL is suggested to be a heuristic in ecological rationality [8][18], which a) gains most of the advantages of both individual and social learning; and b) facilitates the emergent and

collective properties by allowing learned knowledge to be accumulated from one generation to the next.

Due to lack of enough knowledge on the landscape to be searched, the essential behavioral rule is *generate-and-test* rule (R_{GT}). The essential components of an R_{GT} rule include a generating part (R_G), a solution-extracting part (R_S) and a testing part (R_T) [51]. The R_S part has no influence on the problem-solving process. It just exports potential solution(s) during a run. Without loss of generality, it can be supposed that only the R_G part can create new information that contains new potential solution(s), and the R_T part only produces simple reflex behavior, which may determine nontrivial properties for some T_INFO elements in memory.

2.2.3 Working process

The framework runs in two phases. The first phase is *memory initialization* phase, which initializes the T_INFO elements in M_S and in the M_A of all agents according to the F_R. The second phase is *generate-and-test* phase. For a run, the number of cycles is T. The behavior of system in the tth ($t \in [1,T]_{\mathbb{Z}}$) cycle only relies on the status of system in the (t-1)th cycle. By running as a Markov chain process, the system can be analyzed in each cycle.

Each cycle contains two clock steps: the C_PRE and the C_RUN.

At the C_PRE step, the coordinating behavior (R_{CO}) at the environmental level, which summarize the information submitted by all the agents into the M_S, is executed, i.e.

$$\{M_{A(i)}^{(t)} \mid i \in [1,N]_{\mathbb{Z}}\},\ M_S^{(t)} \xrightarrow{R_{CO}} M_S^{(t+1)} \tag{1}$$

For simplicity, here only the observational information is taken into account, i.e. the T_INFO elements in all agents with same I_RC are collected into a T_INFO element in M_S. It is significant since *observational learning* [5] can lead to cumulative evolution of knowledge that no single agent could invent on its own.

At the C_RUN step, if an agent is activated, where its *generate-and-test* rule is executed for generating new information by estimating the distribution of promising space according to available information in M_A and M_S and updating its M_A, i.e.,

$$M_A^{(t)},\ M_S^{(t)} \xrightarrow{R_{GT}} \begin{cases} \{\vec{s}^{(t)}\} \\ M_A^{(t+1)} \end{cases} \tag{2}$$

Naturally, there are two run modes: a) *full-run* mode, where all agents are activated; b) *partial-run* mode, where only N_{ACT} agents ($N_{ACT} \in [1,N]_{\mathbb{Z}}$), which are selected at random, are activated. Under same agent activated times, the system in *partial-run* mode often converges fast since it allows fast information diffusion.

3. IMPLEMENTATION FOR THE TSP

For solving the TSP, the implementation is realized in three steps: a) define the representation of the TSP (F_R); b) initialize the memory; and c) design suitable generate-and-test behavior. According to *no free lunch* theorem [50], the domain knowledge of the problem must be embedded at the implementation stage.

3.1 Representation of the TSP (F_R)

The representation (F_R) provides a basic interface for accessing the essential domain information of the TSP, which includes a TSP landscape and a candidate set for accelerating the local search (LS), the basis of many good heuristic methods.

3.1.1 The TSP landscape

The TSP landscape is quite straightforward. It is a weighted complete graph with V nodes (or called cities) and a cost matrix $D=(d_{ij})$, where d_{ij} represents the length of an edge that connects between cities i an j ($i, j \in [1, V]_{\mathbb{Z}}$). Here we only concern with the symmetric TSP, which have $d_{ij} = d_{ji}$ for every pair of nodes.

The TSP then is to find a minimal-cost Hamiltonian tour that passes through each node once. Each state $\vec{s}$ is a complete tour, which can be represented as a permutation (i_1, i_2, …, i_V) of the integer values from 1 through V, where the cost function is

$$f(\vec{s}) = d_{i_1 i_2} + d_{i_2 i_3} + ... + d_{i_V i_1} \tag{3}$$

The *distance* of two states is defined as the number of edges in which they differ [15][29][34].

In this paper, the original TSP landscape is used in order to allow a fair evaluation of the performance of algorithms. However, it should be mentioned that the transformation of the cost matrix always be an approach for improving the solving performance.

One kind of methods does not change the relative order of the states. As in Lagrangean relaxation [23][40], the length of every tour is increased by $2 \cdot \sum \pi_i$ as we associate with each node i a penalty value π_i and use the transformed cost matrix $D'=(d'_{ij})$, where $d'_{ij} = d_{ij} + \pi_i + \pi_j$. Of course, the order for the same segments of a tour may be changed under different cost matrix.

The other kind of methods indeed changes the global landscape. As in *search space smoothing* [21], the original landscape is transformed into a series of gradually smoother landscapes, which the smoothest one is solved at first and the solutions are used for guiding the following landscapes until the original one.

3.1.2 Candidate set ($\underline{E}_C$) for local search

Local search (LS), also known as neighborhood search, has been shown very effective for hard computational problems, including the TSP. A basic LS algorithm, under a predefined candidate set of neighborhood description, starts from an arbitrary complete tour and repeatedly improves the current tour till it is trapped into a local minimum. 2-Opt, 3-Opt and Lin-Kernighan (LK) algorithm [29] are representative LS methods.

The complete graph for a symmetric TSP has (V^2-V)/2 edges. However, it is intuitively that most of the possible edges will not occur in good tours because they are simply too long [40]. It is therefore reasonable to restrict major attention to promising edges, or called the candidate set, for accelerating local search (LS) methods. The examples of candidate sets include the k ($k \geq 1$) nearest neighbor subgraph, and the Delaunay graph, etc [40]. Of course, it is often expected the candidate set to be smaller so as not to result in a substantial increase in running time.

Helsgaun [23] estimates the chances of a given edge being a member of a good tour by using minimum 1-trees [40]. A 1-tree is a spanning tree with an additional edge combined two nodes with degree 1. The minimum 1-tree is often used for estimating the lower bound on the optimum since it is a relaxation of the TSP. The estimation, or called Held-Karp bound [22], can be improved

substantially on the 1-tree with Lagrangean relaxation [40] by applying *subgradient optimization* [22] for determining a certain transformation of the cost matrix with the suitable set of penalties, which each π_i is associated with each node *i*.

In *α-nearness measure* [23], each edge has a corresponding α-value, which is the minimum length when a 1-tree is required to contain the specified edge based on a transformed cost matrix. Intuitively, the more accurately Held-Karp bound achieved by the transformed cost matrix, the better estimation of α-values for the edges. Then the candidate edges of each node are sorted in ascending order of their *α*-values. Usually the minimum 1-tree shares many edges with an optimal tour. It is intuitive that the smaller *α* is for an edge, the more promising is this edge.

3.1.3 The features of the TSP landscape

The landscape is often studied in various ways. The first way is to understand the landscape in terms of simple parameters [44], in particular, to depict the phase transitions according to certain order parameters [10][17][24][53]. For the TSP, most researches are focused on the matrix *D*. Chessman et al. [10] investigated the standard deviation of the *D* as an order parameter by using a backtrack algorithm. Zhang [53] went deep into the precision of the elements in the *D* by using a branch-and-bound algorithm. The phase transitions in many hard computational problems have revealed interesting regularities across different problems and algorithms [24]. However, most real-world TSP instances are still classified as hard at the existing phase diagrams.

The second way is to understand the landscape based on the hints accumulated from the solving experience of existing algorithms.

Most modern LS variants [2][23][32][45] are essentially under the umbrella of iterated LS (ILS) [32], or called chained LS (CLS) [2], where the basic idea is to modify the current tour by applying a *kick-move* on a previous found tour instead of independently generated one. The intuition behind using chained starting tours is that the strong positive correlation between solution cost and the distance from the closest optimal tour, i.e., better local minima tend to have smaller distance to the closest global optimum by sharing many common partial structures [7][11][54]. The observations have led to the "big valley" hypothesis [7], which argues that the high-quality tours tend to concentrate on a very small subspace around the optimal tour(s) [46].

Moreover, the inherent features of a set of high-quality tours have been exploited in several ways. The first is reduction [29], which locks the edges in the intersection of the edge sets ($\underline{E}_I$) of the tours for speeding up the subsequent search. However, this method is brittle [54], especially as a locked edge is not part of good tour. The second is tour merging [11], which is to look for a high-quality tour in a restrict graph (G_R) consisting of the union of the edge sets ($\underline{E}_U$) of the tours through a "branch-width" algorithm. In fact, the number of edges in a G_R can be quite few due to the large number of shared edges among high-quality solutions. However, optimizing over a G_R can be hard as the original TSP instance. Besides, although a G_R may contain most of edges of an optimal tour, it may not ensure containing at least one optimal tour, then the tour merging may be failed to achieve optimum. The third is backbone guided search [54], which makes using of global information embedded in the pseudo-backbone frequencies for changing the landscape. However, the local search may be trapped into a local minimum unaffected by the changed landscape.

In this paper, the TSP landscape is exploited in the following ways: a) the LS method is applied at the initialization (or early) stage in order to reach the "big valley" as soon as possible [7]; b) The new tour(s) are generated based on the *kick-move* on a previous found tour as in the iterative LS [32]; c) an edge set ($\underline{E}_U$), which is represented in a set of high-quality tours, are used as the guiding information for the kick-move. The basic idea is that, for the edges to be introduced, most of them come from the $\underline{E}_U$ and a few of promising edges are safely introduced by greedy search. Moreover, to use a $\underline{E}_U$ with repeated edges instead of a restrict graph may utilize the pseudo-backbone frequencies implicitly; and d) the diversity among tours is maintained carefully: firstly, each starting tour is conserved in the private memory of each agent; secondly, the partial instead of the full information in $\underline{E}_U$, normally be a single tour, are used for guiding the kick-moves; and thirdly, the diffusion distance for each starting tour is restricted by an additional testing criterion on multiple trails.

3.2 Memory initialization

For simplicity, We use one T_INFO element in M_A, i.e. a tour called $\vec{s}_P^{(t)}$. Hence the element in M_S is $\underline{\vec{s}}_P^{(t)} = \{ \vec{s}_{P(i)}^{(t)} \mid i \in [1, N]_{\mathbb{Z}} \}$.

Following a random initialization, each tour is improved by a LS method, indicated as LS_{MI}. The default choice for LS_{MI} is a 3-Opt with k_{NNI}=20 neighborhood nodes in the candidate set.

3.3 The generate-and-test behavior

For solving the TSP, the generate-and-test behavior can be seen as a *kick-move* on $\vec{s}_a$, the starting tour to be improved, as in ILS [32].

To improve the chances for escaping from the local optimal tours, the double-bridge (DB) [29][32], a non-sequential *kick-move*, is normally used [2]. Helsgaun [23] has extended it into a *2-stage kick-move*, i.e., any infeasible 2- or 3-Opt move (producing an intermediate solution consisting sub-tours) followed by any 2- or 3-Opt move, which produces a valid tour (by merging the sub-tours). The infeasible intermediate solution is significant so as to alter the global shape of a tour [2]. However, the blind mutation should be avoided since it may be ineffective and inefficient.

The *kick-move* guided by a high-quality edge set ($\underline{E}_U$) has been exploited extensively. Many of them [15][28][34][42] can be classified as the *tour-guided 2-stage kick-move* (TG2KM): guided by a tour $\vec{s}_b$ in $\underline{E}_U$, the tour $\vec{s}_a$ is first modified into an infeasible intermediate solution, normally consist disjoint segments or sub-tours, which are then repaired by ingenious greedy mechanisms while considering available information, such as the $\underline{E}_C$.

Many edge-based recombination operators, such as Voronoi Quantized Crossover (VQX) [41], Natural Crossover (NX) [28], Distance Preserving Crossover (DPX) [15], Edge Assembly Crossover (EAX) [34][35], etc., can be taken as the examples.

Guiding by a tour instead of mutating blindly brings an implicit advantage by utilizing the distance between $\vec{s}_a$ and $\vec{s}_b$. Normally, the larger distance, the large perturbation on $\vec{s}_a$ is then allowed. It brings an adaptive control of the balance between exploration and exploitation along with the execution cycles.

Using a single tour instead of the whole information available is simply because the former has inherent mechanism for preventing premature convergence by locally diffusion of information in a

self-organized agent network while the latter needs sophisticated mechanism although it provides more complete information.

Normally, new promising edges can only be safely introduced when repairing the intermediate solution in greedy way. Hence the unexpected edges that are introduced into the intermediate solution, which are not existed in the $\vec{s}_b$ and the set $\underline{E}_C$, should be as less as possible. The following R_{GT} rule, called $R_{GT}^{\boxed{EA}}$, is extended from the EAX [34], because it meets the requirement.

3.3.1 The generating behavior

The generating part (R_G) is applied for generating new tour(s).

The preliminary operation is to build the mappings between the external and internal information sources, i.e., from the $\vec{s}_P^{(t)}$ and the $\$\vec{s}_P^{(t)}$ to the starting tour $\vec{s}_a$ and the guiding edge set $\underline{E}_U$. Here $\vec{s}_a = \vec{s}_P^{(t)}$ so as to preserve information naturally by using the private memory; and $\underline{E}_U = \$\vec{s}_P^{(t)}$ so as to make use of the cumulative pool. Moreover, the $\vec{s}_b$ is a tour selected from the $\$\vec{s}_P^{(t)}$ at random so as to utilize the pseudo-backbone frequencies implicitly.

By defining the G_{AB} as a graph constructed by merging $\vec{s}_a$ and $\vec{s}_b$, the edges on G_{AB} are then divided into AB-Cycles [35], where an AB-cycle is defined as an even-length sub-cycle on G_{AB} generated by tracing different edges of $\vec{s}_a$ and $\vec{s}_b$ alternately. As an AB-Cycle is identified, then the edges making up the AB-Cycle are removed from G_{AB} and the AB-Cycle is stored into the AB-Lib if it only contains the common edges of $\vec{s}_a$ and $\vec{s}_b$. The procedure is repeated till no AB-Cycle could be extracted.

Each AB-cycle in AB-Lib can be used as the guiding information for applying a *kick-move* on $\vec{s}_a$ in two steps [35][49].

The first step is to generate an intermediate solution, i.e., by removing edges of $\vec{s}_a$ in the AB-cycle from $\vec{s}_a$ and adding edges of $\vec{s}_b$ in the AB-cycle to $\vec{s}_a$. The result is a set of disjoint sub-tours. It is obviously that all the changed edges of $\vec{s}_a$ are selected from $\vec{s}_b$. Besides, the edges in $\vec{s}_a$ will be unchanged in the intermediate solution if the edges that connected to a node in $\vec{s}_b$ belong to the edges that connected to the same node in $\vec{s}_a$.

The second step is to modify the intermediate solution into a valid tour by merging its sub-tours. Of course, it is a choice to apply the 2-Opt for merging sub-tours [9]. For reducing the computational cost, a LS version, which can be seen as the first round of 2-Opt, is used: two sub-tours are greedy merged, by deleting a single edge from each sub-tour and adding two edges, where one of them is a node selected at random in the candidate set $\underline{E}_C$, to connect the nodes in both sub-tours with degree 1.

Of course, to find a good tour by simple one trail is not always easy. Fortunately, there already are many AB-cycles in the AB-Lib, which allows us to perform a multi-trail mechanism (MTM), or called iterative child generation (ICG) [35][49], i.e. to generate N_{MTM} candidate tours, where each is based on an AB-Cycle selected in AB-Lib at random. The idea was also exploited by the Fitness-Distance-Based Diversification (FDD) [45] and the brood selection mechanism [48]. The default value for N_{MTM} is set as 30.

3.3.2 The testing behavior

The testing part (R_T) is used for maintaining the M_A, which is similar as updating the starting solution in iterated LS, based on the competition among the generated candidate tours.

The smaller difference between two information sources for the R_G part, the higher probability for falling into one of the local minima associated with the guiding information [7][15]. Hence, it is critically to maintain the diversity of information intentionally since the high-quality solutions are already rather close.

For the competition of generated tours, it may easily get trapped in specific regions of the TSP landscape if we only consider the essential high-quality criterion [49]. The FDD mechanism [45] hence utilizes an additional criterion, i.e., high quality as well as rather distant from the starting tour.

If $\vec{s}_a$ is located at a local valley, then a tour better than $\vec{s}_a$ is located at another valley of the TSP landscape. Here if only such tours are allowing for competition, then it is no long necessarily to keep large distance from the current solution intentionally.

It is expected that the private information in M_A has strong influence on the generated states, so as to keep a local diffusion effect. Hence large adjustments are possible, but are much less probable than small adjustments [4]. For a generated tour $\vec{s}_c$ with better quality than $\vec{s}_a$, the cost function is transformed as follows:

$$f'(\vec{s}) = (f(\vec{s}_c) - f(\vec{s}_a))/(DIS(\vec{s}_a, \vec{s}_c))^{C_D} \qquad (4)$$

where $DIS(\vec{s}_a, \vec{s}_c)$ is a function to reflect the increasing of distance between $\vec{s}_a$ and $\vec{s}_c$, and C_D is called *diffusion coefficient*. As in [34], the $DIS(\vec{s}_a, \vec{s}_c)$ function is defined as half of the number of edges in the corresponding AB-cycle to reduce computational cost for calculating the exact distance.

If C_D=0, $f'(\vec{s})$ represents the high-quality criterion, as in EAX-1AB [34][35] or in brood selection mechanism [48]. If C_D=1, $f'(\vec{s})$ represents the criterion as in EAX-Dis [35]. If C_D is very large, the $\vec{s}_c$ will be selected from the tours with smallest *DIS* value. In fact, the C_D determines the diffusion velocity of information. The default setting is C_D=1.5.

The selected $\vec{s}_c$ is then used for replacing the $\vec{s}_P^{(t)}$ in M_A. Hence $\vec{x}_P$ is a steady-state solution satisfying the above testing criterion.

4. EXPERIMENTAL RESULTS

The experiments are carried out on a set of real-world instances with $1000 \le V < 3000$ nodes in TSPLIB [39].

The results are compared with the iterative LS (ILS-FDD) [45] and the iterated backbone guided LK algorithm (IBGLK) [54].

The ILS-FDD generates a new starting tour for ILS with Fitness-Distance-Based Diversification (FDD) [45]. Their 3-Opt uses some standard speed-up techniques under a candidate list of 40 nearest neighbors. The ILS version has achieved best performance while comparing to three other algorithms, i.e., the genetic algorithm (GA) using DPX-crossover (GA-DPX) [15], the repair-based GA [48], and the MAX-MIN ant system (MMAS) [46].

The IBGLK [54] combines pseudo-backbone utilization with the iterated LK algorithms by considering two issues: the first issue is to construct unbiased samples of local minima for constructing pseudo-backbone; and the second issue is to alternate between BGLK and regular LK once each of them fails. For LK algorithm, length-20 nearest nodes, *don't look bits* [6], and the *two-level tree* [14] tour representation, are used. The IBGLK version has outperformed than three versions of LK algorithms [54], i.e. the *ILK-1-run*, the *ILK-5-run*, and the *ILK-10-run*.

Table 1 gives the setting parameters for the standard case of $MAOS_E$, called #Std, where $\underline{E}_C$ indicate the candidate set. Table 2 gives five other cases by changing one parameter value based on the template of #Std. As same as in [45], 25 trails are run on pr1002, pc1173, d1291, and 10 trails are run on other instances.

Table 1. The parameter setting for the case #Std of $MAOS_E$

Parameter	N	N_{ACT}	$\underline{E}_C$	LS_{MI}	C_D	k_{NNI}	N_{MTM}
Value	300	300	α-measure	3-Opt	1.5	20	30

Table 2. The parameter changes for other cases (based on #Std)

Case	#NA100	#N400	#D1.0	#NN	#2Opt
Changes	N_{ACT}=100	N=400	C_D=1.0	$\underline{E}_C$ =*nearest*	LS_{MI}=2-Opt

Table 3 shows the results for average percentage deviation (Δavg) from the optimum by ILS-FDD [45], IBGLK [54], and the $MAOS_E$ cases. Besides, the average CPU-time (*sec*) for the ILS-FDD and the IBGLK, and the average cycles ($\overline{T}$) for the $MAOS_E$ cases, are listed. We also list the run-time (*sec*) for the case #Std and #400. The result for si1032 is just for reference and is not used in the following discusses since it is not tested by ILS-FDD. At the last row, the average values for all instances are calculated.

For the $MAOS_E$ cases, the calculation time mainly contains two parts: the first is N times of LS executions at the initialization stage; the second is $N_{ACT} \cdot \overline{T}$ times of R_{GT} executions by agents. The less R_{GT} executions is normally preferable, although there are great deviations for the costs between different R_{GT} executions.

Both the ILS-FDD and the IBGLK are coded in "C". The ILS is run on 450MHz P3 processor, and the IBGLK runtime is normalized for a 500MHz Alpha. The $MAOS_E$ is coded in JAVA, and is run on 2.8GHz and 3GHz P4 processors in a public cluster. In fact, to compare the computational cost is not so easy we can only count the run-time instead of CPU-time by JAVA. However, the average run-time of the case #Std is 16.2% of that of the IBGLK. Hence the case #Std can be at least comparable with the IBGLK even by only considering the speed difference of CPUs.

The case #Std can achieve the optimum of thirteen instances in all trails, which is better than both the ILS-FDD and the IBGLK. Besides, here we use *worse than/equal to/better than* (W/E/B) analysis. For #Std vs. the ILS-FDD, the W/E/B is 2/6/12, which means that for the results of #Std, 2 instances was worse than, 6 instances were equal to, and 12 instances were better than that of ILS-FDD. For #Std vs. the IBGLK, the W/E/B is 3/6/11.

The effects of parameters are also concerned. From the number of solved instances, the #Std is slightly better than #D1.0, #NN, and #2Opt. However, as the parameter C_D is increased from 1.0 to 1.5, the average $\overline{T}$ is increased by 14.1% due to the reduced diffusion velocity. Hence for the parameter C_D, the balance between the quality and the cost must be minded. As the α-measure instead of the *nearest* measure for the candidate set is used, the average $\overline{T}$ is reduced by 19.3%, which due to the better estimation of promising edges. Although 3-Opt itself is normally slower than the 2-Opt, the average $\overline{T}$ can be reduced by 42.4%. Hence for the LS_{MI}, the 3-Opt is normally preferable than 2-Opt as the $\overline{T}$ is large.

Then the *partial-run* mode is checked by two cases: #NA100 and #N400. For the #NA100 and the #Std, the time to be costed at the initialization stage is same. Although the average Δavg is slight worse, the R_{GT} execution times of #NA100 is 50.8% of that of #Std. For #N400, it can achieve quite good performance in reasonable run-time (*sec*), where the fails only occurs in five instances, which are mainly due to the enriched socially-shared information and preserved diversity among the larger society.

5. CONCLUSIONS

This paper has described an extended multiagent optimization system ($MAOS_E$) based on autonomy oriented computing, which consists of a society of agents under ecological rationality where they cooperate in a sharing environment to search on a virtual landscape, for solving hard computational problems. Although each agent only possesses limited memory and simple behavioral rules, the collective properties can emerged from their cooperation.

From the viewpoint of computation, each agent is a stochastic algorithm. Although the agents have same organized structure, they become heterogonous since their memories are initialized with different contents. If the agents do not utilize the information in M_S, then the system becomes a *portfolio of algorithms* [20][26]. The studies [20][26][33] have indicated the portfolios may have unequivocal advantage than any of its individual member algorithms running alone for dealing with hard computation problems. In $MAOS_E$, the agents are cooperated through the information located in M_S. Then the multiagent system becomes a *cooperative algorithm portfolio*. The cooperation between agents will introduce statistical correlations between their performance, which is capable of reducing both the expected time and the solving risk when manifested in negative correlations [26].

Under the $MAOS_E$, the realization of a problem solver then can be divided into three parts: a) encode the representation of the problem into the landscape and the possible auxiliary knowledge; b) construct the memory elements at the initialization stage; and c) design the *generate-and-test* behavior guided by the law of *socially-biased individual learning*. The implementation has been demonstrated on the TSP, where all three parts are realized by exploiting the problem features. The extensive experimental results on real-world instances in TSPLIB show its efficiency as comparing to the ILS-FDD and the IBGLK.

One possible extension of current work is to apply the agents for solving other well-studied hard computational problems, which are key components for many practical tasks, by utilizing inherent features. Another possible extension is to study rational strategies for agents to deploy multiple R_{GT} rules under certain knowledge.

The agents may also be adapted to the cooperative distribute problem solving (CDPS) [13][52] by requesting the guiding information from other agents. Of course, the negotiation mechanisms then may be required, especially as the agents use various virtual landscapes for involved task.

6. REFERENCES

[1] Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. An integrated theory of the mind. *Psychological Review*, 111: 1036-1060, 2004.

[2] Applegate, D., Cook, W., and Rohe, A. Chained Lin-Kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, 15: 82-92, 2003.

[3] Baddeley, A. Exploring the central executive. *The Quarterly Journal of Experimental Psychology A*, 49: 5-28, 1996.

[4] Bak, P. *How Nature Works: The Science of Self-Organized Criticality*. Berlin: Springer, 1996.

[5] Bandura, A. *Social Foundations of Thought and Action: a Social Cognitive Theory*. NJ: Prentice-Hall, 1986.

[6] Bentley, J. L. Fast algorithms for geometric traveling salesman problems. *ORSA J. Computing*, 4: 387-411, 1992.

[7] Boese, K. D., Kahng, A. B., and Muddu, S. A new adaptive multi-start technique for combinatorial global optimizations. *Operation Research Letters*, 16: 101-113, 1994.

[8] Boyd, R. and Richerson, P. J. *The Origin and Evolution of Cultures*. New York: Oxford University Press, 2005.

[9] Chan, C. H., Lee S. A., Kao C. Y., et al. Improving EAX with restricted 2-opt. *Genetic and Evolutionary Computation Conference*, Washington DC, USA, 1471-1476, 2005.

[10] Cheeseman, P., Kanefsky, B., and Taylor, W. M. Where the really hard problems are. *International Joint Conference on Artificial Intelligence*, San Mateo, CA, 331-337, 1991.

[11] Cook, W. and Seymour, P. Tour merging via branch-decomposition. *INFORMS J. Computing*, 15: 233-248, 2003.

[12] Dorigo, M., Maniezzo, V., and Colorni, A. The ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Systems, Man, and Cybernetics - Part B*, 26: 1-13, 1996.

[13] Durfee, E. H., Lesser, V. R., and Corkill, D. D. Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering*, 1: 63-83, 1989.

[14] Fredman, M. L., Johnson, D. S., McGeoch, L. A., and Ostheimer, G. Data structures for traveling salesmen. *Journal of Algorithms*, 18: 432-479, 1995.

[15] Freisleben, B. and Merz, P. New genetic local search operators for the traveling salesman problem. *International Conference on Parallel Problem Solving from Nature*, Berlin, Germany, 890 - 899, 1996.

[16] Galef, B. G. and Laland, K. N. Social learning in animals: Empirical studies and theoretical models. *BioScience*, 55: 489-499, 2005.

[17] Gent, I. P. and Walsh, T. The TSP phase transition. *Artificial Intelligence*, 88: 349 - 358, 1996.

[18] Gigerenzer, G. *Adaptive Thinking: Rationality in the Real World*. New York: Oxford University Press, 2000.

[19] Glenberg, A. M. What memory is for. *Behavioral and Brain Sciences*, 20: 1-55, 1997.

[20] Gomes, C. P. and Selman, B. Algorithm portfolios. *Artificial Intelligence*, 126: 43-62, 2001.

[21] Gu, J. and Huang, X. F. Efficient local search with search space smoothing: A case study of the traveling salesman problem (TSP). *IEEE Transactions on Systems, Man, and Cybernetics*, 24: 728-735, 1994.

[22] Held, M. and Karp, R. M. The traveling salesman problem and minimum spanning trees. *Operations Research*, 18: 1138-1162, 1970.

[23] Helsgaun, K. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126: 106-130, 2000.

[24] Hogg, T., Huberman, B. A., and Williams, C. P. Phase transitions and the search problem. *Artificial Intelligence*, 81: 1-15, 1996.

[25] Hordijk, W. A measure of landscapes. Evolutionary Computation, 4: 335-360, 1996.

[26] Huberman, B. A., Lukose, R. M., and Hogg, T. An economics approach to hard computational problems. *Science*, 275: 51-54, 1997.

[27] Jennings, N. R., Sycara, K., and Wooldridge, M. A roadmap of agent research and development. Autonomous Agents and Multi-Agent Systems, 1: 7-38, 1998.

[28] Jung, S. and Moon, B. R. Toward minimal restriction of genetic encoding and crossovers for the two-dimensional Euclidean TSP. *IEEE Transactions on Evolutionary Computation*, 6: 557-565, 2002.

[29] Lin, S. and Kernighan, B. W. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21: 498-516, 1973.

[30] Liu, J., Han, J., and Tang, Y. Y. Multi-agent oriented constraint satisfaction. *Artificial Intelligence*, 136: 101-144, 2002.

[31] Liu, J., Jin, X. L., and Tsui, K. C. *Autonomy Oriented Computing (AOC): From Problem Solving to Complex Systems Modeling*: Kluwer Academic Publishers, 2005.

[32] Martin, O. C., Otto, S. W., and Felten, E. W. Large-step Markov chains for the TSP incorporating local search heuristics. *Operations Research Letters*, 11: 219-224, 1992.

[33] Maurer, S. M., Hogg, T., and Huberman, B. A. Portfolios of quantum algorithms. *Physical Review Letters*, 87: 257901, 2001.

[34] Nagata, Y. The EAX algorithm considering diversity loss. *International Conference on Parallel Problem Solving from Nature*, Birmingham, UK, 332-341, 2004.

[35] Nagata, Y. and Kobayashi, S. Edge assembly crossover: a high-power genetic algorithm for the traveling salesman problem. *International Conference on Genetic Algorithms*, East Lansing, MI, USA, 450-457, 1997.

[36] Newell, A. The knowledge level. *AI Magazine*, 2: 1-20, 1981.

[37] Newell, A. and Simon, H. A. Human Problem Solving. Englewood Cliffs, NJ: Prentice-Hall, 1972.

[38] Olick, J. K. and Robbins, J. Social memory studies: From "collective memory" to the historical sociology of mnemonic practices. *Annual Review of Sociology*, 24: 105-140, 1998.

[39] Reinelt, G. TSPLIB - a traveling salesman problem library. *ORSA Journal on Computing*, 3: 376-384, 1991.

[40] Reinelt, G. *The Traveling Salesman: Computational Solutions for TSP Applications*. Berlin: Springer, 1994.

[41] Romney, A. K., Boyd, J. P., Moore, C. C., et al. Culture as shared cognitive representations. *Proc. Natl. Acad. Sci. USA*, 93: 4699-4705, 1996.

[42] Seo, D. I. and Moon, B. R. Voronoi quantized crossover for traveling salesman problem. *Genetic and Evolutionary Computation Conference*, New York, 544-552, 2002.

[43] Smith, E. E. and Jonides, J. Storage and executive processes in the frontal lobes. *Science*, 283: 1657-1661, 1999.

[44] Stadler, P. F. and Schnabl, W. The landscape of the traveling salesman problem. *Physics Letter A*, 161: 337-344, 1992.

[45] Stützle, T., Grün, A., Linke, S., and Rüttger, M. A comparison of nature inspired heuristics on the traveling salesman problem. *International Conference on Parallel Problem Solving from Nature*, Paris, France, 661-670, 2000.

[46] Stützle, T. and Hoos, H. H. MAX-MIN ant system. *Future Generation Computer Systems*, 16: 889-914, 2000.

[47] Tsui, K. C. and Liu, J. Multiagent diffusion and distributed optimization. *International Joint Conference on Autonomous Agents and Multiagent Systems*, Melbourne, Australia, 169-176, 2003.

[48] Walters, T. Repair and brood selection in the traveling salesman problem. *Int. Conf. on Parallel Problem Solving from Nature*, Amsterdam, the Netherlands, 813-822, 1998.

[49] Watson, J., Ross, C., and Eisele, V., et al. The traveling salesrep problem, edge assembly crossover, and 2-opt. *International Conference on Parallel Problem Solving from Nature*, Amsterdam, the Netherlands, 823-832, 1998.

[50] Wolpert, D. H. and Macready, W. G. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1: 67-82, 1997.

[51] Xie, X. F. and Liu, J. A compact multiagent system based on autonomy oriented computing. *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, Compiègne, France, 38-44, 2005.

[52] Yakoo, M., Durfee, E. H., Ishida, T., and Kuwabara, K. Distributed constraint satisfaction for formalizing distributed problem solving. *International Conference on Distributed Computing Systems*, Yokohama, Japan, 614-621, 1992.

[53] Zhang, W. Phase transitions of the asymmetric traveling salesman. *International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 1202-1207, 2003.

[54] Zhang, W. and Looks, M. A novel local search algorithm for the traveling salesman problem that exploits backbones. *International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, 343-348, 2005.

Table 3. The average results on TSPLIB instances (1000≤*V*<3000) by ILS-FDD [45], IBGLK [54] and $MAOS_E$ cases

F	ILS-FDD (*sec*)	IBGLK (*sec*)	#NA100 ($\overline{T}$)	#N400 ($\overline{T}$ /*sec*)	#Std ($\overline{T}$ /*sec*)	#D1.0 ($\overline{T}$)	#NN ($\overline{T}$)	#2Opt ($\overline{T}$)
dsj1000	0.0091(604.2)	0.01 (2066)	1E-04 (119.5)	**0** (89.4/477)	**0** (78.8/203.9)	3E-4 (58.9)	**0** (100.6)	**0** (103.8)
pr1002	**0** (93.5)	**0.00** (345)	**0** (76.9)	**0** (59.52/338.6)	**0** (56.68/126.6)	**0** (44.92)	**0** (78.4)	**0** (106.08)
u1060	0.009 (671.8)	0.02 (873)	**0** (106.9)	**0** (67.3/411.1)	**0** (63.4/351.7)	**0** (51.3)	**0** (86.3)	**0** (109.6)
~~si1032~~	-	~~**0.00** (772)~~	~~**0** (2.1)~~	~~**0** (1.7/51.7)~~	~~**0** (1.5/21.7)~~	~~**0** (2.5)~~	~~1E-4 (9.2)~~	~~8E-4 (9.8)~~
vm1084	0.011 (340.8)	0.02 (403)	**0** (67.0)	**0** (55.8/388.9)	**0** (52.5/327.9)	**0** (42.9)	**0** (69.9)	**0** (69.9)
pcb1173	0.0011 (493.8)	**0.00** (213)	0.004 (86.9)	7E-4 (68.04/466.4)	6E-4 (61.88/168.9)	0.003 (55.4)	4E-4 (71.7)	**0** (105.7)
d1291	**0** (320.5)	0.08 (1086)	**0** (39.2)	**0** (30.04/345.1)	**0** (28.48/137.7)	**0** (22.36)	5E-4 (39.2)	0.003 (47.2)
rl1304	**0** (23.6)	**0.00** (504)	0.014 (55.8)	**0** (39.8/339.1)	**0** (39.9/162.7)	0.007 (33.6)	0.002 (49.1)	**0** (50.8)
rl1323	**0** (148.3)	**0.00** (582)	**0** (55.3)	**0** (40.7/353.2)	**0** (35.2/279.4)	**0** (30.4)	0.005 (49.3)	**0** (55.5)
nrw1379	0.051 (867.3)	0.06 (440)	0.006 (147.5)	0.003 (102.8/417.2)	0.004 (88.4/385.1)	0.003 (72.8)	0.005 (116.8)	0.006 (176.4)
fl1400	**0** (218.2)	**0.00** (19317)	0.002 (147.7)	**0** (111.1/460.5)	**0** (102.3/448.0)	**0** (94.9)	**0** (115.8)	**0** (116.8)
u1432	0.039 (763.4)	**0.00** (617)	**0** (97.4)	**0** (67.5/334.5)	**0** (59.7/570.3)	**0** (53.5)	0.003 (75.5)	0.008 (132.4)
fl1577	**0** (648.5)	**0.00** (10430)	0.01 (102.1)	0.017 (72.3/690.6)	0.027 (47.5/239.9)	0.034 (42.8)	0.053 (84.7)	0.009 (98.8)
d1655	**0** (803.1)	**0.00** (1869)	8E-04 (94.8)	**0** (67.3/777.3)	**0** (60.7/325.1)	**0** (46.8)	6E-4 (94)	0.002 (129)
vm1748	0.031 (1267.4)	0.01 (829)	**0** (128.3)	**0** (96.1/1133.1)	**0** (86.7/612.5)	**0** (69.3)	**0** (108)	**0** (124.8)
u1817	0.056 (1316.3)	0.16 (680)	0.001 (136.4)	**0** (91.4/1125.2)	0.003 (74.3/439.7)	0.006 (62.9)	7E-4 (101.3)	0.008 (126.3)
rl1889	0.0064(1101.4)	0.22 (1110)	**0** (95.7)	**0** (72.8/567)	**0** (65.8/525.1)	4E-4 (53.3)	**0** (88.9)	**0** (91.2)
d2103	0.001 (1747.7)	**0.00** (8669)	0.036 (100.6)	0.003 (63.9/495.5)	0.019 (44.7/365.6)	0.004 (41.5)	0.076 (50.6)	0.005 (107.5)
u2152	0.061 (2401.9)	0.19 (808)	0.002 (114.8)	**0** (79.4/714.7)	0.002 (71.9/662.7)	5E-4 (57.6)	**0** (84.9)	8E-4 (149)
u2319	0.11 (1233.8)	0.04 (2091)	0.119 (143.5)	0.041 (163.1/1434.6)	0.038 (132.1/1251.8)	0.025 (177.2)	0.067 (158.5)	0.049 (256.2)
pr2392	0.004 (2079.2)	0.14 (715)	**0** (175.5)	**0** (129/1159.6)	**0** (120.5/1123.0)	**0** (89.9)	**0** (152.7)	**0** (225.1)
Average	***0.019 (857.2)***	***0.047 (2682.3)***	***0.0098 (104.6)***	***0.0032 (78.4/621.5)***	***0.0047(68.6/435.4)***	***0.0041(60.1)***	***0.010 (85.0)***	***0.0045 (119.1)***

Social Coordination without Communication in Multi-Agent Territory Exploration Tasks

Paul Schermerhorn*
Artificial Intelligence and Robotics Laboratory
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
pscherm1@nd.edu

Matthias Scheutz
Artificial Intelligence and Robotics Laboratory
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
mscheutz@nd.edu

ABSTRACT

In the recent past, several different methods for coordinating behavior in multi-robot teams have been proposed. Common to most of them is the use of communication to coordinate behavior. For many practical applications, however, communication might not be an option (e.g., because of energy constraints of embedded platforms, limited communication range of wireless transmitters, security risks of potential interception of messages in hostile territory, etc.).

In this paper we examine low-complexity, low-cost strategies without communication for coordinated agent behavior. Specifically, we investigate the utility of a "social preference mechanism" and a "pairing mechanism" in territory exploration tasks, where agents have to explore their environment to find and visit k checkpoints, which only count as "visited" when n agents are present at them at the same time. Experimental results indicate that pairing is the better strategy, raising interesting questions about tradeoffs between agent complexity and group size (e.g., whether fewer, more expensive agents with the ability to visit checkpoints individually are a better choice than more less expensive agents).

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Experimentation, Performance

Keywords

Cooperation and coordination among agents, collective and emergent agent behavior, social choice mechanisms

*Student Author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

Efficient performance of multi-agent tasks often requires some form of coordination between agents. Even in seemingly simple tasks, where agents perform autonomously without the need for help from others, agents may work at odds with one another, leading to wasted effort and reduced efficiency. More complex tasks may even *require* coordination in order to perform them. Communication is often used to facilitate coordination in multi-robot tasks, yet it may not always be feasible to include communication. Other investigations use behavior coordination among robotics agents to study the utility of communication. Arkin and Hobbs [2, 9], for example, compare results with and without communication on robotic *retrieval tasks*. Communication was found to be beneficial if agents communicated their states, which could cause other agents to join in the task. Werger [11], on the other hand, found communication to be unnecessary to achieve task formation in a system which uses behavior-based mechanisms to generate cooperative behaviors, as did Quinn et al. ([6]). Ijspeert et al. [4, 5] found that signaling is helpful when the number of agents is small relative to the number of subtasks. While some of these results are in line with the conclusions of this paper, the tasks involved are very different from the one described below.

Clearly the benefits of communication are inconclusive. Factors in determining the feasibility of communication for multi-robot exploration teams include weight (communication will require additional equipment), communication range, the cost of transmitters, and the energy required to communicate and process communicated information (i.e., communication in itself is useless without some process that utilizes the extra information). In many cases, these additional factors eliminate communication as an option for coordination, so a low-cost, low-complexity strategy is needed for coordination instead. Such a strategy would avoid the high cost of communication, but must provide sufficient coordination to allow good task performance.

In this paper, we present two coordination mechanisms, a "social preference mechanism" and a "pairing mechanism" for multi-agent coordination. We test these new mechanisms in *multi-agent territory exploration* (MATE-n) tasks, which require a group of agents to visit all checkpoints in the environment. In order for a checkpoint to count as "visited," n agents must be present simultaneously (consider a task in which extraplanetary landers with heterogeneous equip-

ment must explore the terrain, but may not be able, individually, to accomplish scientific goals at interesting sites). When $n > 1$, the degree of coordination required to accomplish the task efficiently is substantial. Experimental results show that, while the social preference mechanism is able to improve performance, the pairing mechanism is the better choice for coordination in MATE-n tasks.

This paper proceeds as follows: first MATE-n is specified in detail. Then the methods used are described, including the agent model that serves as a base for the experiments, as well as the experimental setup used. Then experiments and results are presented for the base agent, social preference agents, and pairing agents.

2. THE MATE-N TASK

The *multi-agent territory exploration* (MATE-n) task is inspired by biological foraging tasks, but finds applications in other domains (e.g., mine-detecting robots or frogs searching for mating sites) [8, 7]. Agents explore their environment, looking for marks indicating the presence of a checkpoint. When enough agents (n) are present at a checkpoint, its mark is removed and the agents continue the task, exploring the environment until all checkpoints have been visited by n agents simultaneously (note that agents do not need to *arrive* at the checkpoint simultaneously, only to be present simultaneously). More formally:

Definition 1: A MATE-n task $T(C_n, A, R, D)$ requires a group of agents A, each with sensory range R, to visit a set of checkpoints C, each of which requires n agents present to remove its mark, in a 2D environment where agents and checkpoints are placed according to a probability distribution D.

Performance is defined as the time it takes to remove all checkpoint marks from the environment.[1] Thus, unlike many natural foraging tasks, agents in MATE-n tasks are working as a group. Individual performances are important insofar as they contribute to group performance, but individuals are not rewarded for good performance.

Typically, agents do not have *a priori* knowledge of the distribution D; both agents and checkpoints are distributed randomly in the environment, requiring agents to explore to locate checkpoints. If this information *is* available, it is possible to calculate an optimal offline solution by exhaustively searching the entire space of checkpoints assigned to each agent, along with the order in which each agent visits its assigned checkpoints. Such a solution, however, is computationally intractable.

MATE-n tasks are similar in some ways to Traveling Salesman Problems (TSP), and even more so to multi-Traveling Salesman Problems (MTSP), in which some member of a sales team is required to visit each city. The delivery scheduling problem is an instance of MTSP for which approximations exist [3, 10, 1]. However, because TSP variants require agents to return to their initial positions, no TSP approximation fits exactly the MATE-n task; the optimal visit order is affected by requiring a round trip.

3. METHODS

[1]Other measures of performance could be the total distance traveled by all agents in A, or the difference in length between all agents' paths (as in a load balancing goal)

We begin by investigating the performance of a base agent model in a MATE-n task, then add a new (coordinating) feature F to the architecture and compare the performance of the new agent kind with the performance of the old. The remainder of this section describes the base agent model used to obtain absolute performance, and the experimental setup employed.

3.1 Agent Model

The simplified sensor model employed by the agents ignores sensor modality, assuming that agents can sense all checkpoints in a full 360° radius within a sensory range of R. The agent's sensory input S is combined with its inner states I to produce behaviors according to the agent function F. F is a mapping $F : 2^S \times I \rightarrow D$, where 2^S is the power set of sensory data given as pairs $< d, \alpha >$ of distances $d \in [0, R]$ and angles $\alpha \in [0, 2\pi]$, I is a set of inner states (possibly empty), and $D \in [0, 2\pi]$ is the set of possible directions the agent will move in.

For MATE-1 tasks, a simple agent that targets for collection the closest checkpoint detected is sufficient for reasonably good performance. When $n > 1$, however, these simple agents are susceptible to 'deadlock' states in which each agent moves to the nearest checkpoint and waits for another agent to arrive; when all agents have found (distinct) checkpoints, the task cannot proceed. We therefore introduce the *timeout* agent, which incorporate a *wait timer* w that is started when the agent arrives at a checkpoint and expires after a fixed duration W. At that point, the checkpoint is added to the agent's *filter set*, which forces the agent to ignore the checkpoint. Effectively, the agent does not perceive a checkpoint that is in its filter set. After a fixed cycle count, the checkpoint is removed from the filter set, and the agent can once again perceive the checkpoint (if it is within sensory range).

The following ruleset comprises the agent model for timeout agents:

Rule 1: if no checkpoint is perceived ($S = \emptyset$), perform a random walk $RW(rwd, \beta)$ (i.e., move in the direction of the current heading θ for rwd cycles, then change heading randomly to some value in $[\theta - \beta, \theta + \beta]$)

Rule 2: if some checkpoint C is within visiting range (i.e., the agent is close enough to remove the mark) and there are n agents within visiting range of C, remove the checkpoint's mark and reset w to 0

Rule 3: if some checkpoint C is within visiting range, there are fewer than n agents within visiting range of C, and $w = W$, reset w to 0 and add C to the agent's filter set

Rule 4: if some checkpoint C is within visiting range, there are fewer than n agents within visiting range of C, and $w < W$, increment w

Rule 5: if no checkpoint is within visiting range but some checkpoints are perceived ($S \neq \emptyset$), go directly towards the closest checkpoint (the direction is given by α such that $\min_d [\langle d, \alpha \rangle | \langle d, \alpha \rangle \in S]$)

These simple agents employ a random walk mechanism that changes their heading after rwd cycles without perceiving a checkpoint. The value of rwd was determined for each combination of group size and sensory range employed

by systematically exploring the space of values and selecting those with the best performance. Additionally, the environment is bounded; agents "bounce" off of the edge of the environment with a small amount of random error. Agents will move directly to the nearest perceived checkpoint (when there is one) and wait there for a total of n agents to arrive. After waiting for W cycles, the checkpoint is added to the agent's filter set and the agent can then target other perceived checkpoints or begin foraging via the random walk mechanism. The value of W was determined similarly to *rwd*: a systematic exploration of the performance space of each group size/sensory range combination yielded the value of W with the best performance; this is the value of W for that combination. A counter is associated with each member of the filter set that is incremented at every cycle. When the counter reaches f (i.e., the filter set duration), the checkpoint is removed. The value of f is fixed at 200.

3.2 Experimental Setup

To begin exploring coordination in MATE-n we must choose a reasonable value for n. While there is some degree of coordination required when $n = 1$, for the most part it comes down to keeping out of each other's way; cooperation is not required in visiting individual checkpoints in MATE-1. MATE-2 provides a task in which agents must actively coordinate checkpoint visits if efficient performance is expected. The chances of two agents being at the same checkpoint simultaneously are small if there is no coordination effort. The challenge is similar in kind for $n = 3$, $n = 4$, etc., albeit with increasing difficulty. Hence, for this paper we select $n = 2$ for the number of agents that must be present at a checkpoint.

Each experiment reported in this paper consists of 40 experimental runs using different randomly generated initial conditions in a continuous 2D world, which is limited to an 800 by 800 square region. Sensory range is varied from 25 to 250 in steps of 25, and from 300 to 800 in steps of 50. Group size is varied from 2 (the minimum size required to successfully complete the MATE-2 task) to 10. The same set of 40 initial placements of checkpoints ($|C| = 10$) is used for all experiments. Moreover, the same 40 initial placements of agents is used for all experiments with identical group size (i.e., all experiments with $|A| = 2$ will use the same 40 initial conditions, regardless of sensory range or agent type, etc.). The results reported here are the average cycles to completion of each experiment set for each architectural configuration. Agents are allowed 10,000 cycles within which to complete the task; after that point, the experimental run is considered a failure, and its value is set at 10,000. Thus, 10,000 cycles is the ceiling for performance.

4. TIMEOUT AGENTS

The first set of experiments measures the performance of the base architecture, the timeout agent. Recall that this agent forages for checkpoints in the environment and goes to the nearest one it sees. If there is another agent at the checkpoint, the mark is removed and the agents move on. Otherwise, the agent waits at the checkpoint for W cycles, or until another agent arrives and the mark can be removed.

4.1 Results

The performance results for timeout agents are presented in Figure 1 (left). Overall, the trends are predictable: as sensory range increases, performance increases (i.e., average cycles to completion decreases), to a point. When sensory range is greater than 300, performance levels off, and there is little benefit to increased range. Similarly, performance increases with group size, but improvements diminish as size increases so that, for example, there is little difference in performance between $|A| = 9$ and $|A| = 10$.

A two-way 9x21 ANOVA with *group size* (2-10) and *sensory range* (25-800) as randomized variables and *average cycles to completion* as the dependent variable shows highly significant main effects for group size ($F(8, 7371) = 1983.8795, p < 0.001$) and range ($F(20, 7371) = 207.4370, p < 0.001$), confirming that increasing group size and sensory range improves performance. There is also a highly significant interaction between group size and sensory range ($F(160, 7371) = 5.8574, p < 0.001$); this confirms that larger groups increase performance faster as sensory range increases, reaching earlier the point at which added sensory range is of no benefit. These results are exactly what one would expect; essentially we are increasing the resources available, and should, therefore, expect performance to increase. Similarly, we should expect that a point at which more resources will be of little benefit.[2]

4.2 Analysis

As expected, two trends are evident in the results: larger groups outperform smaller groups, and higher sensory ranges outperform smaller sensory ranges. Several additional functional roles are displayed, each of which is a contributor to poor performance:

Jilted Groom: an agent at a checkpoint whose wait timer expires (the bride never arrives)

Jilted Bride: an agent that targets an occupied checkpoint, but the occupying agent's wait timer expires before the "bride" arrives

Lonely Bachelor: an agent that spends a prolonged period searching for a checkpoint

Unlucky Bachelor: an agent caught in a cycle of waiting for the full length of the wait timer and then foraging again for another checkpoint, only to have its wait timer expire again

Dedicated Bachelor: similar to the unlucky bachelor, except that there are other occupied checkpoints within sensory range, but the agent continues to target unoccupied checkpoints (seemingly preferring to remain a bachelor)

Bridesmaid: an agent that targets a checkpoint that is already targeted by two nearer agents, and so is destined to arrive late, after the checkpoint has been visited ("always a bridesmaid, never a bride")

With low sensory range, agents tend to spend a lot of time as lonely bachelors because of the difficulty of locating a checkpoint. Moreover, once they do locate a checkpoint, they tend to play the part of the jilted groom, because the other agents have the same difficulty locating the occupied checkpoint due to their own low sensory ranges. The wait

[2]These are independent of the type of agent tested, so this pattern will be repeated in subsequent results. Therefore, while we will continue to report the values for the randomized variables group size and sensory range, we will not discuss them specifically in subsequent results, except when they interact significantly with other variables tested.

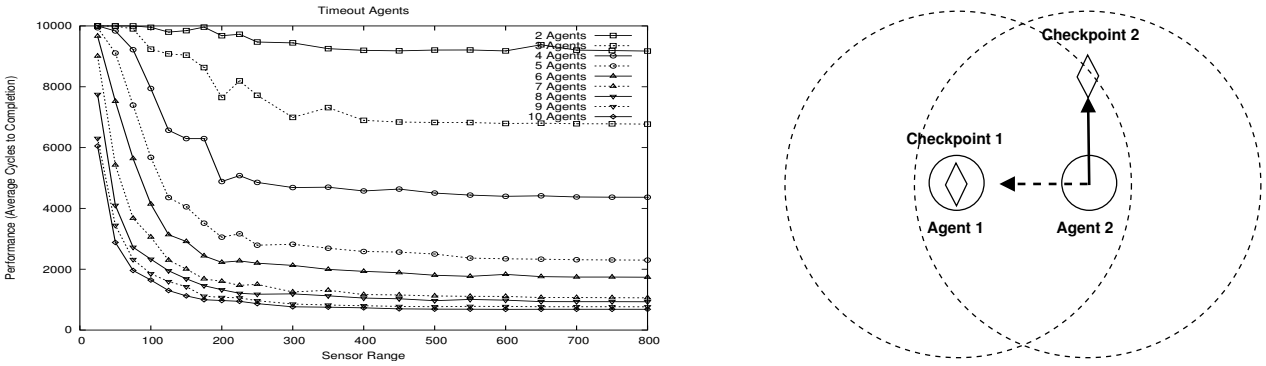


Figure 1: Left: Average performance of timeout agents for two to ten agents and sensory ranges from 25 to 800. Right: A difficult situation for timeout agents to handle: the default behavior of Agent 2 is to move to Checkpoint 2, as a result of Rule 2. However, with Agent 1 at Checkpoint 1, a better strategy would be to move to Checkpoint 1.

timer expires and the groom moves on, returning to the bachelor role. As sensory range increases it becomes easier to locate checkpoints, and the lonely bachelor is seen less often, while the frequency of the groom role increases.

When the number of agents is small, agents have similar difficulties: when they do locate a checkpoint, there is often no other agent in a position to play the role of bride. Instead, because they are unlikely to be placed near each other in the environment, they tend to assume the unlucky bachelor role. Increasing sensory range does not address this problem; agents target the closest checkpoint detected, and there will often be checkpoints between them, preventing potential brides from targeting a groom's checkpoint. Fortunate initial placement can mitigate this problem, and agents may drift toward one another, allowing them to eventually break out of the unlucky bachelor role. However, in the case of 2-agent groups, this seldom happens.

The simulations are limited to 10,000 cycles, as described above. The results for 2 agents in Figure 1 (left) show that they do not even approach 9,000 cycles. This is because they fail to complete in most cases, regardless of the sensory range. 3- and 4-agent groups improve, but still fail in many experimental runs. 5 agents appear to be sufficient to overcome the unlucky bachelor problem.

4.3 Discussion

The timeout mechanism works to break deadlocks by limiting the amount of time agents spend waiting at checkpoints for other agents to arrive. The mechanism is not perfect (e.g., there are times when an agent's wait timer will expire just before another agent arrives, giving rise to the jilted bride), but overall performance is improved. For timeout agents, some wait cycles are inevitable; the first agent arriving at a checkpoint must wait for another agent to arrive (unless we expect them to arrive simultaneously–highly unlikely!). However, some wait cycles are especially wasteful.

Figure 1 (right) presents a scenario in which Agent 1 is waiting at Checkpoint 1 but Agent 2 is closer to Checkpoint 2. In this situation, Agent 2 will move to Checkpoint 2 and wait W cycles. This is less efficient than moving to Checkpoint 1 in most cases, because Agent 1 will wait until its timer expires to leave Checkpoint 1. Furthermore, it is possible that Checkpoint 2 will be in Agent 1's filter set, in which case it will not move there to join Agent 2. Finally, it is possible (and, in our observations, frequently the case) that Agent 2's timer will expire before Agent 1 arrives, in which case they may enter a cycle of flipping between checkpoints without marking any visited. These are examples of the dedicated bachelor functional role, in which Agent 2 appears to intentionally avoid targeting a checkpoint with a waiting groom, although this behavior is really a result of the same control architecture that would require Agent 2 to move to Checkpoint 1 if it were nearer.

As a measure of how often agents make the dedicated bachelor mistake, we tracked *redundant wait cycles*. A redundant wait cycle is one spent by an agent waiting at a checkpoint while another agent is waiting at another checkpoint in sensory range. The average number of redundant wait cycles for all agents in a simulation is the measure of *Redundant Wait Cycles* (RWC) for the simulation:

$$RWC = \frac{\sum_{n=1}^{|A|} RedundantWaitCycles(a[n])}{|A|} \quad (1)$$

Figure 2 (upper left) shows the proportion of the total cycles in Figure 1 that timeout agents spent redundantly waiting. The proportion is substantial, particularly for smaller groups with higher sensory range. Agents in larger groups are not as susceptible to redundant wait cycles because there is more frequently another agent near enough to target the checkpoint at which an agent is waiting. Similarly, when sensory range is low, agents are not as likely to perceive other checkpoints, including other checkpoints with agents waiting. A direct coordination mechanism that targets redundant wait cycles for reduction should be particularly useful for increasing performance in MATE-2 tasks, particularly in small groups with high sensory range.

5. SOCIAL PREFERENCE AGENTS

One simple, and seemingly natural, architectural enhancement is to modify the timeout architecture so that the agent takes into account states of other agents, preferring occupied checkpoints over checkpoints with no agent. The agent model is modified by changing Rule 5:

Rule 5a: if no checkpoint is within visiting range, at least one checkpoint is perceived ($S \neq \emptyset$), and there is an-

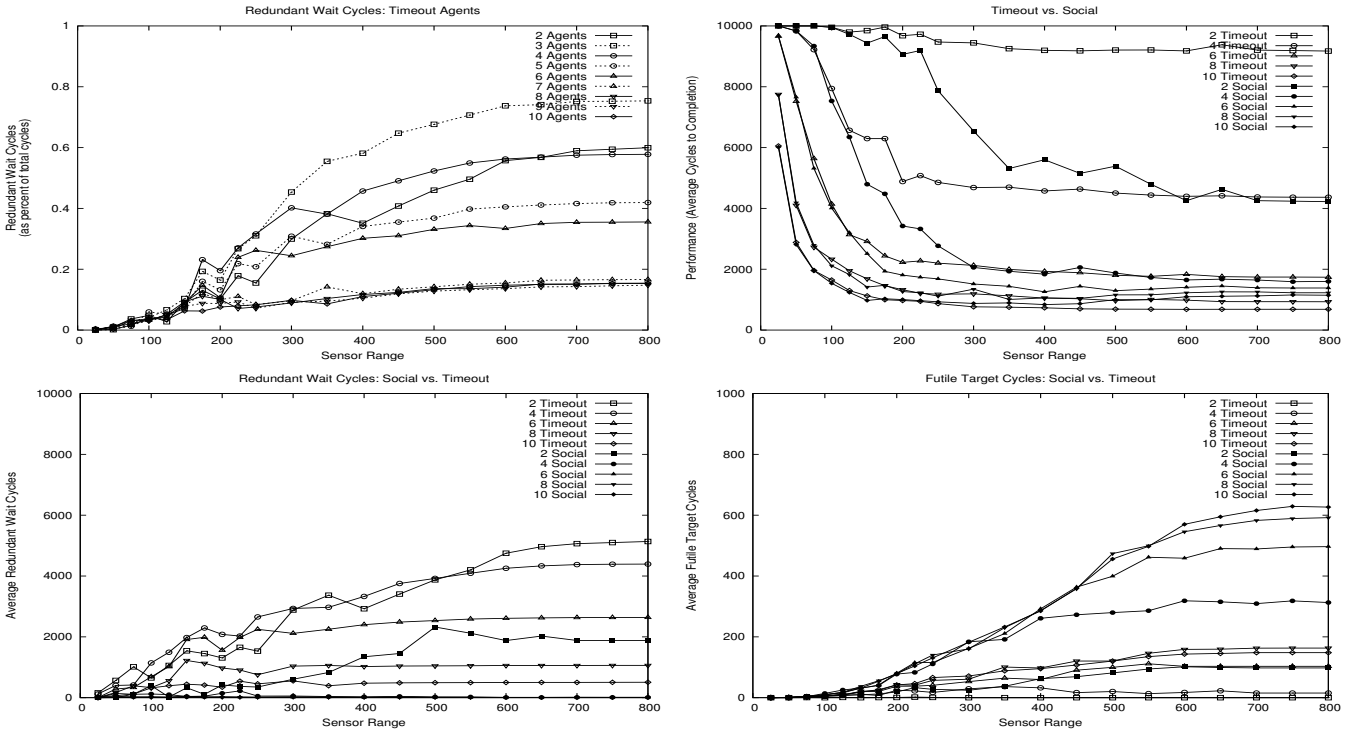


Figure 2: Upper left: Portion of total cycles spent in redundant wait. Upper right: Average performance of timeout agents and social agents. Lower left: Comparison of redundant wait cycles for timeout agents and social preference agents. Lower right: Futile target cycles for target-closest agents and social preference agents.

other agent waiting within visiting range of a perceived checkpoint, go directly towards the closest checkpoint with an agent (the direction is given by α such that $\min_d [\langle d, \alpha \rangle | \langle d, \alpha \rangle \in S]$)

Rule 5b: if no checkpoint is within visiting range, at least one checkpoint is perceived ($S \neq \emptyset$), and there is no other agent waiting within visiting range of a perceived checkpoint, go directly towards the closest checkpoint (the direction is given by α such that $\min_d [\langle d, \alpha \rangle | \langle d, \alpha \rangle \in S]$)

This modified ruleset constitutes the agent model for *social preference* agents. Social preference agents will ignore nearer checkpoints when there is an agent at another checkpoint within sensory range. This modification addresses the inefficiency in Figure 1 (right); social preference agents avoid the dedicated bachelor role in most cases, although they are more susceptible to becoming bridesmaids.

5.1 Results

Social preference agents prefer as targets checkpoints with another agent present. This simple modification leads to dramatically increased performance–in most cases. Figure 2 (upper right) is a comparison of timeout agents and social agents. In virtually all cases, social agents outperform timeout agents. A three-way 9x21x2 ANOVA for *group size* (2-10), *sensory range* (25-800), and *agent kind* (timeout and social preference) as randomized variables with the dependent variable *average cycles to completion* finds highly significant main effects for group size ($F(8, 14742) = 3433.6255, p < 0.001$), sensory range ($F(20, 14742) = 741.4416, p < 0.001$), and agent kind ($F(1, 14742) = 998.0511, p < 0.001$). This confirms that there is an absolute performance difference between groups. There are highly significant two-way interactions between group size and sensory range ($F(160, 14742) = 20.6668, p < 0.001$), between group size and agent kind ($F(8, 14742) = 211.8039, p < 0.001$), and between range and agent kind ($F(20, 14742) = 20.1746, p < 0.001$), and a highly significant three-way interaction between all three factors ($F(160, 14742) = 6.7453, p < 0.001$). The interactions confirm the prediction (based on Figure 2, upper left) that small groups with high sensory range would benefit most from the social preference mechanism. Interestingly, the performance of large groups with high sensory range is actually *harmed* by the social preference modification.

5.2 Analysis

Social preference agents outperform timeout agents overall, with the interaction between group size and agent kind confirming that the mechanism is more effective in small groups, and the interaction between sensory range and agent kind confirming that it is most effective for higher sensory ranges. Performance is nearly identical at small sensory ranges for all group sizes, because in those cases there is rarely an opportunity for Rule 5a to fire; few checkpoints are detected, with or without agents. As sensory range increases, social preference agents soon become better.

The performance advantage of social preference agents is explained by the coordinating effects of the preference modification. Figure 2 (lower left) compares redundant wait cy-

cles of social preference and timeout agents. In most cases, social agents experience fewer.

It may seem surprising that social preference agents experience *any* redundant wait cycles. If they ignore closer checkpoints when they perceive another agent at a checkpoint, the only situation in which two agents should be at different checkpoints within sensory range is when they arrive at their checkpoints simultaneously. This is true when agents perceive all checkpoints within sensory range. However, there is an interaction here between the timeout mechanism's filter set and the social preference mechanism. Consider a situation in which agent A_1 is at checkpoint C_1 and agent A_2 comes within sensory range. A_2 will move toward C_1, whether there are closer checkpoints or not. In the meantime, A_1's wait timer expires. A_1 leaves C_1, placing it in its filter set. While A_1 is traveling to C_2, A_2 arrives at C_1, but A_1 does not perceive C_1, so it does not turn around. When A_1 arrives at C_2, redundant wait cycles begin to accrue.

5.3 Discussion

Social agents see little benefit when sensory range is low, because agents are less likely to see two checkpoints at the same time and, therefore, less likely to be in a situation where their added capability is beneficial. When sensory range is high, agents have more information at any given time to which they can apply the preference. This suggests that the primary obstacle to agent performance is acquiring information; coordination is of little use in the absence of sufficient information about the environment. Once the information hurdle has been overcome, however, the benefit of coordination can be seen in small groups.

Similarly, there is little benefit when group size is high, because there are enough agents in the environment to minimize the need for coordination; agents are not as likely to be penalized by "wrong" decisions, because there is likely to be another agent nearby to visit the occupied checkpoint. Increasing group size is one way to increase the probability that two agents will target the same checkpoint. The value of $|A|$ at which increasing group size is no longer beneficial will depend on the size of the environment and the number of checkpoints that must be visited. In the environments studied, the benefit of introducing the social preference mechanism into groups of eight or more agents is probably not worthwhile. However, for $|A| < 8$, social agents significantly outperform their communicating timeout peers. When group size is small, it will often be the case that there is only one additional agent within sensory range of an occupied checkpoint. In such cases it is important for the second agent to make a good decision, because it is unlikely that another agent will happen along.

Social preference agents are more susceptible to becoming "bridesmaids" because, while they do take the state of the groom into account, they pay no attention to other potential brides. Hence, they spend time futilely targeting a checkpoint when they have no chance of being the bride. A cycle is considered *futile* if the agent targets a checkpoint but subsequently fails to visit it (i.e., because another pair of agents arrives first). More formally, the measure *Futile Target Cycles* (FTC_A^{cyc}) for a simulation run is the average number of futile target cycles for all agents in A in the simulation run (of *cyc* cycles):

$$FTC_A^{cyc} = \frac{\sum_{a \in A} FutileTargetCycles^{cyc}(a)}{|A|} \qquad (2)$$

where $FutileTargetCycles^{cyc}(a)$ is the total number of futile target cycles of agent a in the given simulation run of *cyc* cycles. The average of FTC_A^{cyc} for all simulation runs is taken as a measure of how susceptible the agents were to wasting cycles targeting ultimately unvisitable checkpoints.

As noted, social preference agent performance becomes worse than normal timeout agent performance in large groups with high sensory range. As Figure 2 (lower right) indicates, this is due to an increase in futile target cycles. When social preference agents perceive most or all checkpoints in the environment, any time an agent arrives at a checkpoint, every agent will target that checkpoint, even when it is remote and the agent has no chance of arriving before another agent. This distracts agents from foraging locally, requiring them to return in order to visit checkpoints left behind. Note that, while the difference in scale between the graphs for redundant wait cycles and futile target cycles makes then look similar, the impact of futile target cycles is much smaller. This inefficiency could be dealt with, at least in part, with the addition of mechanisms that observe and use the states of other brides when making control decisions. However, we decided to try a different approach, presented in the next section.

6. PAIRING AGENTS

As we mentioned before, MATE-n tasks for $n > 1$ require greater degrees of coordination than MATE-1 tasks, because when $n = 1$, there is no need to coordinate actual checkpoint visits. Any time an agent arrives at a checkpoint, it is visited. This suggests a different approach to improving performance on MATE-2: rather than adding increasingly more complex mechanisms to attempt to deal with coordination inefficiencies, we can design agents that change the character of the task itself, superimposing a different, simpler task on the original one. Specifically, MATE-2 can be reduced to MATE-1 by simply allowing the agents to "team up" and proceed in pairs. After the initial pairing off phase, agent pairs should be perfectly coordinated: there will never be a wait cycle if both agents always arrive simultaneously. We created *pairing* agents that do this, to see how effective explicitly teaming up can be. The following ruleset constitutes the agent model for the pairing agent A:

Rule S1: if A's $role = scout$ and no other agent is perceived, perform a random walk $RW(rwd, \beta)$
Rule S2: if A's $role = scout$, some other agent O is perceived, O's $role = scout$, and O's leadership characteristic L is greater than A's own, $role = follower$ and $leader = O$
Rule S3: if A's $role = scout$, some other agent O is perceived, O's $role = scout$, and A's own leadership characteristic L is greater than O's, $role = leader$
Rule F1: if A's $role = follower$, follow the leader
Rules L1–L5: if A's $role = leader$, proceed with rules 1–5 for timeout agents, given above, with $W = 10000$

A pairing agent can be in one of three states: scout, leader, and follower. The scouting phase is when the agent is looking for a partner. while scouting, the agent ignores checkpoints, searching only for other agents. Each agent has a "leadership characteristic" L, which is not specified for these agents, but can be any rankable characteristic (e.g., the size of an animal, or the serial number of an extraplanetary rover). When another (scout) agent is located, they

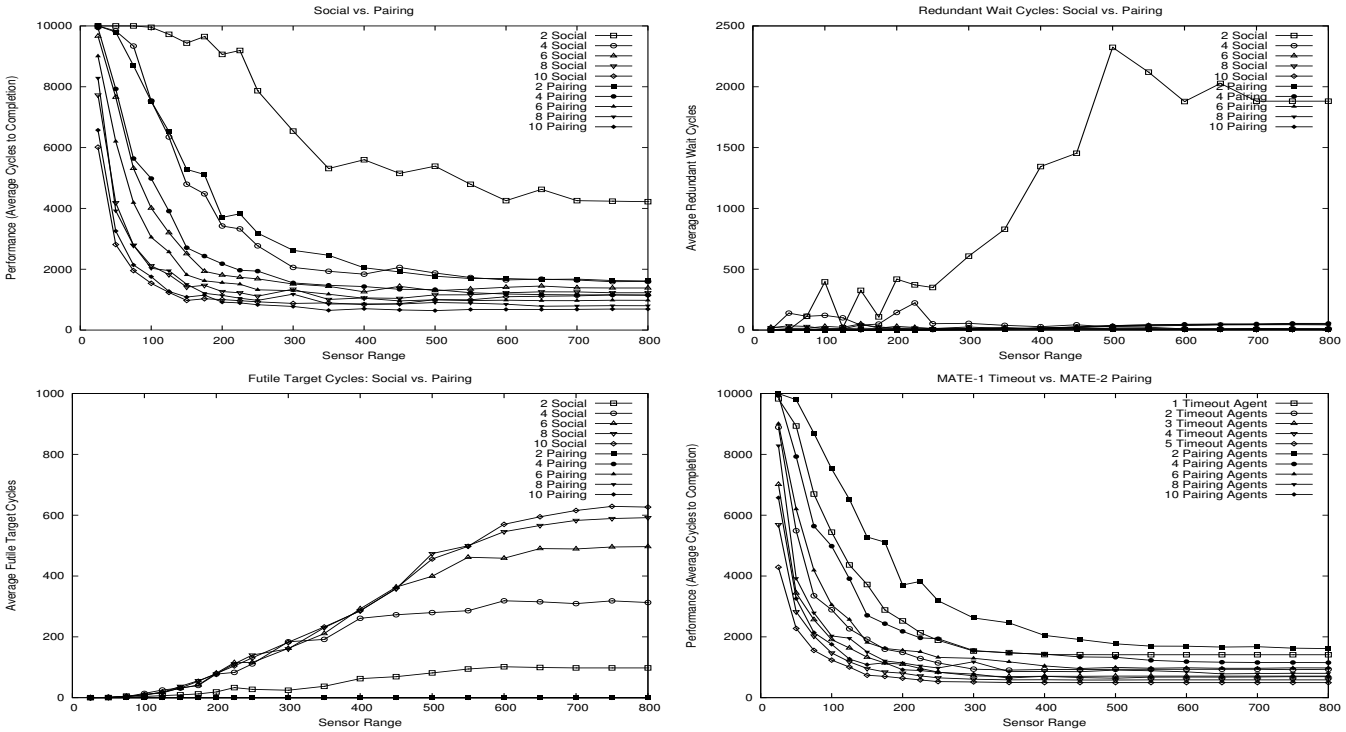


Figure 3: Upper left: Average performance of pairing and social agents. Upper right: Redundant wait cycles for pairing agents and social preference agents. Lower left: Futile target cycles for pairing agents and social preference agents. Lower right: Average performance of 1 to 5 timeout agents in MATE-1 and 2 to 10 pairing agents in MATE-2.

compare their leadership characteristics to determine the leader. If more than two agents are within sensory range, the top two pair off, then the next two, etc. Any odd agent is left out and continues to scout for a partner.

Once agents have formed a team, one enters the leading phase, while the other enters the following phase. In the leading phase, the agent effectively becomes a timeout agent. It explores the environment looking for checkpoints and moving to them when detected. In the following phase, the agent ignores all checkpoints, blindly following the leader wherever it decides to go.

6.1 Results

Figure 3 (upper left) compares pairing agents with non-communicating social agents. The performance improvements in small groups is quite good, although large groups of social agents perform as well as or better than pairing agents at low sensory range. A three-way 9x21x1 ANOVA was conducted with *group size* (2-10), *sensory range* (25-800), and *agent kind* (social and timeout) as the randomized variables and *cycles to completion* as the dependent variable. Group size ($F(8, 14742) = 2357.9752, p < 0.001$), sensory range ($F(20, 14742) = 2039.0519, p < 0.001$), and agent kind ($F(1, 14742) = 1692.0771, p < 0.001$) were all found to be highly significant main effects for cycles to completion, confirming that, overall, pairing agents outperform social preference agents. There is a highly significant two-way interaction between group size and agent kind ($F(160, 14742) = 47.7577, p < 0.001$), confirming that increases in sensory range are most effective in smaller groups, another highly significant two-way interaction between group size and agent kind ($F(8, 14742) = 276.9290, p < 0.001$), confirming that the greatest performance differences between social and pairing agents are to be found in smaller groups, and a final highly significant two-way interaction between sensory range and agent kind ($F(20, 14742) = 13.9703, p < 0.001$), confirming that the performance differences between social and pairing agents varies with sensory range. Finally, a highly significant three-way interaction ($F(160, 14742) = 8.1646, p < 0.001$) is a result of the social agents having greater performance in large groups with low sensory range, but worse performance in the remaining configurations.

6.2 Analysis

Pairing can almost be viewed as the ultimate in coordination for tasks like MATE-n. Once a pair has teamed up, the effect of the above coordination issues is minimal. Figures 3 (upper right) and 3 (lower left) confirm this. These demonstrate that the two sources of inefficiency are virtually eliminated by the pairing mechanism, thereby leading to improved performance. Note that they are not eliminated completely; it is still possible for two leaders to target the same checkpoint, leading to futile target cycles for the second pair. Similarly, leaders and followers do not always arrive at a checkpoint simultaneously,[3] so it is possible, particularly with high sensory range, for two leaders within sensory range to be waiting for their followers to arrive. Note,

[3] They can arrive separately at the first checkpoint they visit, because when they initially pair up, they are at a distance R from each other.

however, that these redundant wait cycles do not have any negative impact on performance.

7. DISCUSSION

The experimental results presented here show that the pairing mechanism is a good solution for coordination in MATE-n, without the need for communication. The degree of coordination achieved is high, after the initial pairing phase. The jilted groom role is eliminated, because agents do not target checkpoints until they have paired with another agent, which will arrive simultaneously, except possibly the first checkpoint visited (see Footnote 3). Similarly, the leader will not leave the checkpoint before the follower arrives, eliminating the jilted bride role. The lonely bachelor can still be a problem, in both the scout and leader/follower phases, when sensory range is low, but this is unavoidable and not a coordination issue. There are no dedicated bachelors, because, again, agents do not begin targeting checkpoints until they are paired with another agent. The final detrimental functional role, the bridesmaid, is the only one that still negatively affects pairing agents, in this case when one *pair* of agents arrives at a checkpoint before another pair. However, the magnitude of its impact is significantly decreased (especially at high sensory ranges) relative to social preference agents, which are particularly susceptible to playing the bridesmaid under these circumstances.

The bottom right graph in Figure 3 compares the performance of pairing agents in MATE-2 with the performance of timeout agents in MATE-1, for the same environmental conditions. The MATE-1 results are for 1–5 agents, whereas the MATE-2 results are for 2, 4, 6, 8, and 10 agents. If we compare the performance of a particular group size A in MATE-1 with the performance of $2 \cdot A$-agent groups in MATE-2, we can get a close approximation of the initial cost of teaming up for pairing agents. The figure shows that, while performance of pairing agents in MATE-2 is similar to the performance of timeout agents in MATE-1, there is a substantial penalty for the initial teaming up period. This suggests that, if it were possible to construct single agents that are capable of performing the task at a checkpoint individually at a reasonable cost, it would be beneficial to do so (thereby completing the reduction of the task to MATE-1). What is a "reasonable cost?" If the new agent kind were to cost less than or equal to twice what the old agent kind cost, this would clearly be a win. Moreover, given the cost of pairing shown in Figure 3, the cost of the enhanced agent could actually be somewhat more than twice that of the original MATE-2 pairing agents, and still achieve a better performance-cost ratio than the pairing agents.

8. CONCLUSION

In this paper we demonstrated that a simple mechanism like pairing can give rise to very high performance in MATE-n tasks. This is not only a good alternative to solutions with communication in situations in which communication is either not possible or is too expensive, but may actually be the better solution overall. Degenerate forms of leadership strategies like this one may in general provide good alternatives to communication for even more complex tasks than MATE-n. Future work will investigate this hypothesis for the $\bigcup_{k=1}^{n}$ MATE$-k$ tasks, in which checkpoints require from 1 to n agents, and there may be no way to predict the value of k for any particular checkpoint without being within sensory range of the checkpoint, where agents can perceive what kind of checkpoint it is (i.e., where they can figure out k). Also, MATE-n tasks in which heterogeneous agent groups are required to visit a checkpoint will be explored. Such heterogeneity requirements are likely to improve the utility of forming an appropriately composed group and proceeding in lockstep; the initial cost of grouping will be higher, however, the cost of breaking the group up due to the random walk mechanism is likely to be much greater.

9. REFERENCES

[1] D. Applegate, W. Cook, S. Dash, and A. Rohe. Solution of a min-max vehicle routing problem. *INFORMS Journal on Computing*, 14(2):132–143, 2002.

[2] R. C. Arkin and J. D. Hobbs. Dimensions of communication and social organization in multi-agent robotic systems. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 486–493, 1992.

[3] V. Bugera. Properties of no-depot min-max 2-traveling-salesmen problem. In S. Butenko, R. Murphey, and P. Pardalos, editors, *Recent Developments in Cooperative Control and Optimization*. Kluwer Academic Publishers, 2003.

[4] A. Ijspeert, A. Martinoli, A. Billard, and L. Gambardella. Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots*, 11(2):149–171, 2001.

[5] K. Lerman, A. Galstyan, A. Martinoli, and A. Ijspeert. A macroscopic analytical model of collaboration in distributed robotic systems. *Artificial Life*, 7(4):375–393, 2001.

[6] M. Quinn, L. Smith, G. Mayley, and P. Husbands. Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors. *Philosophical Transactions of the Royal Society of London, Series A: Mathematical, Physical and Engineering Sciences*, 361:2321–2344, 2003.

[7] M. Scheutz and P. Schermerhorn. Many is more: The utility of simple reactive agents with predictive mechanisms in multiagent object collection tasks. *Web Intelligence and Agent Systems (in Press)*, 3(1), 2005.

[8] M. Scheutz and P. Schermerhorn. Predicting population dynamics and evolutionary trajectories based on performance evaluations in alife simulations. In *Proceedings of GECCO 2005*, 2005.

[9] A. Wagner and R. Arkin. Multi-robot communication-sensitive reconnaissance. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, 2004.

[10] H. Wang and D. Xue. An intelligent zone-based delivery scheduling approach. *Computers in Industry*, 48:109–125, 2002.

[11] B. B. Werger. Cooperation without deliberation: a minimal behavior-based approach to multi-robot teams. *Artificial Intelligence*, 110(2):293–320, 1999.

Analyzing Characteristics of Task Structures to Develop GPGP Coordination Mechanisms

Wei Chen
University of Delaware
Department of Computer & Information Sciences
Newark, DE 19716
wchen@cis.udel.edu

Keith S. Decker
University of Delaware
Department of Computer & Information Sciences
Newark, DE 19716
decker@cis.udel.edu

ABSTRACT

Previous research about multi-agent coordination has concentrated at a high level, e.g. developing communication protocols for coordination, constructing special purpose agents to dictate the coordination behaviors of an entire system, or associating rules or coordination mechanisms with every agent to achieve cooperative behaviors. Much less research addresses multi-agent coordination at a low level: evaluating the effects of agents' task structures upon agents' coordination behaviors. This paper presents an Extended Hierarchical Task Network (EHTN) to represent precisely those structural features that affect coordination. Using this EHTN formalism, an extended set of Generalized Partial Global Planning (GPGP) coordination mechanisms has been developed for multi-agent coordination. Each coordination mechanism is defined in terms of EHTN rewriting rules and an associated set of pre-defined EHTN behaviors. This set of GPGP coordination mechanisms has been applied to a simulated emergency medical service (EMS) system. The experimental results reveal some of the performance relationships between specific mechanisms and external environmental characteristics.

Keywords

Multi-Agent Coordination

1. INTRODUCTION

Well coordinated behaviors can greatly improve agent performance, while inappropriate coordination results in reduced system efficiency, unfinished tasks, misuse of key resources, and even system crashes. It is critical to understand how to achieve well coordinated behaviors for involving intelligent software agents. While much research has focused on high-level coordination approaches, less attention has been focused on the use of task structure detail for coordination.

In order to make use of task structural information, it is key to find a suitable way to represent agents' task structures. STRIPS-style planning systems were developed almost thirty years ago. Most current practical planning systems within the past decade have been based on Hierarchical Task Network (HTN) decomposition. The analysis of the syntax and semantics of HTNs has been well studied [5, 11, 13]. However, the traditional HTNs do not represent what is happening in dynamically changing systems. Particularly, traditional HTNs are not expressive enough to represent certain problems concerning worth-oriented goals, contingencies, and the uncertainties that arise when task plans are in fact distributed over multiple agents. There has been no adequate representation method to capture the task features useful for coordination purposes.

We have developed a highly expressive representation method, Extended Hierarchical Task Networks (EHTNs), to accurately describe the features of agents' tasks, such as information flow and control flow of a complicated task network, the dependency relationship between the tasks of involving agents, and effects of task execution results. We believe that task structural information is helpful for multi-agent coordination, i.e., agents' task structures can be analyzed and manipulated for managing the interdependency relationships among agents.

This paper further articulates a domain-independent approach for specifying an extended set of seventeen GPGP (Generalized Partial Global Planning) coordination mechanisms, which are recast using the EHTN formalism.

We chose a simulated emergency medical service (EMS) system to demonstrate the effectiveness of the extended set of GPGP coordination mechanisms. Based on the experimental results, quantitative and qualitative analysis has been carried out and significant conclusions have been presented to guide agents in selecting the best mechanisms in various environments.

The EHTN task representation language will be presented in Section 2. Our extended set of GPGP coordination mechanisms will be explained in Section 3. A simulated emergency medical service system is briefly introduced in Section 4. The experimental results of the application of these coordination mechanisms in EMS is discussed in Section 5. Finally, we state our conclusion and future work.

2. EXTENDED HIERARCHICAL TASK NETWORKS (EHTNS)

Following the definition of coordination as "managing dependencies between activities"[9], interdependency relationships among multiple agents introduce the problem of con-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

tingency, uncertainty, and worth-oriented goal tradeoffs. Traditional HTNs do not have the language constructs to model the quantitative features of task execution results and the information flow and control flow of the task networks in a distributed environment. For example, one prerequisite input of a task, named TaskA, of Agent A is the result value of a task, named TaskB, of another agent, Agent B; in the absence of any additional information it is unlikely that TaskA in Agent A could be listed as a valid candidate plan step at all. Several approaches have tried to address parts of this problem [7, 10], however, there is no formal representation that is able to quantitatively denote worth-oriented goals, contingencies, and uncertainties distributed across multiple agents. In discussing the Markov decision process (POMDP) approach Boutilier[1] observes that in practice standard AI representations and algorithms survive because real "problems commonly possess structure ... [therefore] specialized representations, and algorithms employing these representations, can achieve computational leverage by exploiting these various forms of structure." It is such a practical approach to extend the existing traditional hierarchical task networks with enriched syntax, semantics, and reasoning capabilities to tackle the aforementioned coordination problem, merging features from TÆMS[12]; the HTN-style agent languages used by RETSINA[**?**] and DECAF[6]; and Erol's formalization of HTNs[5].

It is important to explicitly represent non-local tasks with EHTNs. If one agent's action(s) depends on another agents' action(s), we call this kind of relationship a *coordination relationship* or *interdependency*[4, 12]. An interdependency is *a relationship between a local task (of one agent) and a non-local task (of another agent) where the execution of one changes some performance-related characteristics associated with the other*. A non-local task denotes a task network from a remote agent and is the key to understanding and managing task interdependencies. If non-local tasks can be properly represented, identified, and analyzed, it is possible to develop coordination mechanisms to manipulate non-local tasks based on the task features.

We will briefly present Extended Hierarchical Task Networks (EHTNs) based on the well known work of Erol, et al. [5] and focus primarily on the differences. Note that we only concentrate on an agent's (partially) *local* view—primitive or compound tasks at other agents are represented locally as *non-local tasks* (NLTs). The vocabulary of the language $\mathcal{L}$ is a tuple $\langle V, C, P, F, T, N, NLT, I, O, A, CAF\rangle$, where V, C, P, F, T, N have the same meanings as defined by Erol [5]: V is an infinite set of variable symbols; C is a finite set of constant symbols; P is a finite set of predicate symbols; F is a finite set of *primitive* task symbols, T is a finite set of *compound* task symbols, and N is an infinite set of symbols used for labeling tasks. Our new language constructs are introduced as follows: NLT is a finite set of *non-local task* symbols; I is a finite set of input provision symbols; O is a finite set of outcome symbols including "OK" and "FAIL"; and A is a finite vector of domain-dependent task/action characteristics represented by text values, $\langle A_1, A_2, ..., A_n\rangle$, where A_i is the ith attribute of a task/action (quality, duration, cost, completeness, reputation, etc.). An instance of attribute A_i is shown as a_i. For example, the duration of an information searching task using a specialized database search engine is one minute, the cost is $1.00, and the quality, or satisfaction rate, is 10; while using a free search engine, the corresponding characteristics are two minutes, $0 cost, and the satisfaction rate of 5. CAF is a finite set of characteristic accumulation functions (CAF); a CAF specifies how the value of an attribute A_i of a parent task is derived from the cumulative values of its direct child tasks. Typically functions are *MAX, MIN, SUM*, etc. For example, the cost of a task could be the *sum* of the costs of all its children; the duration of a task is the maximum (*max*) child duration in parallel execution and *sum* in serial execution; and those tasks that only need one child to be completed ("OR" tasks, or "OR" nodes) may have *max* as a quality-CAF, i.e. characteristic accumulation function on quality, etc. We modify Erol's HTN representation to include explicitly named input provisions and outcomes (representing possible contingencies). The information flow and control flow represented with EHTNs can be managed for coordination. Details about the syntax, semantics, and algorithms based on ETHNs are explained in [2]. In Figure 1,

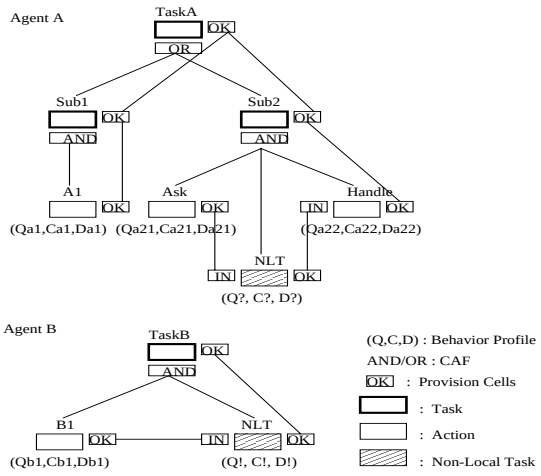


Figure 1: Example Agent Task Structure.

the task networks, TaskA and TaskB, belong to Agent A and Agent B respectively. The new language construct, NLT, of EHTNs means the set of non-local tasks; the task labeled "NLT" within the task network, TaskA, is an example of a non-local task. The cell labeled "IN" associated with the action, "Handle", is a provision, which indicates that the action can not be executed until a value is filled for this provision. The cell labeled "OK" is an output of the action, Handle, which refers to the execution result of this action. The information flow and control flow are indicated by the lines connecting the provisions and outcomes. For example, the outcome "OK" of the action, "Ask", connects to the provision cell, "IN", of the non-local task, "NLT"; the outcome "OK" of "NLT" connects to the provision "IN" of "Handle". The connections indicates that "NLT" will not be executed until "Ask" is finished; and "Handle" will not be executed until a value has been transported from outcome "OK" of "NLT" to "IN" of "Handle". The cell labeled "AND" associated with the sub-task, Sub2, dictates that all child tasks (Ask, NLT, and Handle) should be finished to reach the achievement of their parent task "Sub2". The vector, (Q,C,D), below each action specifies the quantitative features of that action. Notably, the quantitative features of a non-local task are uncertain, which results in the failure of

proper scheduling for Agent A. The existence of uncertainty is because the NLT actually represents the execution of a requested remote task, TaskB of Agent B in this case; Agent A has limited view about other agents and has no knowledge about TaskB. Certain coordination activities may be carried out so that the uncertainty can be removed, as explained in Section 3.

Our proposed coordination process has two steps: (1) locating the interdependencies in the task structure, and (2) applying the coordination mechanisms at those dependencies. We will introduce the application of mechanisms in Section 3. Here we explain the rules for locating interdependencies within agents' tasks.

Given that tasks are represented by our EHTNs, it is easy to detect interdependencies within agents' tasks structures. Any non-local tasks that meet the following conditions are interdependencies, or called *coordination points*, and require the application of a suitable coordination mechanism:[1]

- A non-local task, "NLT", exists within a task network;
- A provision of "NLT" is connected with an outcome of one child action, which shares the same parent task;
- An outcome of "NLT" connects to a provision of another child action of the same parent task.
- There is no other non-local tasks for the parent task.

With the explicit EHTN representation of coordination and the rules for detecting interdependencies among agents' tasks by analyzing structural features, algorithms have been developed for our agents to find out the needs for coordination.

3. AN EXTENDED SET OF GPGP COORDINATION MECHANISMS

Previously, we have introduced the detection of interdependencies by analyzing characteristics of agents' task structures. In this section, we will introduce an extended set of GPGP (Generalized Partial Global Planning) coordination mechanisms and the actual operations for these mechanisms.

Each coordination mechanism consists of (1) a set of coordination communication protocols and (2) a pattern-directed re-writing of our extended hierarchical task networks, EHTNs [2]. Figure 1 shows a general example, Agent A's action, Handle, can not be executed until action B1, is finished by Agent B, (B1 of Agent B enables Handle of Agent A). Thus, Agent B is an *enabler* (enabling the task of a remote agent) and also called a *predecessor*; Agent A is an *enablee* and called a *successor* as well. There are many potential coordination mechanisms for this situation. The coordination component in Agent A detects the interdependency relationships between these two agents (as explained in Section 2), and the task NLT with uncertain characteristics in Agent A is replaced (the task tree is rewritten) by a selected coordination mechanism which may include information communicated about *TaskB* from Agent B.

Our extended set of seventeen GPGP coordination mechanisms are briefly listed and explained as below. *Avoidance* and *sacrificial avoidance*: Agent A removes the branch containing the NLT if the CAF of TaskA is OR (with or without some sacrifice of system quality); *reservation*: Agent A reserves the execution of B1 by Agent B at a future time, so that Agent A is able to schedule its own activities in the mean time; predecessor-side commitments (Agent B): Agent B commits to do task B1 at sometime in the future (*predecessor notice at start*), to do it by a deadline (*predecessor deadline commitment*), to an earliest-start-time (*predecessor EST commitment*), or to notify when complete (*predecessor sending result*); simple successor-side commitments (Agent A): Agent A commits to do the action, Handle, with or without a specific Earliest Start Time (EST) (*successor EST commitment*); Agent A commits to do the action by specifying a deadline (*successor deadline commitment*); polling approach: Agent A consistently queries about the execution of the enabling tasks, B1, (*polling for result*) or the schedule of Agent B (*polling for schedule*); *constant headway / timetabling*: Agent A schedules its tasks according to a constant pattern, e.g., either in a fixed periodic manner (every ten minutes) or following a timetable, e.g, catching a bus based on a published bus schedule; shifting task dependencies: either Agent A sends its task structure (Sub2) to Agent B (*promotion shift*), or Agent B sends B1 to Agent A (*demotion shift*), by learning or mobile code (promotion or demotion); third-party mechanisms: a third agent, Agent C schedules the tasks for both Agent A and Agent B (*third party coordinator*), or Agent C executes B1 for Agent B if Agent B is busy (*third party execution*); *bidding*: Agent A broadcasts a coordination request to all available agents for the engagement of coordination process; or other complex multi-stage negotiation strategies. Details about these coordination mechanisms will be explained in the simulated EMS system shown in Section 4.

Let us explain the operation of *reservation* mechanism as an example. This mechanism is named after the real world activity of reservations. For example, if you want to have dinner in a restaurant, before you go there you had better make a reservation so that at some agreed future time, you will be there and the people there will be ready to serve you to avoid the potential waiting time. Imagine Figure 2 with task $Sub1$ removed so that Agent A has no alternative choice. The reservation mechanism includes a new protocol, instantiated as new task structures (called "GPGP_ReservationR" at Agent B and "GPGP_ReservationE" at Agent A) that processes a message from Agent A indicating a request to do a task (B1 of Agent B) sometime in the future. The reply is a message from Agent B indicating whether the request is agreed or refused; and if agreed, when (if ever) the task will occur. The reservation mechanism rewrites the task structure for Agent A so that a new subtask ("GPGP_ReservationE") is executed and invokes the "GPGP_ReservationR" at Agent B and then processes the return message. The result is an annotation on the non-local task that allows Agent A to predictively schedule task $Sub2$ and execute other potential tasks locally at a time before the enabling task (B1) has been completed.

This mechanism includes both a rewriting of the local task structure and some external communication protocol depicted in Figure 3. Only three rounds of message transfer are needed for the basic reservation process. We assume that there is no need to introduce a confirmation phase in this process. After the re-writing process, the new task structure for Agent A is very similar to that shown in Figure 2,

[1]Formalization of of interdependency detection is presented in [2] together with the full introduction of the EHTNs.

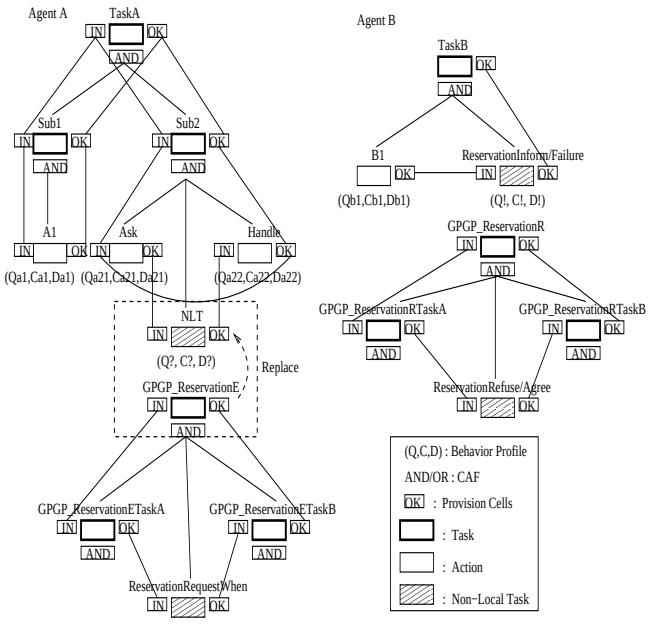


Figure 2: The domain tasks and the inserted GPGP mechanism – coordination by reservation.

except that the task NLT is replaced with the task network GPGP_ReservationE in the dashed box with links is altered. Details about the other mechanisms are in [2].

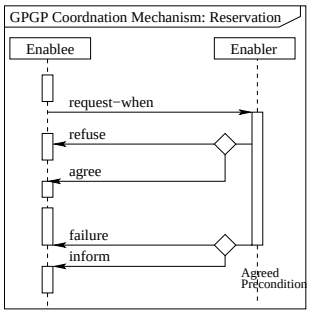


Figure 3: Sequence diagram of GPGP coordination mechanism protocol: Reservation.

4. A SIMULATED EMERGENCY MEDICAL SERVICE (EMS) SYSTEM

Emergency Medical Service, EMS, is defined as a comprehensive, coordinated arrangement of health and safety resources designed to provide expedient care to victims of sudden illness and injury. This paper focuses on *prehospital EMS*, which is generally defined as the response process from the start of an incident to the hospitalization of victims. In EMS systems events happen rapidly and unexpectedly; decision making is highly time-critical based on involving response organizations, institutions, and even geographical sites; doctors, nurses, and paramedics are constantly on the move. Such high-velocity environments pose special computing and communication needs. Information flow among the related entities needs to be highly effective; however, communication channels could be radically impaired based on certain environmental changes, such as traffic jams and power failures. Thus, effective dynamic coordination is key to EMS systems.

A typical EMS process is briefly explained as shown in Figure 4: when an emergency happens, an incident call is reported to the EMS system dispatcher; based on the nature of the incident, e.g. the type of the incident, whether there is any victim at the incident, the dispatcher contacts an appropriate response agency or multiple agencies and sends appropriate vehicles to the incident location; the response vehicle finds its way to the incident location; then proper management of the incident is provided until the incident is considered finished; if any victim needs further medical care, an ambulance will be called to the incident location and the victim will be transported to a proper medical facility by the ambulance; if the incident still requires additional care, other appropriate response vehicles will be dispatched to the incident location as well; after the incident is finished, all participating response vehicles find their ways back to their base stations or move on in response to other incident calls. Further description about the simulated EMS system can be found in [3, 2]. In this paper, we will concentrate on explaining the task structures of response agents within EMS and the application of our extended GPGP mechanisms in this environment.

The first interdependency is named *EMS coordination point one*. The coordination points are also shown as dashed boxes in Figure 4. When an incident happens, the dispatcher needs to deliver the incident message to certain response agents; response agents will send a message back to the dispatcher indicating whether they are available to manage this incident; based on response agents' replies, the dispatcher decides which agent should be selected in response to this incident. That is, the dispatcher can not decide which agent to dispatch for an incident until reply messages arrive; furthermore, the task of a response agent can not start until the notification of the the selection of certain agents. Clearly the dependency is between the message-sending task of the dispatcher and the task of a response agent.

The second interdependency is named *EMS coordination point two* and the task structures of the involving agents are shown in Figure 5[2]. If a police agent has been selected to respond to a police incident, after it arrives at the incident location, it finds that the incident develops and is not simply a police call any more—a victim of the incident needs immediate medical care and potentially needs to be transported to a hospital for treatment; thus the police agent sends a message to a local ambulance agent about the victim; the police agent needs to wait for the arrival of an ambulance; in the meantime the police agent can not carry out other tasks, e.g. to respond to another incident, until the arrival of the ambulance at the incident location; it is easy to figure out that the dependency is between the police agent's tasks and the arrival of the ambulance for immediate medical treatment for victims.

The third interdependency is named *EMS coordination point three*: When an ambulance arrives at the incident location, it finds out that a victim needs to be transported to a local hospital for further treatment; however, open beds

[2]Due to the paper length constraint, the figures for the other two interdependencies are not shown. See [2] for details.

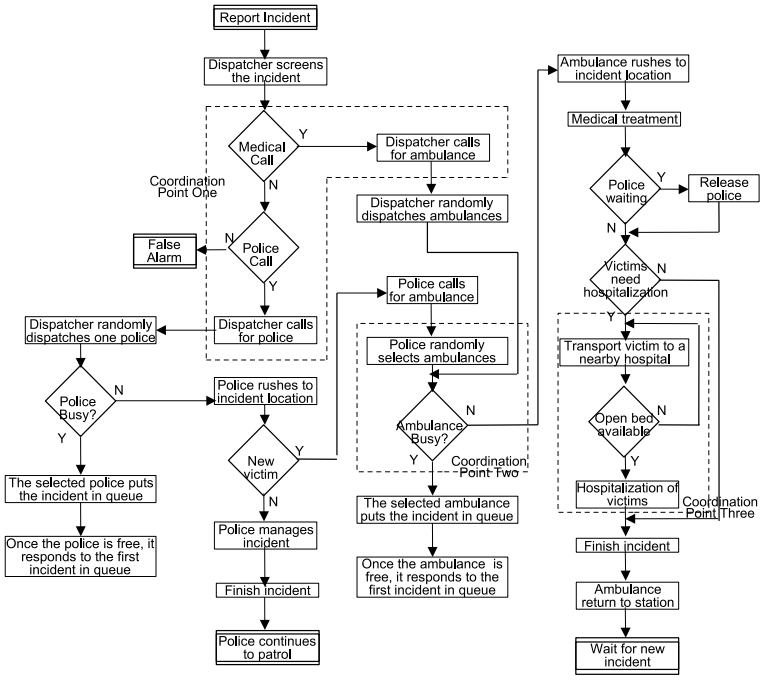


Figure 4: Diagram of EMS operation

may not be always available at hospitals; the hospitalization of a victim for further medical treatment depends on the availability of open beds in the hospital at the time of the victim's arrival; if an ambulance containing the victim arrives at the hospital and there are no any open beds available, the victim has to be transported to another hospital, which is a major delay for further treatment for the victim. In this case the dependency is between further medical treatment for victims by hospitals and the acknowledgment of available open beds from a hospital.

5. EXPERIMENTAL RESULTS

The EMS experimental environment is simplified to include two classes of agents: three police cars and two ambulances move around in local area map in response to incident calls and travel from their base stations (for ambulances) or their current patrol locations (for police agents) to incident locations. There are many other dependencies in an EMS environment. However, we will apply our extended set of GPGP coordination mechanisms to these three coordination points.

Different types of agents have different tasks; tasks for the same type of agents share the same structures. Initial task structures for ambulance, police car, and hospital are shown in Figure 5. These tree-style structures represent tasks for response agents. Top-level root nodes indicate agents' goal tasks. Leaf nodes represent the actions each agent has to execute. The cloud-shape nodes are communication interfaces for an agent to receive and send messages to remote agents. For coordination processes, these communication interfaces could represent non-local tasks as well. The task structures depicts the decomposition of corresponding root tasks into low level tasks until executable actions. The symbols, Input and OK, are the input and output of a task or an action. The links connecting the input and output represent information flow. The meanings of various types of nodes, the task structural information, and task decomposition are further discussed in [6]. Figure 5 only shows main tasks and actions of response agents as a simplified version. The

Table 1: Application of extended GPGP mechanisms to the three coordination points in emergency medical service (EMS) framework.

Index	Mechanisms	P1	P2	P3
1	Avoidance		Y	
2	Sacrifice Avoidance		Y	
3	Reservation		Y	Y
4	Predecessor EST Commitment		Y	
5	Predecessor Deadline Commitment		Y	
6	Predecessor Notice at Start		Y	
7	Predecessor Sending Result		Y	
8	Successor Deadline Commitment	Y	Y	Y
9	Successor EST Commitment	Y	Y	Y
10	Demotion		Y	
11	Promotion			
12	Third Party Execution	Y	Y	Y
13	Third Party Coordinator	Y		
14	Polling for Result	Y	Y	Y
15	Polling for Schedule	Y	Y	Y
16	Constant Headway / Timetabling			Y
17	Bidding	Y	Y	Y

seventeen GPGP coordination mechanisms can be suitably applied to the aforementioned corresponding coordination points as shown in table 5. P1 means Coordination Point One. A "Y" cell in the table shows that the mechanism (to the left of the row) can be applied to the correspond-

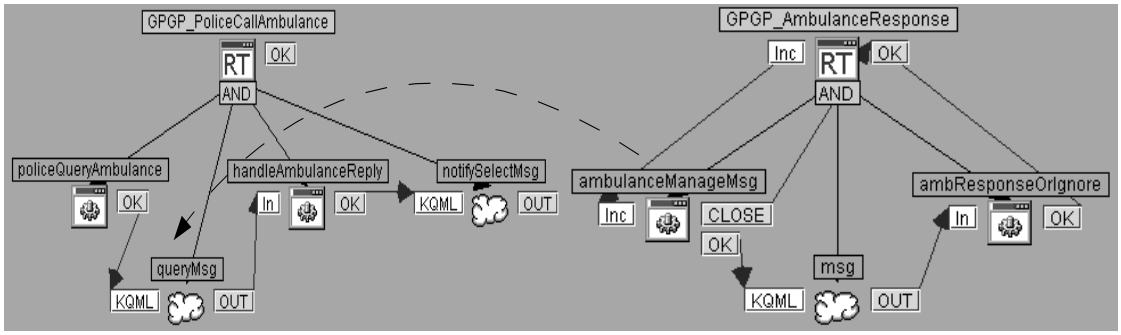


Figure 5: EMS coordination point two: between police and ambulance.

ing coordination point (indicated by the coordination point number to the above of the column); a blank cell indicates the mechanism is not applicable to the corresponding coordination points. For example, the first mechanism, *avoidance*, can be suitably applied to coordination point two, but not to coordination point one and three, simply because there is no alternative local task available; the second mechanism, *reservation*, can not be suitably applied to coordination point one, but can be applied to point two and three; the seventeenth mechanism, *bidding*, can be applied to all three coordination points safely; while *promotion* can not be applied to any of the coordination points.

We define an event, or an accident, that causes a 911 emergency call as an *incident*. There are various types of incidents requiring different response organizations. Coordination during response processes may involve all types of participating agents. An incident is represented as *incident(time, type, severity, location, duration, number, victims[number].(life, life_function))*.

Time means the start time of an EMS process. There are three types of incidents in the model: *medical calls*, *police calls*, and *mixed calls*. A medical call requires ambulances to provide incident victims with primitive treatment and potential transportation to a nearby hospital if needed. A police call requires police agents; it is modeled as a traffic accident or a criminal accident and requires the police to manage the situations, e.g. clearing road or catching criminals. A mixed call starts as a police call and requires police agents at the beginning; upon the arrival of a police agent, the police agent finds out that there are new victims associated with the police call; thus, the police agent calls for ambulances to take care of the victims, as if it was an a medical call. *Severity* indicates whether there are any injured victims associated with the incident and the level of the injury: $(severity = 0)$ means low severity injury and an ambulance is needed for primitive treatment, e.g., CPR or other first aid assistance; $(severity = 1)$ means high severity and an ambulance is not only needed to provide the above basic treatment, but also to transport the victims to a nearby hospital for further medical treatment. An *incident location* means the spot where the incident happens; it is represented with coordinate $Incident.Location(x, y)$; *Duration* indicates how long an incident lasts, e.g how long a response agent is kept on the incident location; in other words, duration indicates the working time for a response agent to manage the incident; an actual duration in experiments follows exponential distribution and will be described later. *Number* represents the number of the victims of an incident; candidate values are (0, 1, 2); if the number is greater than a single ambulance's capacity, multiple ambulances may be dispatched. Notably, the capacity of an ambulance is one, i.e. each ambulance is able to transport one victim at a time. The compound variable, *victim*, is one of the major resources in the EMS framework. *Victim[number].life* represents the victim's initial life value immediately following the incident and this value drops according to the victim's *life function*. A life function indicates how a victim's life value drops as time goes on after an incident. If *life* of a victim drops to zero before entering an appropriate medical facility, it indicates the death of this victim. There are two kinds of life functions defined: sharp linear reduction and flat linear reduction according to time change:(1) $f_1 = (\text{Initial Value} - a_{sharp} * t)$ and (2) $f_2 = (\text{Initial Value} - a_{flat} * t)$, where a_{sharp} and a_{flat} are coefficients indicating sharp and less sharp life value reductions. Based on different values of an incident, different response agents will participate and different operations will be carried out.

In EMS systems, the occurrence of incidents follows a Poisson distribution; and the distribution of time intervals between successive incidents follows an exponential distribution. The time interval, t, from the occurrence of a current incident until the occurrence of a next incident is calculated as the following equation shows.

$$t = \frac{\log(1 - rand())}{-\lambda},$$

where rand() is a uniformly generated random number within the range of $[0, 1)$, $\lambda = \frac{1}{\text{mean}}$, mean is the average value of a series of expected time intervals for next incidents.

There are three main evaluation factors for EMS at a system level: *response time* for quality control, *survival rate* for effectiveness control, and *coordination cost* for efficiency control. In our EMS model, the start of a response time is *when an emergency call is received*; and the end time is *when any appropriate response agent first arrives at the incident location*. Survival rate evaluates the effectiveness of EMS and reflects the overall system performance; survival rate is defined as the rate of victims surviving from an incident reported by a particular EMS system. Coordination cost is another important factor, which reflects system resource consumption by coordination processes, e.g. extra communication for coordination purpose, extra computation for coordination mechanism execution.

The experiments was carried out in a local area network,

agents may reside in different Sun workstations. Each round of experiment consists of one hundred incidents, which have been generated randomly following an exponential distribution at locations in a local area map.

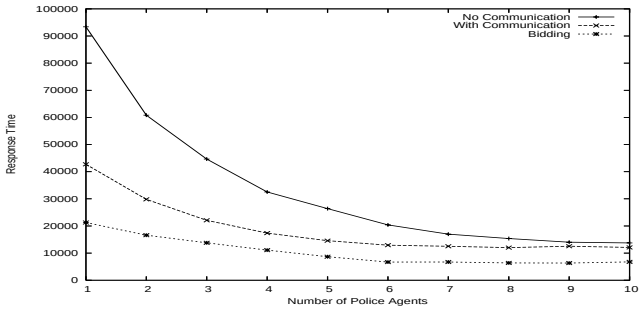


Figure 6: Response time based on changing number of agents (mean=20s).

The first experiment is to estimate the response time of three selected mechanisms, No Communication, With Communication, and Bidding, based on changing number of agents. Different numbers (from one to ten) of agents have been used in each round of experiment. Each response time is an average value of the one hundred response times according to the one hundred incidents for each run. The experimental result is shown in Figure 6. The result indicates that when the number of response agents is limited, e.g., below four, *Bidding* mechanism achieves the shortest response time; coordination without communication has the worse response time; the availability of communication improves response time significantly. As the number of agents increases, the agents are distributed across a local area and the response times approximates with each other when the number is above eight. This experiment leads us to the most suitable number of limited resources (agents) for a certain EMS environment.

Now let us introduce the experiments on the performance of various coordination mechanisms applied to different coordination points in our EMS system. The selection of each kind of response agents is: three police agents, two ambulance agents, two hospitals, one dispatcher. These facts are randomly generated and follow an exponential distribution: next incident arrival time (average arrival time is ten seconds), service time for response agents (average time for a police or an ambulance agent to work at an incident location is two seconds), hospital service schedule time (the average schedule cycle, the arrival of a next victim not from this EMS framework, occupying the open beds, is ten seconds; the service time for a victim is four seconds.). Initial life value of a victim is twenty seconds. The coefficient of life function with sharp reduction is simply one; the coefficient of life function for less sharp reduction (life value decrease slowly.) is 0.5. The EMS evaluation factors of GPGP coordination mechanisms applied to coordination points based on Table 5 will be presented in detail.

As the tasks from Figure 5, Figure 7 shows the sorted average EMS response time of the selected thirteen mechanisms. The *base case* is the situation where there is no GPGP coordination mechanism deployed. If there is an alternative local task, the coordination task can be removed;

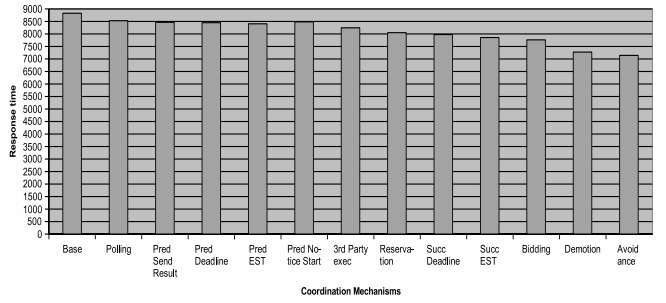


Figure 7: Average response time of selected GPGP Coordination Mechanisms on EMS Coordination Point Two.

thus the *Avoidance* mechanism has the shortest response time. When there is no alternative tasks, *Demotion* mechanism is the best, while *Polling* takes the longest time. Other mechanisms have difference performance in between.

In response to the performance of the selected coordination mechanisms, the shortest response time results in the highest victim survival rate, because a response agent may reach the incident location and provide treatment to the victim immediately. The survival rates of different coordination mechanisms maintain the sequence as in Figure 7 and are shown in Figure 8. The survival rate is affected by the response time: *Avoidance* produces the highest survival rate; *Demotion* is the best if coordination cannot be avoided; *Polling* has the lowest survival rate; other mechanisms fall in between.

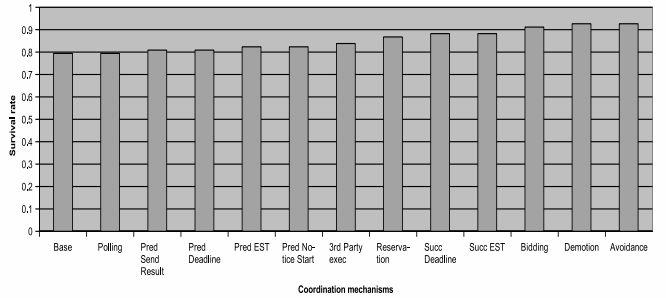


Figure 8: Average survival rate of selected GPGP Coordination Mechanisms on EMS Coordination Point Two.

Figure 9 shows EMS coordination cost versus selected coordination mechanisms applied to coordination point two. Given that *Polling* is carrying out persistent queries, it does results in the highest coordination cost. *Avoidance* has little coordination cost (the only cost is the detection of interdependency and its local alternative task). Other coordination costs by the rest of the mechanisms are as shown.

Due to the distinctive structural features of the three coordination points described earlier, different coordination mechanisms generate different results for the three EMS evaluation factors. We have shown coordination point two in detail as an example. The figures of the experimental results of the mechanisms applied to coordination point one

and three are omitted here. But, the most important factor, EMS response time, of the selected mechanisms for coordination point one and three will be listed as follows. Seven coordination mechanisms have been applied to coordination point one, the order from the longest response time to the shortest is: Base Case, Polling, Successor Deadline Commitment, Success EST Commitment, Third-Party Execution, Bidding, and Third-Party Coordinator. Similarly, eight coordination mechanisms have been applied to point three, and the order from the longest response time to the shortest is: Base Case, Polling, Third-Party Execution, Success EST Commitment, Successor Deadline Commitment, Constant Headway, Bidding, and Reservation.

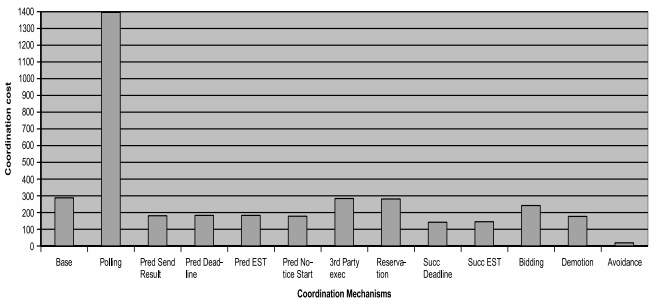


Figure 9: Average coordination cost of selected GPGP Coordination Mechanisms on EMS Coordination Point Two.

The above experiments have shown that different coordination mechanisms may perform best at different coordination points. It is important to see how the overall system performances be improved by a combination of the best mechanisms. In this experiment, we select a *combination* environment: *bidding* mechanism applied to coordination point one, *demotion* applied to point two, *reservation* applied to point three. The base case is that there is no any coordination mechanism applied to any coordination point. The next table shows the result and concludes that a careful

Table 2: System performances caused by a combination of the best mechanisms applied to the corresponding three coordination points .

Environments	ResponseTime	SurvivalRate	TotalCost
Base Case	9523	80%	1010 msec
Combination	3598	100%	624 msec
Improvement	37.8%	125%	61.8%

selection of our extended set of GPGP coordination mechanisms at different coordination points significantly improves the overall system performances.

6. CONCLUSIONS AND FUTURE WORK

We have briefly introduced a highly expressive representation, EHTNs, to explicitly denote the characteristics of agents' task structures. Given EHTNs, interdependencies can be detected and analyzed. An extended set of domain-independent GPGP coordination mechanisms has been developed to address the coordination problem. We also introduced a simulated EMS system for demonstrating the performance of various mechanisms applied to selected interdependencies. We have experimentally demonstrated that the application of our mechanisms improves system performance and different mechanisms may outperform others in different environments (or coordination points within an environment). We have demonstrated that the development of coordination mechanisms based on the analysis and alteration of agents' task structures is a promising approach.

The final experiment about overall system performance is a greedy approach, meaning that selecting the best mechanism for each coordination point without considering potential relationships among these coordination points, i.e. it is not determined whether the result shown in table 5 is the best possible improvement, since other combinations of mechanisms may be better. The experiments about other potential combinations of mechanisms, and of course learning appropriate sets of mechanisms, are part of future work.

7. REFERENCES

[1] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *JAIR*, 11(1):1–94, 1999.

[2] W. Chen. *Designing an Extended Set of Coordination Mechanisms for Multi-Agent Systems*. PhD thesis, Computer and Information Sciences, University of Delaware, 2005.

[3] W. Chen and K. Decker. The analysis of coordination in an information system application - emergency medical services. In *AOIS05*, LNCS-3508, P36–51. Springer, 2005.

[4] K. Decker and J. Li. Coordinating mutually exclusive resources using gpgp. *JAAMAS*, 3(2):133–157, 2000.

[5] K. Erol, D. Nau, and J. Hendler. A critical look at critics in htn planning. In *Proceedings of the 4th IJCAI*, Montreal, Aug. 1995.

[6] J. Graham and K. Decker. DECAF a flexible multi-agent system architecture. *JAAMAS*, 7(1), 2003.

[7] B. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[8] G.Weiss. *Multiagent Systems, A Modern Approach to DAI*, pages 151–157. MIT Press, 1999.

[9] T. Malone and K. Crowston. The interdisciplinary study of coordination. In *ACM Computing Surveys*, pages 87–119, Mar. 1994.

[10] L. Ngo, P. Haddawy, and H. Nguyen. A modular structured approach to conditional decision-theoretic planning. In *AIPS98*, P111–118, 1998.

[11] R. Parr. Flexible decomposition algorithms for weakly coupled markov decision problems. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, Morgan Kaufmann, 1998.

[12] T. W. V. Lesser, K. Decker. Evolution of the gpgp/t?ms domain-independent coordination framework. *JAAMAS*, 1(2):87–143, 2004.

[13] Q. Yang. Formalizing planning knowledge for hierarchical planning. *Computational Intelligence*, 6(1):12–24, Feb. 1990.

Norm-Oriented Programming of Electronic Institutions*

A. García-Camino,
J.-A. Rodríguez-Aguilar, C. Sierra
IIIA-CSIC, Campus UAB, 08193 Bellaterra, Spain
{andres, jar, sierra}@iiia.csic.es

W. Vasconcelos
Dept. of Computing Science, University of Aberdeen,
Aberdeen AB24 3UE, United Kingdom
wvasconcelos@acm.org

ABSTRACT

Norms constitute a powerful coordination mechanism among heterogeneous agents. We propose means to specify and explicitly manage the normative positions of agents (permissions, prohibitions and obligations), with which distinct deontic notions and their relationships can be captured. Our rule-based formalism includes constraints for more expressiveness and precision and allows the norm-oriented programming of electronic institutions: normative aspects are given a precise computational interpretation. Our formalism has been conceived as a machine language to which other higher-level normative languages can be mapped, allowing their execution.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*Law*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multi-agent systems*

General Terms

Languages

Keywords

Norms, Electronic institutions, Multi-agent system programming

1. INTRODUCTION

A major challenge in multi-agent system (MAS) research is the design and implementation of *open* multi-agent systems in which coordination must be achieved among self-interested agents defined with different languages by several designers. Norms can be used for this purpose as a means to regulate the observable behaviour of agents as they interact in pursuit of their goals [13, 2, 3, 8]. There is a wealth of socio-philosophical and logic-theoretical literature on the subject of norms (*e.g.*, [11, 12]), and, more recently, much attention is being paid to more pragmatic and implementational aspects of norms, that is, how norms can be given a computational interpretation and how norms can be factored in in the design and execution of MASs (e.g. [1, 6, 7]).

A normative position [11] is the "social burden" associated with individual agents, namely, their obligations, permissions and prohibitions. Depending on what agents do, their normative positions may change – for instance, permissions/prohibitions can be revoked, or obligations, once fulfilled, may be removed. Ideally, norms, once captured via some suitable formalism, should be directly executed, thus realising a computational, normative environment wherein agents interact. This is what we mean by *norm-oriented programming*. We try to make headway along this direction by introducing an executable language to specify agents' *normative positions* and manage their changes as agents interact via speech acts [10].

In this paper we present a language that acts as a "machine language" for norms on top of which higher-level normative languages can be accommodated. This language can represent distinct flavours of deontic notions and relationships. Although our language is rule-based, we achieve greater flexibility, expressiveness and precision than production systems by allowing constraints to be part of our rules and states of affairs. In this way, normative positions can be further refined. Hence, constraints are considered as first-class citizens in our language.

Although in this paper we restrict to a particular class of MASs, namely electronic institutions [4], our work sets the foundations to specify and implement open regulated MASs via norms.

Our main goal is to produce a language that supports the specification of coordination mechanisms in multi-agent systems by means of norms. For this purpose, we identify below the desirable features we expect in candidate languages.

Explicit management of normative positions. As a result of agents' observable, social interactions, their normative positions [11] change. Hence, the first requirement of our language is to support the *explicit management* of agents' normative positions.

General purpose. We require that our language captures different deontic notions along with their relationships. In other words, the language must be of *general purpose* so that it helps MAS designers to encode any axiomatisation, and thus specify the widest range of normative systems as possible.

Pragmatic. We pursue a "machine language" for norms on top of which higher-level languages can be accommodated. Along this direction, and from a language designer's point of view, it is fundamental to identify the *norm patterns* (*e.g.*, conditional obligation, time-based permissions and prohibitions, continuous obligation, and so on) in the literature to ensure that the language supports

*This work was partially funded by the Spanish Science and Technology Ministry as part of the Web-i-2 project (TIC-2003-08763-C02-00). García-Camino enjoys an I3P grant from the Spanish Council for Scientific Research (CSIC).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

their encoding[1]. In this way, we shall guarantee the expressiveness of our language, and also address pragmatic concerns by providing *design patterns* to guide and ease MAS design.

Declarative. In order to ease MAS programming, we shall also require our language to be *declarative*, with an implicit execution mechanism to reduce the number of issues designers ought to concentrate on. As an additional benefit, we expect its declarative nature to facilitate verification of properties of the specifications.

2. A RULE LANGUAGE FOR NORMS

The building blocks of our language are first-order terms (denoted as τ) and implicitly, universally quantified atomic formulae (denoted as α) without free variables . We shall make use of numbers and arithmetic functions to build terms; arithmetic functions may appear infix, following their usual conventions[2]. We also employ arithmetic relations (*e.g.*, $=, \neq$, and so on) as predicate symbols, and these will appear in their usual infix notation with their usual meaning. Atomic formulae with arithmetic relations represent *constraints* on their variables and have a special status, as we explain below. We give a definition of our constraints, a subset of atomic formulae: a constraint γ is an atomic formula of the form $\tau \lhd \tau'$, where $\lhd \in \{=, \neq, >, \geq, <, \leq\}$. We need to differentiate ordinary atomic formula from constraints. We shall use α' to denote atomic formulae that are *not* constraints.

Intuitively, a state of affairs is a set of atomic formulae. As we will show below, they can store the state of the environment[3], observable agent attributes and the normative positions of agents: a state of affairs $\Delta = \{\alpha_0, \ldots, \alpha_n\}$ is a a finite and possibly empty set of implicitly, universally quantified atomic formulae.

Our rules are constructs of the form *LHS* $\rightsquigarrow$ *RHS*, where *LHS* contains a representation of parts of the current state of affairs which, if they hold, will cause the rule to be triggered. *RHS* depicts the updates to the current state of affairs, yielding the next state of affairs. The grammar in Fig. 1 defines our rules, where x is a variable name and *LHS** is a *LHS* without set constructors (see below). The Us represent the updates: they add (via operator $\oplus$) or remove (via operator $\ominus$) atomic formulae αs. Furthermore, we make use of a special kind of term, called a *set constructor*, represented as $\{\alpha' \mid LHS^*\}$. This construct is useful when we need to refer to all α's for which *LHS** holds,e.g., $\{p(A,B) \mid A > 20 \wedge B < 100\}$ is the set of atomic formulae $p(A,B)$ such that $A > 20$ and $B < 100$.

R	::=	*LHS* $\rightsquigarrow$ *RHS*
LHS	::=	*LHS* $\wedge$ *LHS* $\mid \neg$(*LHS* $\wedge$ *LHS*) $\mid$ Lit
RHS	::=	U $\wedge$ *RHS* $\mid$ U
Lit	::=	$\alpha \mid \neg\alpha \mid x = \{\alpha' \mid LHS^*\}$
U	::=	$\oplus\alpha \mid \ominus\alpha$

Figure 1: Grammar for Rules

We need to refer to the set of constraints that belongs to a state of affairs. We call $\Gamma = \{\gamma_0, \ldots, \gamma_n\}$ the set of all constraints in Δ. Given a state of affairs Δ, relationship $constrs(\Delta, \Gamma)$ holds iff Γ is the smallest set such that for every $\gamma \in \Delta$ then $\gamma \in \Gamma$. In the definitions below we rely on the concept of *substitution*, that is, the set of values for variables in a computation, as well as the concept of its application to a term [5].

We now define the semantics of our rules as relationships between states of affairs: rules map an existing state of affairs to a new state of affairs. We adopt the usual semantics of production rules, that is, we exhaustively apply each rule by matching its *LHS* against the current state of affairs and use the values of variables obtained in this match to instantiate the *RHS* via $\mathbf{s}^*$: $\mathbf{s}^*(\Delta, LHS \rightsquigarrow RHS, \Delta')$ holds iff $\mathbf{s}_l^*(\Delta, LHS, \{\sigma_1, \ldots, \sigma_n\})$ and $\mathbf{s}_r(\Delta, RHS \cdot \sigma_i, \Delta'), 1 \leq i \leq n, n \in I\!N$, hold. That is, two states of affairs Δ and Δ' are related by a rule *LHS* $\rightsquigarrow$ *RHS* iff we obtain all different substitutions $\{\sigma_1, \ldots, \sigma_n\}$ that make the left-hand side match Δ and apply these substitutions to *RHS* (that is, $RHS \cdot \sigma_i$) in order to build Δ'.

Our rules are *exhaustively* applied on the state of affairs thus considering all matching atomic formulae. We thus need relationship $\mathbf{s}_l^*(\Delta, LHS, \Sigma)$ which obtains in $\Sigma = \{\sigma_0, \ldots, \sigma_n\}$ *all* possible matches of the left-hand side of a rule: $\mathbf{s}_l^*(\Delta, LHS, \Sigma)$ holds, iff $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ is the largest non-empty set such that $\mathbf{s}_l(\Delta, LHS, \sigma_i), 1 \leq i \leq n, n \in I\!N$, holds. We now define the semantics of the *LHS* of a rule: $\mathbf{s}_l(\Delta, LHS, \sigma)$ holds between state Δ, the left-hand side of a rule *LHS* and a substitution σ depending on the format of *LHS*:

1. $\mathbf{s}_l(\Delta, LHS \wedge LHS', \sigma)$ holds iff $\mathbf{s}_l(\Delta, LHS, \sigma')$ and $\mathbf{s}_l(\Delta, LHS', \sigma'')$ hold and $\sigma = \sigma' \cup \sigma''$.
2. $\mathbf{s}_l(\Delta, \neg\, LHS, \sigma)$ holds iff $\mathbf{s}_l(\Delta, LHS, \sigma)$ does not hold.
3. $\mathbf{s}_l(\Delta, \alpha', \sigma)$ holds iff $\alpha' \cdot \sigma \in \Delta$ and $constrs(\Delta, \Gamma)$ and $satisfiable(\Gamma \cdot \sigma)$ hold.
4. $\mathbf{s}_l(\Delta, \gamma, \sigma)$ holds iff $constrs(\Delta, \Gamma)$ and $satisfiable((\Gamma \cup \{\gamma\}) \cdot \sigma)$ hold.
5. $\mathbf{s}_l(\Delta, x = \{\alpha' \mid LHS^*\}, \sigma)$ holds iff $\sigma = \{x/\{\alpha' \cdot \sigma_1, \ldots, \alpha' \cdot \sigma_n\}\}$ for the largest $n \in I\!N$ such that $\mathbf{s}_l(\Delta, \alpha' \wedge LHS^*, \sigma_i), 1 \leq i \leq n$

Cases 1-3 depict the semantics of atomic formulae and how their individual substitutions are combined to provide the semantics for a conjunction. Case 4 formalises the semantics of our constraints when they appear on the left-hand side of a rule: we apply the substitution σ to them (thus reflecting any values of variables given by the matchings of atomic formula), then check satisfiability of constraints.[4] Case 5 specifies the semantics for *set constructors*: x is the set of atomic formulae that satisfy the conditions of the set constructor.

We now define the semantics of the *RHS* of a rule: Relation $\mathbf{s}_r(\Delta, RHS, \Delta')$ mapping a state Δ, the right-hand side of a rule *RHS* and a new state Δ' is defined as:

1. $\mathbf{s}_r(\Delta, (\mathsf{U} \wedge RHS), \Delta')$ holds iff both $\mathbf{s}_r(\Delta, \mathsf{U}, \Delta_1)$ and $\mathbf{s}_r(\Delta_1, RHS, \Delta')$ hold.
2. $\mathbf{s}_r(\Delta, \oplus\alpha', \Delta')$ holds iff $\Delta' = \Delta \cup \{\alpha'\}$.
3. $\mathbf{s}_r(\Delta, \oplus\gamma, \Delta') = \mathbf{true}$ iff $constrs(\Delta, \Gamma)$ and $satisfiable(\Gamma \cup \{\gamma\})$ hold and $\Delta' = \Delta \cup \{\gamma\}$.
4. $\mathbf{s}_r(\Delta, \ominus\alpha, \Delta')$ holds iff $\Delta' = \Delta \setminus \{\alpha\}$

Case 1 decomposes a conjunction and builds the new state by merging the partial states of each update. Case 2 caters for the insertion of atomic formulae α' which do not conform to the syntax of constraints. Case 3 defines how a constraint is added to a state Δ: the new constraint is checked whether it can be satisfied with constraints Γ and then it is added to Δ'. Case 4 caters for the removal of atomic formulae.

We extend $\mathbf{s}^*$ to handle sets of rules: $\mathbf{s}^*(\Delta_0, \{\mathsf{R}_1, \ldots, \mathsf{R}_n\}, \Delta_n)$ holds iff $\mathbf{s}^*(\Delta_{i-1}, \mathsf{R}_i, \Delta_i), 1 \leq i \leq n$ hold.

The semantics above define an infinite sequence of states $\langle\Delta_0, \Delta_1, \ldots\rangle$ if $\mathbf{s}^*(\Delta_i, \{\mathsf{R}_1, \ldots, \mathsf{R}_n\}, \Delta_{i+1})$, that is, Δ_{i+1} (obtained by applying the rules to Δ_i) is used to obtain Δ_{i+2} and so on. Fig. 2 illustrates how this sequence can accommodate the intervention of agents sending/receiving messages. The diagram shows an initial state Δ_0 (possibly empty) that is offered (represented by "$\Rrightarrow$") to a set of agents $\{ag_1, \ldots, ag_n\}$. These agents exchange messages, adding a record

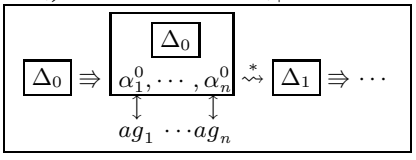


Figure 2: Semantics as a Sequence of Δ's

[1] Work available in `http://www.iiia.csic.es/~andres/NOPLforEIs.pdf`

[2] We adopt Prolog's convention using strings starting with a capital letter to represent variables and strings starting with a small letter to represent constants.

[3] We refer to the *state of the environment* as the set of atomic formulae that represent aspects of the environment in a given point in time.

[4] Our work builds on standard technologies for constraint solving – in particular, we have been experimenting with SICStus Prolog constraint satisfaction libraries.

(via "$\updownarrow$") $\{\alpha_1^0, \ldots, \alpha_n^0\}$ of these messages to Δ_0. After the agents add their utterances, then the rules are exhaustively applied (represented by "$\overset{*}{\rightsquigarrow}$") to $\Delta_0 \cup \{\alpha_1^0, \ldots, \alpha_n^0\}$. The resulting state Δ_1 is, on its turn, offered to agents, and so on.

Our work extends *electronic institutions* (EIs) [4], providing them with an explicit normative layer. There are two major features in EIs : the *states* and *illocutions* (*i.e.*, messages) uttered (*i.e.*, sent) by those agents taking part in the EI. Illocutions I are terms $p(ag, r, ag', r', \tau, t)$ where p is an illocutionary particle (*e.g., inform,ask*); ag, ag' are agent identifiers; r, r' are role labels; τ is a term with the actual content of the message and $t \in I\!N$ is a time stamp. We shall refer to illocutions that may have uninstantiated (free) variables as *illocution schemes*, denoted by $\bar{\mathsf{I}}$. Another important concept in EIs that we employ here is that of a *scene*. Scenes offer means to break down larger protocols into smaller ones with specific purposes.

We differentiate seven kinds of atomic formulae in our state of affairs Δ, with the following intuitive meanings:

1. $oav(o, a, v)$ – object (or agent) o has an attribute a with value v.
2. $att(s, w, \mathsf{I})$ – an agent attempted to get illocution I accepted at state w of scene s.
3. $utt(s, w, \mathsf{I})$ – I was accepted as a legal utterance at w of s.
4. $ctr(s, w, t_s)$ – the execution of scene s reached state w at time t_s.
5. $obl(s, w, \bar{\mathsf{I}})$ – $\bar{\mathsf{I}}$ ought to be uttered at w of s.
6. $per(s, w, \bar{\mathsf{I}})$ – $\bar{\mathsf{I}}$ is *permitted* to be uttered at w of s.
7. $prh(s, w, \bar{\mathsf{I}})$ – $\bar{\mathsf{I}}$ is *prohibited* at w of s.

We only allow fully ground attributes, illocutions and state control formulae (cases 1-4 above) to be present[5]; however, in formulae 5-7 s and w may be variables and $\bar{\mathsf{I}}$ may contain variables. We shall use formulae 4 to represent state change in a scene in relationship with global time passing. We shall use formulae 5–7 above to represent normative positions of agents within EIs.

Thus, we can write rules for prohibiting (or permitting) actions by default, for permitting (or prohibiting) certain actions and for choosing which actions will be prevented or only sanctioned. For example, in the context of a Dutch auction [9], agents can be prevented from making unsupported bids or only sanctioned:

$$\left(\alpha_0 \wedge \neg\,(\alpha_1 \wedge T2 > T) \wedge oav(Ag, credit, C) \wedge C < P \right) \rightsquigarrow \oplus\alpha_2$$

where

$$\begin{cases} \alpha_0 = utt(dutch, w_3, inform(Au, auct, A, buyer, offer(It, P), T)) \\ \alpha_1 = utt(dutch, w_3, inform(Au, auct, A, buyer, offer(It, P), T_2)) \\ \alpha_2 = prh(dutch, w_4, inform(A, buyer, Au, auct, bid(It, P_2), T_3)) \end{cases}$$

The rule above prevents agents from issuing bids in a Dutch auction they cannot afford (their credit is insufficient). It states that if agent Ag's credit is less than P (the last offer the auctioneer called for item It, at state w_3 of scene $dutch$), then agent Ag is prohibited to bid.

$$\left(X = \{\ \alpha_0 \mid \alpha_1 \wedge \neg\,(\alpha_2 \wedge T_2 > T_1) \wedge T_0 > T_1\ \} \wedge \begin{array}{c} \\ |\,X\,| = 1 \wedge oav(A_1, credit, C) \wedge C < P \end{array} \right) \rightsquigarrow$$

$$\left(\begin{array}{c} \ominus oav(A_1, credit, C) \wedge \\ \oplus oav(A_1, credit, C_2) \wedge \oplus\alpha_3 \wedge \\ \oplus(C_2 = C - P * 0.1) \wedge \oplus(P_2 = P * 1.2) \end{array} \right)$$

where

$$\begin{cases} \alpha_0 = utt(dutch, w_4, inform(A_1, buyer, Au, auct, bid(It, P), T_0)) \\ \alpha_1 = utt(dutch, w_3, inform(Au, auct, all, buyer, offer(It, P), T_1)), \\ \alpha_2 = utt(dutch, w_3, inform(Au, auct, all, buyer, offer(It, P), T_2)) \\ \alpha_3 = obl(dutch, w_5, inform(Au, auct, all, buyer, offer(It, P_2), T_3)) \end{cases}$$

The previous rule punishes agents when issuing a winning bid they cannot pay for. More precisely, the rule punishes agent A_1 by decreasing his credit a 10% of the value of the good being auctioned. The oav predicate on the *LHS* of the rule represents the current credit of the offending agent. The rule also adds an obligation for the auctioneer to restart the bidding round along with the constraint that the new offer should be greater than 120% the old price.

These examples illustrate how our language addresses the explicit management of normative positions and the pragmatic concerns raised in the desiderata.

[5] We allow agents to utter whatever they want (via att formulae). However, the illegal utterances may be discarded and/or may cause sanctions, depending on the semantics of the specified deontic notions. The utt formulae are thus *confirmations* of the att formulae that turn illocutions into legal.

3. CONCLUSIONS AND FUTURE WORK

In this paper we have introduced a formalism for the explicit management of the normative position of agents in electronic institutions. Ours is a rule language in which constraints can be specified and changed at run-time, conferring expressiveness and precision on our constructs. The semantics of our formalism defines a kind of production system in which rules are exhaustively applied to a state of affairs, leading to the next state of affairs. The normative positions are updated via rules, depending on the messages agents send. Our formalism addresses the points of a desiderata for normative languages introduced above.

We would like to generalise our language to cope with arbitrary actions, rather than just speech acts among agents – this would allow our work to address any type of open multi-agent system. We would also like to improve the semantics of the language in order to support the use of temporal operators for the management of time along the lines of [6].

4. REFERENCES

[1] A. Artikis, L. Kamara, J. Pitt, and M. Sergot. A Protocol for Resource Sharing in Norm-Governed Ad Hoc Networks. volume 3476 of *LNCS*. Springer-Verlag, 2005.

[2] R. Axelrod. *The complexity of cooperation: agent-based models of competition and collaboration*. Princeton studies in complexity. Princeton University, New Jersey, 1997.

[3] F. Dignum. Autonomous Agents with Norms. *Artificial Intelligence and Law*, 7(1):69–79, 1999.

[4] M. Esteva. *Electronic Institutions: from Specification to Development*. PhD thesis, Universitat Politècnica de Catalunya (UPC), 2003. IIIA monography Vol. 19.

[5] M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, New York, U.S.A., 1990.

[6] A. García-Camino, P. Noriega, and J. A. Rodríguez-Aguilar. Implementing Norms in Electronic Institutions. In *Procs. 4th AAMAS*, 2005.

[7] A. García-Camino, J. A. Rodríguez-Aguilar, C. Sierra, and W. Vasconcelos. A Distributed Architecture for Norm-Aware Agent Societies. AUCS/TR0503, Dept of Computing Sc., Univ. of Aberdeen, Aberdeen, UK, 2005.

[8] F. López y López. *Social Power and Norms: Impact on agent behaviour*. PhD thesis, Univ. of Southampton, June 2003.

[9] P. Noriega. *Agent-Mediated Auctions: The Fishmarket Metaphor*. PhD thesis, Universitat Autònoma de Barcelona (UAB), 1997. IIIA monography Vol. 8.

[10] J. Searle. *Speech Acts, An Essay in the Philosophy of Language*. Cambridge University Press, 1969.

[11] M. Sergot. A Computational Theory of Normative Positions. *ACM Trans. Comput. Logic*, 2(4):581–622, 2001.

[12] Y. Shoham and M. Tennenholtz. On Social Laws for Artificial Agent Societies: Off-line Design. *Artificial Intelligence*, 73(1-2):231–252, 1995.

[13] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons, Chichester, UK, Feb. 2002.

Learning to Identify Winning Coalitions in the PAC Model

Ariel D. Procaccia Jeffrey S. Rosenschein
School of Engineering and Computer Science
The Hebrew University of Jerusalem
Jerusalem, Israel
{arielpro, jeff}@cs.huji.ac.il

ABSTRACT

We consider PAC learning of *simple* cooperative games, in which the coalitions are partitioned into "winning" and "losing" coalitions. We analyze the complexity of learning a suitable concept class via its Vapnik-Chervonenkis (VC) dimension, and provide an algorithm that learns this class. Furthermore, we study constrained simple games; we demonstrate that the VC dimension can be dramatically reduced when one allows only a single minimum winning coalition (even more so when this coalition has cardinality 1), whereas other interesting constraints do not significantly lower the dimension.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity;
I.2.6 [**Artificial Intelligence**]: Learning—*Concept Learning*;
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Algorithms, Theory

Keywords

PAC Learning, Coalition Formation

1. INTRODUCTION

A significant portion of recent research in multiagent systems has focused on learning. Nevertheless, very few investigations have been devoted to learning in coalition formation, an area of game theory that is exceedingly relevant to multiagent systems. Classical models of coalition formation assume that the values of all coalitions are known *a priori*, but this assumption is unreasonable in many (indeed, almost all) multiagent settings. Surprisingly, even fewer researchers have attempted to apply PAC[1] (Probably Approximately Correct) learning theory to multiagent settings, although this model has been extensively studied by researchers in computational learning theory (readers who are not familiar with this model are urged to consult [4]).

In this paper, we endeavor to remedy these shortcomings by studying PAC learning of *simple* cooperative games, in which each coalition is either "winning" or "losing". A basic structure is induced by the assumption that if a coalition is winning, any coalition that contains it is also winning. Simple games are a suitable model for numerous n-person conflict situations.

The Vapnik-Chervonenkis (VC) dimension is a combinatorial measure of the "richness" of a concept class[2], and is proportional to the difficulty of learning the class. We define several concept classes, which characterize settings of learning simple games with different restrictions, and identify the complexity of learning these classes by calculating their VC dimensions.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. THE MODEL

A *cooperative n-person game in characteristic form with side payments* is a pair $(N; v)$, where $N = \{1, 2, \ldots, n\}$ is a set of players, and v is the *characteristic function*, which assigns a real number $v(C)$ to each *coalition* $C \subseteq N$. $v(C)$ is the *value* of C.

Simple games are games where each coalition has a value of either 1 or 0. A coalition C is said to be *winning* if $v(C) = 1$, and *losing* if $v(C) = 0$. 2^N, the powerset of N, is partitioned into $\mathscr{W}$, the set of winning coalitions, and $\mathscr{L}$, the set of losing coalitions. A standard assumption is that this partition satisfies:

$$[C_1 \in \mathscr{W} \wedge C_1 \subseteq C_2] \Rightarrow C_2 \in \mathscr{W}. \quad (1)$$

We wish to find a way to identify winning coalitions. In other words, given a coalition, we would like to label it by "winning" or "losing", or equivalently, by 1 or 0, respectively. Essentially, we would like to learn a function from coalitions to $\{0, 1\}$.

The assumption given in Equation (1) allows us to easily represent such functions. Indeed, we say that $C \subseteq N$ is a minimum winning coalition if and only if

$$C \in \mathscr{W} \wedge \forall i \in C,\ (C \setminus \{i\}) \in \mathscr{L}.$$

[1]The PAC model is also known as the *formal* model.
[2]A concept class is a binary function whose domain is the set of possible samples.

A simple game can be concisely represented by a set of minimum winning coalitions. Furthermore, a set of minimum winning coalitions may be regarded as a function from the set of coalitions to $\{0,1\}$: a coalition is winning (labeled by 1) if and only if it is a superset (in the weak sense) of one of the minimum winning coalitions.

REMARK 1. Surprisingly, learning to identify minimum winning coalitions (with the above formalization in mind) is equivalent to learning monotone[3] DNF formulas. Indeed, one can associate players with variables, and coalitions with terms. A hypothesis consisting of a set of minimum winning coalitions is essentially a disjunction of terms. When learning minimum winning coalitions, the sample space is the space of all coalitions; these can be identified with assignments to variables, where a coalition C induces an assignment of 1 to the variables associated with the players in C, and 0 to all other players. This equivalence allows us to leverage some of the research on monotone DNF in order to prove certain results.

For the sake of completeness, we give a swift description of the VC dimension; more details can be found in [4].

DEFINITION 1. Let $\mathcal{C} : X \to \{0,1\}$ be a concept class, $S = \{x_1, x_2, \ldots, x_m\} \subseteq X$, and let

$$\Pi_{\mathcal{C}}(S) = \{\langle h(x_1), h(x_2), \ldots, h(x_m)\rangle : h \in \mathcal{C}\}.$$

If $|\Pi_{\mathcal{C}}(S)| = 2^m$, then S is considered *shattered* by $\mathcal{C}$.

In other words, if S is shattered by $\mathcal{C}$, then $\mathcal{C}$ realizes all possible dichotomies on S.

DEFINITION 2. The *Vapnik-Chervonenkis (VC) dimension* of a concept class $\mathcal{C}$, denoted VC-dim($\mathcal{C}$), is the size of the largest set S that is shattered by $\mathcal{C}$. If $\mathcal{C}$ shatters arbitrarily large sets, then VC-dim($\mathcal{C}$) $= \infty$.

It is known that the VC dimension of a concept class almost completely characterizes the complexity of learning it in the PAC model. Therefore, we focus on calculating the VC dimension of several concept classes of interest.

3. LEARNING UNCONSTRAINED GAMES

In order to calculate the VC dimension of the concept class of minimum winning coalitions, when no restrictions are imposed on the game, we first need to address a natural combinatorial problem.

Given the numbers $\{1, 2, \ldots, n\}$, we would like to find a maximal antichain of subsets, i.e., a family of subsets such that for any two subsets, neither one is contained in the other. Finding an antichain of size $\binom{n}{\lfloor n/2 \rfloor}$ is easy: we simply choose all the subsets of size $\lfloor n/2 \rfloor$ (see Table 1). But can one do better? The theorem gives a negative answer.

THEOREM 1 (SPERNER'S THEOREM). *Let $\mathcal{F}$ be a family of subsets of $\{1, 2, \ldots, n\}$, such that for all $A, B \in \mathcal{F}$: $A \not\subseteq B$. Then $|\mathcal{F}| \leq \binom{n}{\lfloor n/2 \rfloor}$.*

Now we can straightforwardly prove the main theorem.

[3]DNF formulas in which all the literals are positive.

		$\emptyset$			
	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	
{1,2}	**{1,3}**	**{1,4}**	**{2,3}**	**{2,4}**	**{3,4}**
	$\{1,2,3\}$	$\{1,2,4\}$	$\{1,3,4\}$	$\{2,3,4\}$	
		$\{1,2,3,4\}$			

Table 1: Subsets of {1,2,3,4}, sorted by size. A maximal antichain is constructed by choosing all subsets of size 2; Sperner's Theorem states that one cannot construct a larger antichain.

THEOREM 2. *Let $\mathcal{C}^*$ be a concept class, in which each concept is a set of minimum winning coalitions. Then:*

$$\textit{VC-dim}(\mathcal{C}^*) = \binom{n}{\lfloor n/2 \rfloor}.$$

PROOF. We first show that the VC-dimension is at least $\binom{n}{\lfloor n/2 \rfloor}$, by producing a set of size $\binom{n}{\lfloor n/2 \rfloor}$ which is shattered by $\mathcal{C}^*$. Consider the set S of all coalitions of size $\lfloor n/2 \rfloor$. Clearly, any dichotomy on S can be realized by the concept that contains as minimum winning coalitions exactly the coalitions in S that are labeled by 1.

On the other hand, the VC-dimension is at most $\binom{n}{\lfloor n/2 \rfloor}$. Indeed, consider a set S of more than $\binom{n}{\lfloor n/2 \rfloor}$ coalitions. By Theorem 1, there are two coalitions $C_1, C_2 \in S$ such that $C_1 \subseteq C_2$. The dichotomy in which C_1 is labeled by 1 and C_2 is labeled by 0 cannot be realized. □

CONTRACT, given as Algorithm 1, is a learning algorithm for $\mathcal{C}^*$: it returns a set of minimum winning coalitions which is consistent with the given samples (assuming one exists).

Algorithm 1

```
1: procedure CONTRACT({(C_i, y_i)}_{i=1}^m)    ▷ y_i ∈ {0,1} is the label of coalition C_i
2:     W^m ← ∅                                  ▷ Set of minimum winning coalitions
3:     for i = 1 to m do                        ▷ All sample coalitions
4:         if y_i = 1 ∧ ∀R ∈ W^m, R ⊈ C_i then
5:             W^m ← W^m \ {R ⊆ N : C_i ⊆ R}
6:             W^m ← W^m ∪ {C_i}
7:         end if
8:     end for
9:     return W^m
10: end procedure
```

4. LEARNING CONSTRAINED GAMES

Veto games are cooperative games where any coalition with nonzero value contains a distinguished player, called the *veto* player. In some veto games, the inclusion of the veto player in a coalition is also a sufficient condition for the coalition to be winning, not just a necessary one; in this case, we say the veto player is a *dictator*. Some real-world n-person conflict situations are simple games with a dictator.

That being the case, in order to identify winning coalitions in a simple game with a dictator, it is enough to pinpoint this distinguished player. The set of minimum coalitions contains exactly one coalition of cardinality 1: the dictator.

The following proposition follows from [1].

PROPOSITION 3. *Let $\mathcal{C}_1$ be a concept class, in which each concept is a single minimum winning coalition of cardinality 1. Then:*

$$VC\text{-}dim(\mathcal{C}_1) = \lfloor logn \rfloor.$$

It is worthwhile to generalize our requirement of a single dictator player; we now assume there is a coalition W such that for all coalitions C, C is winning if and only if $W \subseteq C$. Since W is the only minimum winning coalition, the goal of the learning process in such games is to recognize W.

By the equivalence between minimum winning coalitions and monotone DNF, it is possible to derive the following proposition from [2].

PROPOSITION 4. *Let $\mathcal{C}_1^*$ be a concept class, in which each concept is a single minimum winning coalition. Then:*

$$VC\text{-}dim(\mathcal{C}_1^*) = n.$$

REMARK 2. Another possible generalization is having at most k minimum winning coalitions; denote this concept class by $\mathcal{C}_k^*$. An upper bound for this case is given by Schmitt [3]. In particular, if $k = O(1)$, then VC-dim$(\mathcal{C}_k^*) = O(n)$.

Proper simple games are another type of common constrained games. In such games it holds that

$$\forall S \subseteq N,\ S \in \mathscr{W} \Rightarrow N \setminus S \in \mathscr{L}. \quad (2)$$

However, this constraint does little to reduce the VC dimension, compared with unconstrained simple games.

PROPOSITION 5. *Let $\mathcal{C}_p^*$ be a concept class, in which each concept is a set of minimum winning coalitions in a proper simple game. Then:*

$$VC\text{-}dim(\mathcal{C}_p^*) \geq \binom{n-1}{\lfloor \frac{n-1}{2} \rfloor}.$$

PROOF. We must exhibit a set S of size $\binom{n-1}{\lfloor \frac{n-1}{2} \rfloor}$ which is shattered by $\mathcal{C}_p^*$. Let

$$S = \left\{ C \subseteq N :\ 1 \in C \wedge |C| = \left\lfloor \frac{n-1}{2} \right\rfloor + 1 \right\}.$$

It holds that the cardinality of S is as desired. Moreover, for any dichotomy on S, one may choose exactly the subsets labeled by 1 as the minimum winning coalitions: since the intersection of all subsets in S is not empty, the constraint (2) is not violated. □

Another popular constraint is the elimination of *dummy* players.

$$\forall i \in N \exists C \subseteq N \ s.t.\ i \in C \wedge C \in \mathscr{W} \wedge (C \setminus \{i\}) \in \mathscr{L}. \quad (3)$$

Informally, for every player there is a winning coalition which cannot do without it. The elimination of dummy players also does not substantially reduce the VC dimension.

PROPOSITION 6. *Let $\mathcal{C}_d^*$ be a concept class, in which each concept is a set of minimum winning coalitions in a simple game with no dummy players. Then:*

$$VC\text{-}dim(\mathcal{C}_d^*) \geq \binom{n-1}{\lfloor \frac{n-1}{2} \rfloor}.$$

PROOF. We must exhibit a set S of size $\binom{n-1}{\lfloor \frac{n-1}{2} \rfloor}$ which is shattered by $\mathcal{C}_d^*$. Let

$$S = \left\{ C \subseteq N :\ 1 \notin C \wedge |C| = \left\lfloor \frac{n-1}{2} \right\rfloor \right\}.$$

It holds that the cardinality of S is as desired. Given a dichotomy on S, let S^+ be the set of coalitions in S that are labeled by 1, and let B be the set of players that are not members of any of the coalitions in S^+. The set of minimum winning coalitions is $S^+ \cup \{B\}$; the purpose of including B is to ensure that constraint (3) is not violated. It remains to show that this is a legitimate set of minimum coalitions in a proper simple game, which realizes the dichotomy. Since $1 \in B$, B is not a subset of any of the coalitions in S, and in particular is not a subset of any of the coalitions in $S \setminus S^+$ — so it is not the case that the addition of B to the set of minimum winning coalitions mistakenly labels a coalition in $S \setminus S^+$ by 1. Clearly, neither is B a superset of any of the coalitions in S^+; thus it is not the case that there are two winning coalitions such that one is a superset of the other. Since the coalitions in S^+ are included in the set of minimum winning coalitions, the dichotomy is realized. Moreover, constraint (3) is satisfied: every $i \in N$ is contained in one of the coalitions in S^+, or in B, and these are all *minimum* winning coalitions. □

5. CONCLUSIONS

Simple games have natural interpretations in multiagent systems. Such games can be concisely represented by the set of minimum winning coalitions; this set may be regarded as a function from the set of coalitions to $\{0, 1\}$.

We have shown that the VC-dimension of the concept class which contains sets of minimum winning coalitions is $\binom{n}{\lfloor n/2 \rfloor}$; this implies that in general, it is very difficult to learn to identify winning coalitions in the PAC model.

We have also discussed constrained simple games, and have proven that restricting the set of minimum winning coalitions to a single coalition, or a dictator player, greatly reduces the VC-dimension. These constrained games are thus efficiently learnable. Nevertheless, other popular constrained games, such as proper simple games and games with no dummy players, are almost as hard to learn as unconstrained games.

6. ACKNOWLEDGMENT

This work was partially supported by grant #039-7582 from the Israel Science Foundation.

7. REFERENCES

[1] N. Littlestone. Learning quickly when irrelevant attributes abound. *Machine Learning*, 2(4):285–318, 1988.

[2] T. Natschlager and M. Schmitt. Exact VC-dimension of boolean monomials. *Information Processing Letters*, 59:19–20, 1996.

[3] M. Schmitt. On the capabilities of higher-order neurons: A radial basis function approach. *Neural Computation*, 17(3):715–729, 2005.

[4] M. Vidyasagar. *A Theory of Learning and Generalization*. Spinger-Verlag, 1997.

A Deployed Multi-Agent Framework for Distributed Energy Applications

Geoff James[1], Dave Cohen[2], Robert Dodier[2], Glenn Platt[3], and Doug Palmer[1]

[1]CSIRO ICT Centre
Locked Bag 17
North Ryde NSW 1670
Australia
+61 2 9325 3276
{first.last}@csiro.au

[2]Infotility Inc
1327 Spruce St. Suite 201
Boulder CO 80301
USA
+1 720 210 1984
{first}@infotility.com

[3]CSIRO Energy Technology
PO Box 330
Newcastle NSW 2300
Australia
+61 2 4960 6120
{first.last}@csiro.au

ABSTRACT
In this paper, we describe the adaptation of an open-source multi-agent platform for distributed energy applications and the trial deployment of resource-controller agents. The platform provides real-time, two-way communication and decision making between distributed energy resources – loads and generators – in electricity distribution networks. Such a decentralized architecture improves grid reliability, allows consumers to play a more active role in their energy usage, benefits the network by alleviating the effects of peak wholesale prices and network constraints, and creates new business opportunities in a deregulated market. Agents have been deployed to control hardware at trial locations in Australia, providing a realistic test environment for the platform, and medium-scale trials are anticipated in the near future.

Categories and Subject Descriptors
J.2 [**Physical Sciences and Engineering**]: Engineering.

General Terms
Algorithms, Management, Design, Experimentation.

Keywords
Applications of autonomous agents and multi-agent systems; collective and emergent agent behavior; frameworks, infrastructures and environments for agent systems.

1. INTRODUCTION
The GridAgents Framework is being developed by Infotility, under contract with the U.S. Department of Energy, and CSIRO. It consists of an adaptive, intelligent agent-based software system which can provide real-time, two-way communication and decision making between distributed DER system nodes. Distributed Energy Resources (DER) have the potential to provide significant benefits to the electric grid infrastructure and to society as a whole. DERs are typically smaller-scale energy and storage technologies sited close to electrical and thermal loads. DER technologies are increasingly being considered to augment the existing power delivery infrastructure as part of grid reliability programs (emergency response, price response, ancillary services, etc.). They also offer a solution to electricity market participants exposed to peak wholesale prices and constrained network infrastructure. DER has the potential to provide reactive power (generated close to loads) which is highly beneficial for maintaining grid voltage stability. DER combined with next-generation sensing, communication, and control technologies will enable a "path change" by providing the ability to intelligently respond to grid reliability and market signals in a cost-effective manner. The traditional site-specific benefits of DER (efficiency, environmental, local reliability, etc.) can be enhanced by the use of next-generation distributed computing technologies such as those used in this work. Another important aspect of the proposed agent-based software is that it enables the ability to create on-the-fly "aggregate" blocks of capacity for presentation to the energy markets. This functionality along with intelligent islanding and other control concepts has been proposed as various forms of the "microgrid" concept by several research organizations including the DOE. The design is based on a distributed, bottom-up approach to responding to grid contingencies and is intended to supplement the existing power distribution network communication infrastructure.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

2. SELECTING A PLATFORM
Agent-based software applications require a foundation to manage basic operations. Various agent toolkits, platforms, toolkits, and methodologies, both proprietary and open-source, were reviewed in order to select a platform to build upon for the GridAgents Framework under development. An agent platform acts as a middleware layer, providing services such as lifecycle management, resource management, messaging, security, directory services, and behaviors. By using such a platform, a programmer is not required to re-invent (and debug) the wheel for each agent project.

The available platforms offer a wide range of programming models and assumptions about locality, communications and agent sophistication. The GridAgents Framework is expected to operate over a large range of hardware platforms and sites. In particular, the CSIRO trial deployments use personal digital assistants (PDAs) as computing resources attached to various loads and generators. The use of PDAs requires an agent platform capable of operating on limited devices, a requirement that quickly eliminates most platforms from consideration. Three agent platforms capable of being deployed on PDAs were evaluated: Cougaar [5], Cybele [6], and JADE [7]. In addition, the general-purpose .NET platform [8] was also evaluated as a sanity-check, to ensure that using an agent platform really did result in a reduction in effort.

A set of detailed criteria were developed in order to choose the agent platform as part of the requirements assessment process. The evaluation methodology used was an informal version of the i-Mate process [3]. Although .NET and Cybele were quickly eliminated from consideration, Cougaar and JADE provided similar facilities, different implementation models and contrasting strengths and weaknesses. Eventually, JADE was chosen on the grounds of a cleaner design and implementation, better documentation and the ease with which agents could be deployed on limited devices. JADE is by no means perfect; a considerable amount of effort has since gone into architectural issues, such as publish/subscribe notification, agent discovery and deployment. However, JADE has stood up as the "least-worst" choice.

3. ARCHITECTURE AND MESSAGING

DER agents appear to each other to fulfill certain functions which we have termed "roles". In this system, a role is the ability to send and receive a certain fixed set of message types. The means by which an agent determines its message content is invisible to other agents, and other agents know only the allowed message types. The roles include buyer, seller, bulletin board, physical device, and others. An agent which fulfills the seller role will respond to a request for a price function or a request to conclude a transaction. An agent which fulfills the physical device role will respond to requests for status information or a request to set the device state. As other agents see only these functions, the DER agent system is a "plug and play" system, in which an agent enjoys broad latitude within which it can implement the role functions in various ways, so long as all functions for a given role are implemented.

An agent may fulfill more than one role. As agents interact only through the role interfaces, a new type of agent can coexist happily with others so long as the role interfaces are implemented appropriately. There is no need to restart the whole agent system when a new agent type comes on-line. This is an especially important consideration in systems comprising multiple hosts, as it may be impractical to coordinate the operations of different hosts. It is feasible, though perhaps somewhat involved, to create agents which are not JADE agents, by constructing FIPA messages of appropriate types and contents. At this time we have no plan to implement non-JADE agents, but we see no need to rule out the possibility.

Data persistence is accomplished in the DER agent system through the bulletin board role. A bulletin board is just a repository of messages organized according to a label, which is called the topic. Agents can query the bulletin board by topic, receiving the messages under that topic in response. Aside from messages exchanged through role interfaces, public information in the DER agent system is published via a bulletin board. The types of information available may include prices, parameters, notices, and measurements. It is anticipated that the contents of a bulletin board may be stored in a conventional database, or published in a format accessible outside the DER agent system, such as an XML-based syndication format.

4. APPLICATIONS AND ALGORITHMS

Whether the application is grid security or services for electricity market participants, the fundamental algorithmic requirement of a DER multi-agent network is the provision of aggregated capacity through load shifting, load shedding, or distributed generation by a multiplicity of resources. Aggregated capacity is characterized by a quantity of kW or MW and also time and timeliness, reliability or "firmness", and traceability to a known set of customers. Note that reliability may be represented as a profile rather than a single value: 20 MW at 99.99% firmness, 40 MW at 99% firmness, and 70 MW at 90% firmness would be an example. This is enough information for a user to make an informed risk calculation. At the resource agent level the ability to shift or curtail load or dispatch local generation requires planning ahead, with a planning horizon long enough to account for time constants inherent in the loads and generators.

Each agent in the DER agent system attempts to choose among available alternatives according to the criterion of minimum operating cost and subject to constraints. This local optimization may lead to a solution with identifiable global properties, such as a Nash equilibrium, although we have not proved any such properties yet. We assume that each agent has a planning and execution cycle, which may be regular or irregular, and coordinated or uncoordinated with other agents. The result of the planning phase is a series of actions to be carried out in the future. In the execution phase, the agent carries out the plan by executing commands to a physical device, or perhaps issuing commands to subordinate agents which result in a command to a device.

The planning problem is generically phrased as a constrained optimization problem. The appropriate method for solving such problems varies according to the form of the cost function, and thus agents may employ various techniques for planning. Agents which aggregate or broker the output of other agents may employ integer linear programming to solve their planning problems, as their costs are the sum of the prices stated by their subordinate agents and their resources are the sum of the resources supplied by their subordinates. Aggregating agents may also employ genetic optimization [1] to optimize subsidiary agents for soft-real-time application.

As an exact solution to an optimization problem may be very time-consuming, we are also developing methods which provide "any-time" solutions based on selfish agents seeking to optimize local cost functions. Although the resulting global solution may be suboptimal, this may be a worthwhile sacrifice it the net result of the selfish agents' actions – their emergent behavior – converges to a useful solution in a short time.

Planning algorithms present a scalability problem: there are many free parameters for each resource, combinatorial optimisation of any kind tests computational resources quite severely, and our aim

is to coordinate many thousands of agents. Decentralised algorithms are therefore of great interest because they use the multi-agent network as a parallel computer and thereby promise better scaling properties provided that messaging overheads do not defeat this purpose. A decentralised algorithm does not yet exist for planning as we've defined it, although a candidate with many of the capabilities we require is the market-based algorithm of Kok et al. [2], and other algorithms under development will be reported in the near future. All candidate planning algorithms presume a defined group of agents that should plan together and a defined connectivity between agents in the group.

The approach we are taking is to construct a meta-algorithm that governs the groups of agents within which planning is used. Decentralised clustering is an example of a multi-agent algorithm that creates groups, and it has been shown to scale better than an equivalent centralised algorithm [4]. We are developing an adaptation of decentralised clustering to govern the formation of planning groups of energy agents and the topology of messaging connections within each group. The crucial requirement is to interleave the two levels of algorithm so that time scales are compatible and sufficiently short for the system responses desired.

5. DEPLOYMENT

To test the behavior of various multi-agent system approaches and showcase our technology, we have established a comprehensive experimentation and testing hardware platform. The platform is made up of a variety of loads and generators, configured for agent control, and subject to real-world constraints:

- Two cool rooms typical of a small business setting.
- An heating, ventilation and air-conditioning (HVAC) system in an office building.
- A 30 kW gas turbine or "microturbine".
- A large photovoltaic (PV) system.
- A small wind turbine and solar-tracking PV array, 1 kW each, combined with battery storage.

The interacting set of loads and generators are electrically connected to form a microgrid, their controlling agents being designed to coordinate demand and supply while reacting intelligently to national electricity market price signals. Agents controlling loads and generators are run on personal digital assistants (PDAs) while personal computers (PCs) are used for implementing more computationally intensive agent roles:

- A web scavenging agent which obtains current and forecast electricity market prices and weather data.
- A snapshot agent which monitors various real-time parameters of the agent system.
- An aggregation agent which offers a price schedule for resources from subordinate agents.
- A manager agent which directs the operation of a number of loads and sources.

The hardware platform has a comprehensive Internet Protocol (IP) based infrastructure for inter-agent communications. This infrastructure uses a variety of physical layer technologies, including Ethernet and Wireless LAN mechanisms, with Bluetooth and cable based connections used for serial communications to distributed energy devices.

Throughout this implementation work, and the subsequent experimental phase, a number of challenges and their solutions have been encountered. We believe many of these challenges would exist in *any* geographically distributed multi-agent system implemented in common computing and communications hardware. Space does not allow detail here, but the authors are happy to be contacted about such issues as agent communication, firewalls, network addressing, processing overheads, unique and diverse hardware installations, reliable and fail-safe behavior, and message content encoding.

6. CONCLUSIONS

We have developed the GridAgents Framework for managing distributed energy resources – customer loads and generators – by adapting the JADE multi-agent platform. Several aspects peculiar to our application are described in this paper, many of them based on experience of deployment on distributed processors controlling typical small-business loads and generators. We have developed algorithms to aggregate capacity and provide a distributed energy service to meet network and market goals. This work continues.

Future work will include advanced demonstrations of the GridAgents Framework for Microgrid applications, automated demand response applications, and electric distribution network control applications. A "smart grid" project is planned for 2006 with a large U.S. utility for embedded intelligence and control for compact networks. In Australia our next goal, in collaboration with utility, hardware, and telecommunications partners, is a moderate-scale deployment of the order of 1,000 agents.

7. ACKNOWLEDGMENTS

Everyone in the development and deployment team made an important contribution to the results reported here. We gratefully acknowledge the work of Matt Blyde, Matthew Chadwick, Yan Kitsis, Marteen Rojo, Peter Wang, and Sam West.

8. REFERENCES

[1] Guo, Y., Li, J., and James, G. Evolutionary Optimisation of Distributed Energy Resources. *Proceedings, 18th Australian joint conference on Artificial Intelligence (AI'05)*, Sydney, December 2005.

[2] Kok, J,K., Warmer, C.J., and Kamphuis, I.G. PowerMatcher: Multiagent control in the electricity infrastructure. *Proceedings, 4th international joint conference on Autonomous Agents and Multi Agent Systems (AAMAS'05)*, Utrecht, July 2005.

[3] Liu, A. and Gorton, I. Accelerating COTS Middleware Acquisition: The i-Mate Process. *IEEE Software, 20*, 2 (2003), 72-79.

[4] Ogston, E., Overeinder, B., van Steen, M., and Brazier, F. A Method for Decentralized Clustering in Large Multi-Agent Systems. *Proceedings, 2nd international joint conference on Autonomous Agents and Multi Agent Systems (AAMAS'03)*, Melbourne, July 2003.

[5] http://www.cougaar.org/

[6] http://www.i-a-i.com

[7] http://jade.tilab.com/

[8] http://www.microsoft.com/net

Behaviosites: A Novel Paradigm for Affecting Distributed Behavior

Amit Shabtay
ashabtay@gmail.com

Zinovi Rabinovich
nomad@cs.huji.ac.il

Jeffrey S. Rosenschein
jeff@cs.huji.ac.il

School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel

ABSTRACT

We present the Behaviosite paradigm, a new approach to affecting the behavior of distributed agents in a multiagent system, which is inspired by biological parasites with behavior manipulation properties. Behaviosites are special kinds of agents that "infect" a system composed of agents operating in that environment. The behaviosites facilitate behavioral changes in agents to achieve altered, potentially improved, performance of the overall system. Behaviosites need to be designed so that they are intimately familiar with the internal workings of the environment and of the agents operating within it, and behaviosites apply this knowledge for their manipulation, using various infection and manipulation strategies. To demonstrate and test this paradigm, we implemented a version of the El Farol problem, using behaviosites.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed AI—*Intelligent Agents, Multiagent Systems*

1. INTRODUCTION

Parasites have been examined in biomimetics, but one of the most interesting abilities of biological parasites has received little attention in technological contexts — their ability to manipulate and alter their host's behavior. In rare cases, biological parasites can actually benefit the host or the host's society.

In this paper, we present a novel paradigm that employs a special kind of agent (called a *behaviosite*) that manipulates the behavior of other agents so as to achieve altered, possibly improved, performance of the entire system. The behaviosite (by definition) is not itself necessary for the normal conduct of the system; thus, it is termed a kind of "parasite".

Within the field of MultiAgent Systems (MAS), the behaviosite is closely related to the idea of adjustable autonomy (AA) [6]. In the context of the Behaviosite paradigm, an agent usually transfers some of its autonomy to the behaviosite unwittingly, and in any case the agent is better described as *ceding autonomy*.

Introducing behaviosites to a system may be either planned as part of the overall system design, or added (after the fact) to an already working system. When planned as part of the overall system, it is possible to choose between applying *internal behaviosites* (design hooks inside agents, so that behaviosites can attach to the agent and manipulate them) or *external behaviosites* (which manipulate only the input/output of the agent, vis-à-vis the environment). When applied to an already working system, external behaviosites are usually the only option.

Section 2 presents a formalization of the Behaviosite paradigm. Section 3 discusses the El Farol problem, and how behaviosites could be employed in its solution. Section 4 concludes with a discussion of our approach and of future work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. THE BEHAVIOSITE MODEL

2.1 Overall Structure

The Behaviosite paradigm is composed of three parts: the environment, agents (also referred to as hosts), and behaviosites. Environment, agents, and behaviosites will be referred to collectively as "the system" (as in ecosystem).

Environment: encapsulates all of the external factors, conditions, and influences that affect a community of agents and behaviosites.

Agent society: A society of agents is a system composed of multiple, interacting, possibly cooperating agents.

Behaviosite: Most basically, a specialized type of agent. A behaviosite is an additional property/information added to a system with a society of agents, and is not (and should not be) a property of the agent or the environment. The behaviosite is not required for the system, but should be beneficial in some sense; too many behaviosites can degrade some aspects of the system.

2.2 Required Traits

Benefiting the system: Behaviosites must prove beneficial to the system in some respect, since they come with costs. Behaviosites may influence a system in two desired ways: they may increase social utility, or they may create new features in a working system. Both are accomplished by altering the behavior of some of the agents.

System knowledge: A behaviosite must be designed with deep understanding of how the system works: agent-agent interactions, agent-environment interactions, and also internal workings of the agents in some cases. One type of system in which behaviosites can be beneficial is a system that works in a suboptimal equilibrium. The internet is one example of a system in suboptimal equilibrium, with regard to the congestion problem caused by packet routing. The El Farol problem presents another, similar example.

Not a property of the agent nor of the environment: Behaviosites exist in the middle ground between agents and the environment, and should not be a property of either of them.

2.3 Optional Traits

Hidden or apparent infection: We usually will not want agents to know who is infected by the behaviosite, since such knowledge might help individual agents exploit the situation. However, there are some settings in which such knowledge can benefit the system (as also occurs in nature).

Finding the host: A behaviosite designer may endow the environment with the responsibility of infecting agents in the system using some strategy (like in the parasitized El Farol problem), or it may leave the task of infection to the behaviosites themselves, thus making the behaviosites more autonomous.

Behaviosite communication: Behaviosites may communicate with one another within a host (the host may be infected with more than one behaviosite), or across hosts. This may enable the behaviosites to be a catalyst for the creation of norms or social laws.

2.4 Placing the Behaviosite

To alter host behavior, the behaviosite should be placed somewhere along the flow of the system, as can be seen in Figure 1.

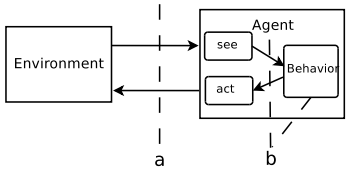


Figure 1: (a) marks where external behaviosite can alter host's behavior, by altering its input/output. (b) marks where internal behaviosite can alter host's behavior, by altering internal data or by partial or full replacement of behavior modules.

Behaviosites can be divided into two groups, external behaviosites, and internal behaviosites. External behaviosites can alter the input or output of the agent vis-à-vis the environment (Figure 1a). In this way, the host designer(s) need not know of the possible existence of a behaviosite now or in the future. Moreover, the behaviosite designer may introduce them into an existing system.[1] On the other hand, internal behaviosites usually require the system to be designed "plug and play" for the behaviosite (Figure 1b).

Internal behaviosites can have two main manipulation methods: changing the host's internal data (Figure 1b, interaction between see-behavior-act modules) or replacing some or all of its behavioral modules (Figure 1b, behavior module). The hook for the second manipulation method is very easy to program, by using the behavioral design patterns suggested by the "gang of four" [3].

2.5 Cost of Manipulation

In different scenarios, behaviosites have different types of costs. The first cost that should be considered is the cost to the system designer, for there is the immediate issue of designing behaviosites and testing them. Other costs of behaviosites are system specific, such as balancing behaviosite complexity and number with the benefit they give to the system, cost of possibly increased variation in the society, oscillations, run-time costs, and so forth.

3. PARASITIZED EL FAROL PROBLEM

3.1 The El Farol Problem

In the El Farol problem [1], N (e.g., 100) agents decide independently each week whether to go to the El Farol bar or not. Comfortable capacity is limited, and the evening is enjoyable only if the bar is not overcrowded (specifically, fewer than 60 agents out of the possible 100 attend). No communication or collusion is possible, and the only information available to the agents is the number of attendees in the past. An agent will go if it predicts less than 60 agents will go, and will stay home otherwise. Arthur's [1] utility function was:

$$Util(ag[i]) = \begin{cases} x & attended\ and\ undercrowded \\ 0 & did\ not\ attend \\ -y & attended\ and\ overcrowded \end{cases}$$

where $x = y = 0.5$. A set of personal deterministic strategies combined with a simple learning algorithm facilitated system convergence to the capacity after some initial learning time. Moreover, membership kept changing, and some degree of fairness was maintained. However, the system fluctuated chaotically around the capacity, resulting in low personal and social utility.

The learning algorithm used in the parasitized El Farol simulation was the additive updating learning algorithm, introduced in [5]. Basically, each agent has a pool of simple, personal deterministic strategies. Each such behavior has a weight, initially distributed uniformly, and updated each round.

3.2 Applying Behaviosites

The El Farol bar problem presents a distributed system with suboptimal social and personal utility. The main idea of the parasitized El Farol problem is to increase social utility with as few side effects as possible, using behaviosites. The main problem was that agents learn and adapt themselves to new situations — behaviosites' effects can vanish, or in each round behaviosites will need to make a stronger impact to achieve the same effect.

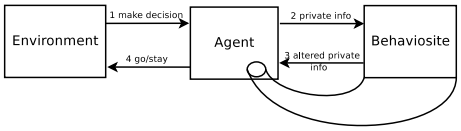


Figure 2: The environment/agent/behaviosite relationship

The system is composed of environment, agents (hosts) and behaviosites (Figure 2). Each agent has an internal behaviosite field that is always occupied, usually with the null behaviosite (see [3]) who has no effect over his host's behavior. Each round, the environment infects some of the agents according to an infection strategy, and infection lasts only one round. Agents are asked to provide a decision and if the behaviosite is not the null behaviosite, it alters the relevant host's data (the behaviosite is placed between "see" and "behavior" of Figure 1).

Environment infection strategies: The environment had three strategies of infection. In the first, all agents were candidates for infection (*infect all*). In the second, only the agents attending in the given round were candidates for infection (*infect attending*). In the third, all the agents attending in the given round were candidates for infection, but only when the bar was overcrowded (*infect overcrowded*). In each strategy, only a percentage of the candidates for infection actually got infected (responsibility of the behaviosite design), depending on the infection rate (0%-100%).[2]

Behaviosite manipulation strategies: The behaviosite replaced the parasitized agents' belief regarding the current capacity with a

[1]Of course, these external behaviosites affect inter-agent communication only if it, too, travels via the environment.

[2]Another possible design is that the entire infection strategy is a property of the behaviosite, but this was not implemented.

lower one, common to all behaviosites. Since capacity information was kept as the private history of the agent, this decrease was also considered in future rounds by the agent. This was intended to enforce a strict approach; agents were affected by the world according to their personal beliefs, not according to global truth.

3.3 Parasitized El Farol Simulation Results

The parasitized El Farol problem was simulated for 2000 rounds, with 100 agents in the system. 3 different capacities were tested: 50, 60, and 80, where the behaviosite manipulation strategy was to decrease capacity for parasitized agents (50 decreased to 40, 60 also decreased to 40, and 80 decreased to 60). For each capacity, 3 different environment infection strategies were tested, as mentioned above: *infect all*, *infect attending*, and *infect overcrowded*, giving a total of 9 different simulations. For each of the 9 simulations, a different percentage of the agents who were candidates for infection were actually infected, with infection rates ranging from 0% to 100% with jumps of 10% (11 different simulations for each). Each of these 99 different simulations was repeated 50 times.

3.3.1 Mean Attendance

In the parasitized El Farol problem, we would like to increase the social utility, without lowering by too much the mean attendance. As would be expected, the *infect all* strategy decreased the mean attendance severely, while the *infect overcrowded* strategy resulted in a relatively low effect on the mean attendance. The outcome of the *infect attending* strategy depended on the bar's capacity. For capacity of 80, the *infect all* and *infect attending* strategies yielded very similar results and for capacities of 50 and 60, *infect attending* strategy's effect resembles the *infect overcrowded* strategy.

3.3.2 Mean Utility

The main objective in the parasitized El Farol problem simulation was to show that it is possible to increase social utility using behaviosites. The *Infect all* strategy for capacities of 60 and 80 showed improvement of 7–9 times that which was achieved in a system without behaviosites, and was relatively close to the optimal. For capacity of 50, improvement was 27 times the minimal, though still near zero (0.065 out of possible 0.25).

Infect overcrowded resulted in a moderate utility increase, with respect to other strategies. For capacities of 60 and 80, the maximal mean utility was about 2.5–3.5 times that which was achieved in a system without behaviosites. For the capacity of 50, the maximal mean utility was 22 times better, but still near zero (0.034).

The *Infect attending* outcome depended heavily on the level of the used capacity. For high capacities (specifically 80), the strategy's behavior was very much like *infect all*, and for low capacities (specifically 50), the strategy's outcome was very much like *infect overcrowded*. In between (specifically 60), the behavior of this strategy combined features of both.

3.3.3 Forced Justice

Two previous attempts to increase social utility in the El Farol problem resulted in two proposed solutions, both with undesirable features in certain settings. One feature was the need to charge utility from attending agents [4], which basically kept the social utility within the system. Another led to the creation of bullies [2] — regulars and casuals. In the original El Farol problem, the membership of the agents kept changing, thus keeping some level of social fairness. Social fairness was calculated by the following formula:

$$1 - \frac{1}{\#trials} \sum_{t \in trials} \frac{Personal\ \ Attendance\ \ SD[t]}{Mean\ \ Attendance[t]}$$

For capacities of 50 and 60, *infect attending* and *infect overcrowded* strategies increase social fairness as a function of infection rate. For capacity of 80, they improved social fairness only a small amount, since the vast majority of agents already go to the bar. However, the *infect all* strategy increased social fairness as a function of infection rate until a certain maximum was reached, and then decreased again. For capacity of 80, it reached exactly the level of social fairness of capacity 60, as would be expected.

4. DISCUSSION AND FUTURE WORK

We presented the Behaviosite paradigm in the spirit of biomimetics — parasites manipulating their hosts' behavior. We specified the behaviosite concept, which in essence is a special type of agent, which infects and manipulates other agents to achieve altered performance of the system. We described the parasitized El Farol problem, and showed that social and personal utilities and social fairness can be increased using behaviosites; in some cases, mean attendance deviates from the capacity by only a small amount.

The parasitized El Farol problem simulated self-interested agents with external behaviosites. However, behaviosites could also be integrated into a cooperative society to form new norms or social laws, or to eliminate ill-functioning agents.

Behaviosites are not just a way of propagating false information within a system. The intimacy of the agent-behaviosite relationship induces far more powerful effects. Lies can be disregarded or overcome by agents. However, when a host is parasitized, the behaviosite is considered an almost integral part of the agent. Agents can doubt external information, but do rely on their internal beliefs.

In the future, we would like to further strengthen the Behaviosite paradigm by showing it is applicable and desirable in other scenarios. An appealing application is the automatic generation of stories; it is a rapidly growing field, with high potential in the gamers community. Behaviosites are an excellent solution for altering stories by changing some of the characters' behavior and adding new, unpredictable system behavior in a distributed manner.

5. ACKNOWLEDGMENT

This work was partially supported by grant #039-7582 from the Israel Science Foundation.

6. REFERENCES

[1] W. B. Arthur. Inductive reasoning and bounded rationality (the El Farol problem). *Amer. Econ. Review*, 84(406), 1994.

[2] A. M. Bell and W. A. Sethares. The El Farol problem and the internet: Congestion and coordination failure. Computing in economics and finance, Society for Computational Economics, March 1999.

[3] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series, 1994.

[4] A. Greenwald, J. Farago, and K. Hall. Fair and efficient solutions to the Santa Fe bar problem. Grace Hopper Celebration of Women in Computing, October 2002.

[5] A. Greenwald, B. Mishra, and R. Parikh. The Santa Fe bar problem revisited: Theoretical and practical implications. *Festival on Game Theory: Interactive Dynamics and Learning, SUNY Stony Brook*, July 1998.

[6] M. Nickles, M. Rovatsos, and G. Weiss, editors. *Post-proceedings of the First International Workshop on Computational Autonomy — Potential, Risks, Solutions*, volume 2969 of *Lecture Notes in Computer Sciences*. Springer-Verlag, 2003.

How can agent know the global information without close coordination ?

Satoshi Kurihara
kurihara@ist.osaka-u.ac.jp
8-1, Mihogaoka, Ibaraki, Osaka, 567-0047 Japan
Osaka Univ., Japan

Kensuke Fukuda
fukuda@nii.ac.jp
National Institute of Informatics, Japan

Shinya Sato
sato@ingrid.org
NTT Network Innovation Labs., Japan

Toshiharu Sugawara
sugawara@t.ecl.net
NTT Communication Science Labs., Japan

ABSTRACT

In the multi-agent systems, even if each agent behaves selfish or cooperative, its behavior influences the other agents' behaviors and their behaviors influence its behavior cyclically. Thereore a certain information about the relative position of each agent among all the agents may be able to be extracted only from its behavior. If this information can be extracted, each agent has the possibility to improve its efficiency only by seeing its own behavior without knowing about the information of the other agents. In this paper, we verified this hypothesis by using the competitive multi-agent simulation environment named Minority Game, and confirmed that each agent's performance gain was actually possible. This analytical technique may be especially useful for massively multi-agent systems.

1. INTRODUCTION

Generally, in the multi-agent systems, each agent only has local view and cannot obtain the global information of the agents as a whole. In this case, for the each agent to be able to do is to infer the global view or its relative position by getting information from the other agents. But, there is no confirmation that the correct information can be obtained. At this point, in the multi-agent system, even if each agent behaves selfish or cooperative, its behavior influences the other agents' behaviors, in other words, its behavior changes the state of the environment formed by the agents. Then, the changed environment also influences each agent's behavior, and this cycle continues. Hence, it can be thought that the behavior of each agent may include some certain information about its relative position in the agents as a whole. If these information actually exist and each agent can extract one of these information, it can know its position by only checking its own behavior and may be able to improve its behavior efficiently. The purpose of this paper is to verify the possibility of this assumption by using the Minority Game (MG)[**?**]. In MG, a lot of agents play a simple game and even though they have quite local view and behave quite selfish, certain cooperative behaviors are emerged. So, some information about each agent's performance must be included in the behavior of each agent in MG.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

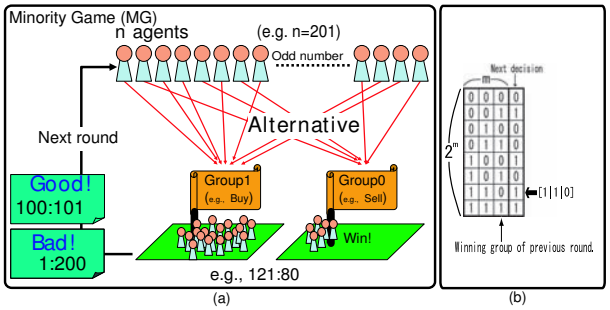


Figure 1: Rules and strategy table

2. MINORITY GAME

Firstly, we review the rules of MG (see Fig. **??**(a)). There are n (odd number) autonomous agents, each of which independently chooses between two alternatives (*group-0* or *group-1*) according to its own behavioral rules, that is, it does not see other agents' behaviors. After all of the agents choose one alternative or the other, and the agents that finish in the minority group are considered to be winners. This is one round of the game. Each winner agent is awarded one point, and the total numbers of winner agents is the profit in this round of the game. The each agent makes its selection by using one of multiple strategy tables that it holds. The entities in the table contain all combinations of m past history of the minority group choices and next decisions that corresponds to each of the combinations for the next round (see Fig. **??**(b)). At the beginning of the game, each agent prepares s ($s \geq 2$) strategy tables, and the next decision entries of each strategy table is stored randomly by 0 or 1. After the game starts, the entries of each table cannot be modified. In the first round of the game a *m-past-winner-group-history* is randomly set, for example, to [1|1|0], and each agent randomly choises one of its holding s strategy tables and sees the next decision entry corresponding to [1|1|0] (see Fig. **??**(b)). And if the next decision entry is *group-1*, the agent selects *group-1*. Then if the agent becomes winner, one point is assigned to the agent and also to the selected strategy table. And if the agent loses, one point is deducted from the selected tables. After the score of the strategy

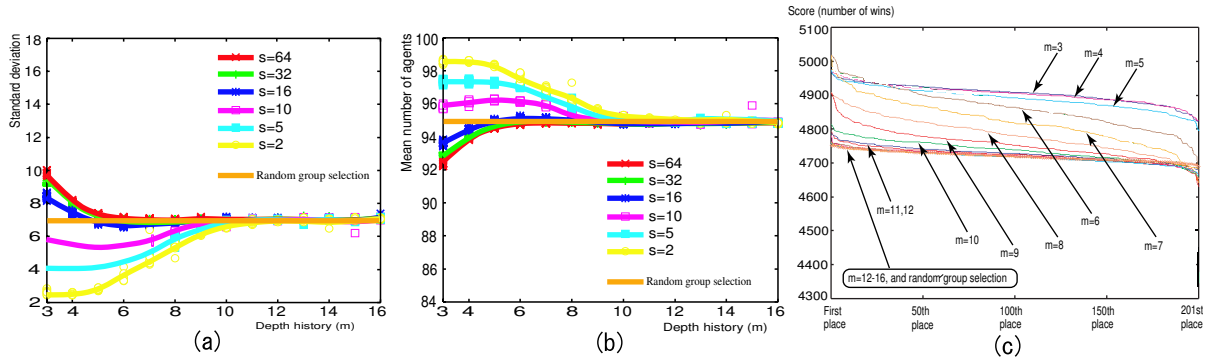


Figure 2: (a) Standard deviations of the number of winning agents. (b) Mean numbers of winning agents

tables of all the agents are updated, then the *m-past-winning-group-history* is updated from [1|1|0] to [1|0|1] because *group-1* becomes winner-group in this round of the game. In the second and subsequent rounds, each agent selects the strategy table that has the highest score. The round is repeated a predetermined number of times, and the final result of the game is the total number of points acquired by winner-agents.

2.1 Coordinated behaviors

Then, the following overall order is emerged through such the simple rules. The game was played for the number of rounds described below; One trial for each parameter pair, (s, m), $\{s|2, 5, 10, 16, 32, 64\}$, $\{m|3\ to\ 16\}$, is 10,000 rounds of the game by 201 agents; ten trials were conducted for each pair. Fig. **??**(a) shows the standard deviation of the number of winning agents, and Fig. **??**(b) shows the mean numbers of winning agents. The horizontal lines in Fig. **??**(a) and (b) represent the standard deviation and mean value when all of the agents choice *group-0* or *group-1* randomly for each round. These graphs show that, for the lower values of s, especially for s = 2, the standard deviation became lowest and the mean number of winning agents became highest when m was from about three to five. Fig. 2(c) shows the rankings of the 201 agents by average score after 10,000 rounds ($s = 2$). In the case where each agent randomly selected *group-0* or *group-1*, all of the agents were able to get approximately 4750 points. In contrast to this, the mean score was high for small m (m = 3 to 5), and although some few differences between agents can be seen in the scores, all or almost all of the agents were able to achieve stable high scores. These simulation results mean that some kind of cooperative behavior among the agents was driving the winning-group ratio closer to 100:101 in small s, especially in s = 2, and m. Especially, the most interesting characteristic is that, although we would expect behavior based on longer histories to be more efficient, m larger than 10 produce results that are the same as those of random behavior. In this paper, we mainly ran the game by several number of agents like n = 101, 201, and 301, and since our interesting is in the cooperative behavior of agents, we ran the game by s=2 (each agent has two strategy tables).

3. AGENT BEHAVIOR AND POWER-LAW

For each agent, its holding strategy tables are very important to decide its behavior, so, in [**?**], we investigated how the strategy tables are used by each agent to analyze each agent detail behavior. Then, up to now, we have discovered the following curious fact, While the high winning ratio agents used both strategy tables, there was no typical fixed period for the continuous use of one strategy table. That is , a self-similarity was visible in these results. Fig. **??** shows, on log-log scales, per-agent histograms of the periods over which either of the strategy tables was continuously used. Results for m=3, 7, and 14 are given for (a) n = 101, and (b) n = 301. Power-law can be seen in both cases of m=3 (graphs were nearly straight lines) and this also reflects the presence of a self-similarity. To the contrary, when m=14 power-law is not seen. Interestingly, the histograms for agents with m=7 show a mix of the two types of results; some agents are similar to the graphs for m=3, while others are similar to those for m=14. It can be thought that the straight lines are for *cooperative agents*, while the curves are for *non-cooperative agents*. So, we calculated the all slopes of the histograms in Fig. 3 (see Fig. **??**). As you can see, the consistent relation can be seen that is the slope of high winning-ratio agent is more horizontal (nearer to 0) than that in low winning-ratio agent. That is, it can be thought that each agent can know its rough position among the group by seeing whether the way of its own strategy table selection obeys the power-law or not.

4. IMPROVEMENT OF WINNING-RATIO USING POWER-LAW FEATURE

The easiest way to change the agent's behavior is to renew its holding strategy tables. So, the following experiments in which each agent renews its strategy tables by three rules were done. As for the experiment, we ran the game by 101 agents (n = 101, s=2). After the game is repeated 10,000 times (10,000 times is one turn), each agent's efficiency is evaluated by the following three kinds of criteria, and each agent which is identified as inefficient renews its strategy tables, then next 10,000 times is started cyclically. In the experiment we repeated the renewing process 10 times that is we ran the game total 100,000 times. We prepared these three evaluation-rules:

(evaluation-1) Since the agent with its power-law slope being nearly 0 has high winning-ratio, we randomly select u pieces of agents whose power-law slope become lower than the previous turn, and selected u pieces of agents renew their strategy tables. In this evaluation, since the calculation of each agent's power-law is computable only from its behavior, each agent need not know the total of agents in the game. However since this evaluation uses only the change of slope even the high winning-ratio agent may renew its strategy tables.
(evaluation-2) We randomly select u pieces of agents whose power-law slope become lower than the certain value (0.75), and selected agents renew their strategy tables. In this evaluation, we use the following knowledge that is, there are consistent relation between the

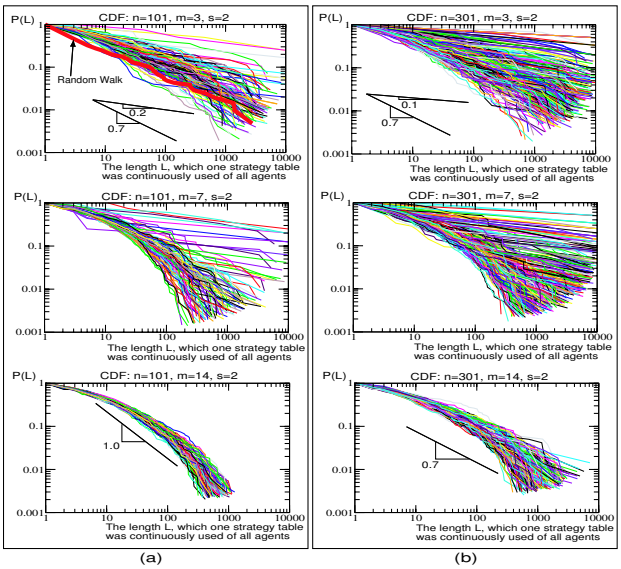


Figure 3: No. of steps L over which either strategy table was in continuous use for (a) n=101 and (b) n=301 with (upper) For m=3 (all 101 agents and 301 agents), (middle) For m=7 (all 101 agents and 301 agents) (lower) For m=14.

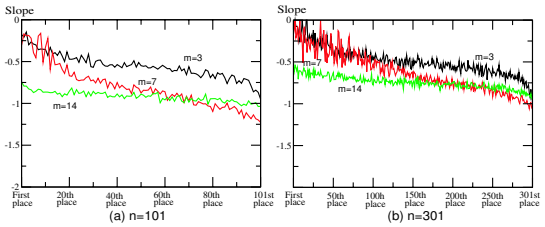


Figure 4: Slopes of the histograms in Fig. 5.

power-law slope and the winning-ration in each agent and this consistent relation does not depend on the total of participating agents in the game. In this evaluation, each agent can know its approximate position without knowing the total of participating agents, and agents only having low winning-ratio can renew their strategy tables.

(evaluation-3) We select u pieces of agents having low score, and selectedagents renew their strategy tables. In this evaluation, it is necessary for each agent to know the following global information, each agent's score and the number of total of participating agent. This evaluation can be thought as the best way to select the agent to renew its strategy table. We prepared this evaluation for the comparison to above evaluation-1 and -2.

Fig. **??** shows the score of 101 agents of nine kinds of evaluation settings. We ran the three kinds of games (u=5, u=10, and u=20) for each evaluation. And as mentioned above, in each situation, strategy table renewing process was repeated 10 times and score of agents after the each renewing process was plotted to the graph of each evaluation setting. As a result, in all settings it can be confirmed that the winning-ratio has improved in all almost agents. First of all, u = 5 was the best in all three kinds of evaluation. The reason can be thought as follows; if we select too many agents, many renewed strategy tables may destroy the current good strategy table's combination of high winning-ratio agents and thus the effect of renewing may be lost. As for the evaluation-1 and -2,

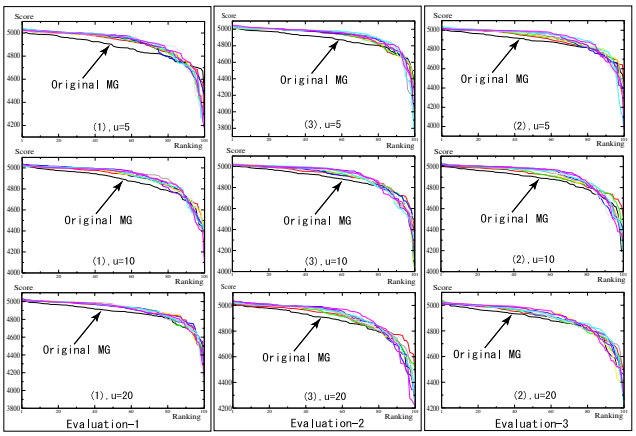


Figure 5: Score of 9 kinds of evaluation settings

evaluation-2 was better than evaluation-1 as we expected because in evaluation-2, only the agents having low winning-ratio are able to renew their strategy tables. Consequentially, evaluation-2 was the most effective though it had been expected that evaluation-3 was at first most effective. While the evaluation-3 uses the several global information, it pays attention only to the score transition of agent. On the other hand, the evaluation-1 and -2 pay attention to the agent behavior. So, in the following two situations, "the score is good by chance even if the evaluation of behavior is bad" and "the score is bad by chance even if the evaluation of behavior is good", agent should not renew its strategy tables in the former situation, and should renew them in the latter situation, thus this experiment result shows paying attention to agent behavior is effective as the evaluation methodology.

5. CONCLUSION

In this paper, we verified whether the efficiency of each agent behavior was able to be improved only by seeing its behavior in MG. In MG, important information for the agent to know the efficiency of its behavior was power-law. Power-law is most typical factor of scal-free network. For example, in the Internet it is known that power-law can be seen in the flow of packets when the number of packets of flowing is the maximum in the traffic [**?**]. This means if the flow of traffic can be controlled to follow of power-law when the flow of traffic becomes an inefficiency situation due to some troubles, the flow of traffic may be corrected to an efficient situation. In the future work, we also plan to apply this coordination methodology to an actual system like the ubiquitous information communication systems and to verify whether an efficient performance improvement is possible.

6. REFERENCES

[1] D. Challet, M. Marsili and R. Zecchina, Phase Transition in a Toy market (1999); on the major Minority Game web site: http://www.unifr.ch/econophysics/minority/

[2] S. Kurihara, K. Fukuda, T. Hirotsu, O. Akashi, S. Sato and T. Sugawara, How collective intelligence emerge in complex environment?, *Proceedings of Bio-ADIT 2004*, pp. 498 - 511, (2004)

[3] K. Fukuda, M. Takayasu, and H. Takayasu, Analysis of Minimal Model of Internet Traffic, *Traffic and Granular Flow'01 (M. Fukui, Y. Sugiyama, M. Schreckenberg, and D. E. Wolf Eds.)*, Springer, (2002).

Strategic voting when aggregating partially ordered preferences

M. S. Pini, F. Rossi, K. B. Venable
Department of Pure and Applied Mathematics
University of Padova, Italy
{mpini,frossi,kvenable}@math.unipd.it

T. Walsh
NICTA and School of CSE, UNSW
Sydney, Australia.
tw@cse.unsw.edu.au

ABSTRACT

Preferences of a single agent are often partially ordered. For example, it may be hard to compare a novel with a biography. In such a situation, the agent may want the novel and the biography to be considered incomparable. We consider here how to aggregate the partially ordered preferences of multiple agents in order to return a set of most preferred outcomes. We define the notion of strategy-proofness for such a scenario. This is when preference aggregation cannot be manipulated. We prove that if there is no dictator, agents can manipulate the result by voting strategically to determine the most preferred outcomes. This extends the well-known theorem by Gibbard and Satterthwaite for total orders.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Distributed Artificial Intelligence*

General Terms

Theory

Keywords

Preference aggregation, strategy-proofness, partial orders, Gibbard Satterthwaite theorem

1. INTRODUCTION

Many problems require us to combine the preferences of different agents. For example, when planning a wedding, we must combine the preferences of the bride, the groom and possibly some or all of the in-laws. Typically such preferences are partial orders over the outcomes. We can view many different mechanisms for preference aggregation in terms of voting. Each agent is voting for their preferred outcomes. When aggregating preference orders, one may be interested in obtaining a combined ordering among the outcomes, or just the set of most preferred outcomes (also called the winners). The first choice, in which the aggregation function is called a *social welfare* function, provides more information about the combined preference ordering since it tells us the ordering between any pair of outcomes. The second scenario, in which the aggregation function, called a *social choice* function, returns a set, is less informative but often enough when we are just interested in choosing one of the most preferred outcomes.

In this paper we will consider social choice functions. Thus the result of aggregating preferences will be just a set of outcomes. A very desirable property of preference aggregation is non-manipulability (also called strategy proofness). It should not be possible for agents to manipulate the election by voting strategically. Strategic voting is when agents express preferences which are different from their real ones, to get the result they want. If this is possible, then the preference aggregation rule is said to be manipulable. For social choice rules on *totally* ordered preferences, the Gibbard Satterthwaite theorem [3] proves that it is not possible to be at same time non-manipulable and have no dictators. Either there is a dictator (that is, an agent who gets what he wants by voting sincerely) or a manipulator (that is, an agent who gets what he wants by lying). In either case, there is an agent who gets what he wants no matter what the other agents say. In this paper we extend this result to *partially* ordered preferences. Even in this more general case, we prove that it is impossible for a social choice function to have no dictator and be non-manipulable at the same time. As with total orders, we conjecture that there will be ways around this negative result. For example, it may be that certain social choice functions on partial orders are computationally hard to manipulate. As another example, it may be that certain restrictions on the way agents vote (like single-peaked preferences for total orders) guarantee strategy-proofness.

2. SOCIAL CHOICE ON PARTIAL ORDERS

In some situations, the result of aggregating the preferences of a number of agents might not need to be an order over outcomes. It might be enough to know the "most preferred" outcomes. Social choice functions identify such most preferred outcomes.

A **profile** p is a sequence of n orderings $p_1, \ldots, p_n$ over outcomes, one for each agent $i \in \{1, \ldots, n\}$. A social choice function on total orders is a mapping from a profile to the optimal outcome, or winner. With partial orders, (i.e. binary relation which are reflexive, transitive, antisymmetric

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

and not always complete), there can be several outcomes which are incomparable and optimal. We can therefore consider a generalization in which a social choice function is a mapping from a profile to a non-empty set of outcomes, called the optimal outcomes, or the winners.

Let us now consider some properties of such social choice functions. Our definition of these properties is a generalization of the corresponding definition for social choice function on totally ordered profiles [1]. We say that a social choice function f is **unanimous** iff

- for any profile p with $a \in top(p_i)$ for every agent i, then $a \in f(p)$;
- for any profile p with $\{a\} = top(p_i)$ for every agent i, then $f(p) = \{a\}$.

It is instead **monotonic** iff, given two profiles p and p',

- if $a \in f(p)$ and, for any other alternative b, $a >_{p_i} b$ implies $a >_{p'_i} b$, $a \bowtie_{p_i} b$ implies $a \bowtie_{p_i} b$ or $a >_{p'_i} b$, for all agents i, then $a \in f(p')$;
- if $f(p) = A$ and for all $a \in A$, for all b, $a >_{p_i} b$ implies $a >_{p'_i} b$ and $a \bowtie_{p_i} b$ implies $a \bowtie_{p'_i} b$ or $a >_{p'_i} b$, for all agents i, then $f(p') = A$.

We define three notions of dictators for social choice functions: a **strong dictator** is an agent i such that, for all profiles p, $f(p) = top(p_i)$; a **dictator** is an agent i such that, for all profiles p, $f(p) \subseteq top(p_i)$; a **weak dictator** is an agent i such that, for all profiles p, $f(p) \cap top(p_i) \neq \emptyset$. Notice that, in any profile p, if a is the unique top of a weak dictator i, then $a \in f(p)$. However, this is not true if a is not the unique top of i.

We say a social choice function is **strongly fair**, **fair** or **weakly fair** if it is unanimous, monotonic, and does not have a strong dictator, dictator or weak dictator, respectively. Since a strong dictator is a dictator, and a dictator is a weak dictator, weak fairness implies fairness which itself implies strong fairness. In [6], we generalize the Muller-Satterthwaite theorem [5] to social choice functions over partial orders without ties, for weak dictators. That is, weak dictators are inevitable if we have two or more agents, three or more outcomes and the social choices function is unanimous and monotonic. We rewrite here this theorem formally since it will be used in the rest of the paper.

Theorem 1 ([6]) *If we have at least two agents and at least three outcomes, and a social choice function on partial order without ties is unanimous and monotonic, then there is at least one weak dictator.*

3. STRATEGY PROOFNESS

The Gibbard Satterthwaite theorem on totally ordered preferences [3, 7] proves that either we have a dictator or the social choice function can be manipulated. That is, agents can manipulate the result using tactical voting. We may now wonder if a similar relationship holds between weak dictators and non-manipulability. To answer this question, we must generalize the notion of non-manipulability (also called strategy proofness) to social choice functions on partially ordered preferences, and then consider the Gibbard Satterthwaite theorem [3, 7] in this more general context.

We will show that weak dictators are inevitable if we have at least two agents and three outcomes, and the social choice function is strategy proof and onto. These conditions are identical to those in the Gibbard Satterthwaite theorem for total orders. A social choice function is strategy-proof if it is best for each agent to order outcomes as she really prefers and not to try to order them tactically, with the hope of getting a better result. More precisely, the social choice function must never allow an agent to get a preferred outcome among the winners by ordering outcomes in a way that contradicts his or her true preferences [1].

Formally, a social choice function is **strategy proof** if, for every agent i, for every pair of profiles p and p', which differ only for the agent i's ranking, that is, p_i, we have that

- $\forall a \in f(p) - f(p')$, $\forall b \in f(p')$, if $a \bowtie_{p_i} b$, then $a \bowtie_{p'_i} b$ or $a <_{p'_i} b$; if $a <_{p_i} b$, then $a <_{p'_i} b$;
- $\forall a \in f(p) - f(p')$, $\exists b \in f(p')$ such that, if $a >_{p_i} b$, then $a \bowtie_{p'_i} b$ or $a <_{p'_i} b$; if $a \bowtie_{p_i} b$, then $a <_{p'_i} b$.

In other words, a social choice function is strategy proof if an agent can remove an element a from the set of winners only by worsening its rank with respect to at least one the new winners b, and not improving it with respect to any other new winner. This means that it is not possible for an agent to make a disappear from the set of winners by improving its ranking in her preference ordering. In fact, this would be tactical voting.

In general, even in the totally ordered case, most voting procedures involving three or more alternatives are not strategy proof [1]. This is true also in the partially ordered case. Let us consider the following two social choice functions: f_1 is such that $f_1(p) = \bigcup_i top(p_i)$, that is, this function returns the union of the sets of optimal elements of each agent and f_2 is the Pareto function, which returns the optimal elements of the ordering returned by the Pareto social welfare function (where $a > b$ if all agents say $a > b$, otherwise $a \bowtie b$).

Let us now consider two profiles p and p' on three alternatives a, b and c such that $p = (p_1 = (c > a \wedge c \bowtie b \wedge a \bowtie b)$, $p_2 = (c \bowtie a \wedge c \bowtie b \wedge a \bowtie b)$, $p_3 = p_1)$ and $p' = (p'_1 = p_1$, $p'_2 = (c > a > b)$, $p'_3 = p_3)$. Then, for both such social choice functions, the set of winners in profile p is $\{a, b, c\}$, while in profile p' is $\{b, c\}$. Thus a has disappeared by passing from p to p' but its ranking has improved with respect to b in agent 2. Thus both social choice functions are not strategy-proof.

In order to prove the generalization of the Gibbard Satterthwaite theorem to the partial order case we first prove the following result, which relates strategy proofness to unanimity and monotonicity.

Theorem 2 *If a social choice function is strategy proof and onto, then it is unanimous and monotonic.*

PROOF. The proof is composed by two parts. Part 1 shows that if a social choice function f is strategy proof then it is monotonic, while part 2 shows that if f is onto and monotonic then it is unanimous.

Part 1. Consider two profiles p and p', which differ only for the ranking of *agent* i.

Assume that $a \in f(p)$ and that for any other alternative b, $a >_{p_i} b$ implies $a >_{p'_i} b$ and $a \bowtie_{p_i} b$ implies $a \bowtie_{p_i} b$ or $a >_{p'_i} b$. We want to show that $a \in f(p')$. For the sake of

contradiction, assume that $a \notin f(p')$. Since f is strategy proof, then $\exists c \in f(p')$ such that one of the following holds: (i) if $a >_{p_i} c$, then $a \bowtie_{p'_i} c$ or $a <_{p'_i} c$, (ii) if $a \bowtie_{p_i} c$, then $a <_{p'_i} c$. If the first holds then there is an element c which is worse than a in p_i and that becomes strictly better than or incomparable to a in p'_i. This contradicts the fact that for any other alternative b, $a >_{p_i} b$ implies $a >_{p'_i} b$. If the second holds then there is an element c which is incomparable with a in p_i and becomes strictly better then a in p'_i. This contradicts the fact that for any other alternative b, $a \bowtie_{p_i} b$ implies $a \bowtie_{p'_i} b$ or $a >_{p'_i} b$.

Assume now that $\forall a \in f(p)$, for any other alternative b, $a >_{p_i} b$ implies $a >_{p'_i} b$ and $a \bowtie_{p_i} b$ implies $a \bowtie_{p_i} b$ or $a >_{p'_i} b$. We want to show that $f(p) = f(p')$. For the sake of contradiction, we can assume that $\exists a$ such that $a \in f(p)$ and $a \notin f(p')$ or that $\exists a$ such that $a \in f(p')$ and $a \notin f(p)$. If $\exists a$ such that $a \in f(p)$ and $a \notin f(p')$ then, since f is strategy proof, the same reasoning above leads to the same contradictions. If instead $\exists a$ such that $a \in f(p')$ and $a \notin f(p)$, then since f is strategy proof, then $\exists c \in f(p)$ such that one of the following holds: (i) if $a >_{p'_i} c$, then $a \bowtie_{p_i} c$ or $a <_{p_i} c$, (ii) if $a \bowtie_{p'_i} c$, then $a <_{p_i} c$. In the first case there is an element a which is stricly smaller than or incomparable to c in p_i that becomes strictly greater than c in p'_i. This is in contradiction either with the fact that for any other alternative b, $c \bowtie_{p_i} b$ implies $c \bowtie_{p'_i} b$ or $c >_{p'_i} b$ or with the fact that $c >_{p_i} b$ implies $c >_{p'_i} b$. If the second case holds then there is an element a that is smaller than c in p_i and that becomes incomparable with c in p'_i. This contradicts the assumption that for any other alternative b, $c >_{p_i} b$ implies $c >_{p'_i} b$.

Consider two profiles q such that $f(q)$ $\{a\}$ and a profile q' such that *for every agents* i and for every alternative b, $a >_{q_i} b$ implies $a >_{q'_i} b$. We want prove that $f(q')$ $\{a\}$, that is the first part of the definition of the monotonicity for social choice functions. Since we can move from $q = (q_1, ...q_n)$ to $q' = (q'_1, ...q'_n)$, passing from $q = (q_1, q_2..., q_n)$ to $(q'_1, q_2..., q_n)$, and $(q'_1 q'_2, ..., q_n)$ to $(q'_1, q'_2, ..., q_n)$ and so on, and we have shown above that at each step a remains in the set of winners, $a \in f(q')$. The same reasoning holds for profiles q such that $f(q) = A$ and q' such that for every agent i, for $\forall a \in A$, for every other alternative b, such that $a >_{q_i} b$ implies $a >_{q'_i} b$. In this case we conclude that $f(q') = A$. This is the second part of the definition of monotonicity for social choice function. We have thus shown that f is monotonic.

Part 2. Since f is onto, then for every subset S of alternatives there is a profile p such that $f(p) = S$. If $S = \{a\}$, where is a is an alternative, since f is onto, there is a profile p such that $f(p) = \{a\}$. If we consider the profile p', obtained from profile p bringing a to the very top of every agent, then for strict monotonicity on profiles p and p', that we have just proved, $f(p') = a$. Therefore, whenever a is the unique top of every agent's ranking in a profile $\bar{p}$, then $f(\bar{p}) = \{a\}$. Because a is arbitrary then f satisfies Pareto efficiency in the case of unique top for every agent.

If $S \supset \{a\}$, since f is onto, then there is a profile p_1 such that $f(p_1) = S \supset \{a\}$. If we consider a profile p'_1 where a is at the top (not unique) for every agent, then, for monotonicity on profiles p_1 and p'_1, $f(p'_1)$ must contain a. Therefore, whenever a is one of the tops of every agent's ranking in a profile $\bar{p}$, then $f(\bar{p})$ $\{a\}$. Because a is arbitrary, then f satisfies Pareto efficiency in the case of not unique top for every agent. Q.E.D.

We now use this result, together with the extension of the Muller-Satterwaithe result (Theorem 1), to prove the main theorem stating that it is not possible for a social choice function to be at the same time strategy proof and onto, and have no weak dictators.

Theorem 3 *If there are at least two agents and at least three outcomes and the social choice function is strategy-proof and ontothen there is at least one weak dictator.*

PROOF. By Theorem 2, if f is onto and strategy proof then it is monotonic and unanimous. Moreover, Theorem 1 states that if f is monotonic and unanimous then there is at least one weak dictator. Q.E.D.

4. RELATED WORK

Efforts have been made to weaken the conditions of the Gibbard Satterthwaite theorem. For example, in [2], the domain of the social choice function has been generalized to preferences over sets of outcomes and an impossibility result proved. The Gibbard Satterthwaite theorem has been shown to be robust to several other restrictions of the domain of the social choice function. Here we have shown that this theorem is robust also to including incomparability in the preference ordering.

5. REFERENCES

[1] K. J. Arrow and A. K. Sen and K. Suzumara. *Handbook of Social Choice and Welfare.* North-Holland, Elsevier, 2002.

[2] S. Barbera. Strategy-Proofness and Pivotal Voters: A Direct Proof of the Gibbard-Satterthwaite Theorem. *International Economic Review*, Vol.24, No.2, pages 413–417, 1983.

[3] A. Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41, 1973.

[4] K. Konczak and J. Lang. Voting procedures with incomplete preferences. IJCAI 05 Workshop on Advances in Preference Handling, 2005.

[5] E. Muller and M. A. Satterthwaite. The equivalence of strong positive association and strategy-proofness. *Economic Theory*, 14, 1977.

[6] M. S. Pini, F. Rossi, K. B. Venable, T. Walsh. Aggregating partially ordered preferences: impossibility and possibility results. *Proc. TARK X*, ACM Digital Library, June 2005.

[7] M. A. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.

Role-Based Semantics for Agent Communication

Embedding of the 'Mental Attitudes' and 'Social Commitments' Semantics

Guido Boella, Rossana Damiano
Università di Torino
Italy
{guido,rossana}@di.unito.it

Joris Hulstijn
Vrije Univ. Amsterdam
The Netherlands
jhulstijn@feweb.vu.nl

Leendert van der Torre
University of Luxembourg
Luxembourg
leendert@vandertorre.com

ABSTRACT

In this paper we illustrate how a role-based semantics for agent communication languages can embed the two main existing models of agent communication languages, respectively based on 'mental attitudes' and 'social commitments' semantics. These two models have been presented as incompatible approaches, but recently we illustrated for persuasion dialogues and using our normative multi-agent systems framework, that they can be seen also as complimentary ones. Independently from our own multi-agent model, in this paper we illustrate for the speech act 'inform' how the role based semantics embeds the other two semantics.

1. SYNTHESIS

In [1] we argue that a *role metaphor* can be used to bridge the gap between the mental attitudes approach [4] and the social commitments approach [8] to the semantics of agent communication languages. We show how dialogues can be modelled as games in normative systems and how mental attitudes can be attributed not only to agents, but also, in a public manner, to the roles of the game. The dialogue moves allow an agent playing a role to modify the roles' mental states, as specified by so-called counts-as conditionals or constitutive norms defining the game. The player of a role is expected to act as if he has the mental attitudes attributed to his role and to prevent his role's mental attitudes from becoming incoherent, just as he does for his own private mental attitudes.

Thus, we embed the mentalistic approach by showing how mental attitudes can be attributed to public roles instead of agents. The motivation of maintaining a mentalistic semantics, albeit referred to roles, is to be able to reuse the extensive FIPA work on the agent communication semantics. It is sufficient to refocus the model from the agents' beliefs and goals to the roles' beliefs and goals. However, though the mentalistic approach is embedded, the following drawbacks of the FIPA approach are not inherited.

- Communication is intersubjective and public, while beliefs and goals are private and, thus, not accessible by the receiver.
- An independent observer cannot verify whether agents conform to the agent communication semantics [11].
- The sincerity assumption that is necessary to reason with mentalistic semantics does not apply to non-fully cooperative dialogue contexts, like argumentation and negotiation.

Moreover, we embed the social approach by showing how roles as descriptions of expected behavior maintain the normative character of social semantics. Roles are useful because mental attitudes attributed to roles of a dialogue game capture the public character of meaning which is offered by commitment approaches. The following limitations of the social commitments approach are not inherited by our role-based semantics:

- Commitment interpreted as an obligation is too strong in cooperative dialogues, where the notion of expectation is sufficient. Moreover obligations are associated with sanctions, which are specific only in some kinds of dialogues.
- Commitment is related to the notion of obligation, while the notion of permission is not considered even if it is necessary to explain the notion of concession. Walton and Krabbe [10] argue that after an 'inform' the receiver can make a concession explicitly, or implicitly by not challenging or failing to challenge the 'inform'. In case of concessions the receiver becomes weakly committed to the proposition: he can no longer make the speaker committed to defending the proposition by challenging him, albeit the receiver does not have to defend the proposition himself if challenged. This situation cannot be expressed in social semantics.
- Propositional commitment is not clearly distinguished from action commitment. Thus it has the same conditions of fulfillment as action commitment, while propositional commitment for Walton and Krabbe is fulfilled when the receiver concedes, instead of when the commitment is true in the world.
- The action of informing is not related with any effect on the hearer, only the attitude of the speaker is considered. A precondition that the speaker does not believes that the receiver knows what is communicated is not expressible.

Our approach in [1] discusses persuasion dialogues and assertive speech acts, and uses our normative multi-agent systems framework describing roles via the agent metaphor and formalizing mental attitudes of the agents in Input/Output logic [7]. In this paper we consider the general problem of embedding mentalistic and social semantics in a role based semantics, independently from the dialogue type or agent model. In the following section we use visualizations of the communicative act 'inform' to illustrate how the role-based semantics can be seen as a natural combination of the 'mental attitudes' and 'social commitments' semantics. In the final section we sketch some formal aspects.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. EMBEDDING IN ROLE SEMANTICS

Figure 1 visualizes the communicative act 'inform' in a mental attitudes approach like the one provided by FIPA [4]. This figure should be read as follows. The two ovals represent the two agents x and y. Within the ovals the mental state of the agent is represented. The top half of the oval contains the beliefs (B) of the agent, and the lower half his goals (G). Agent x thus has the belief p and the goal that agent y believes p; in a modal language, these may be represented by $B_x p$ and $G_x B_y p$. Note that the outer modalities are not represented in the figure. Agent y believes p too if he trusts x. The arrow between the agents visualizes a communication channel.

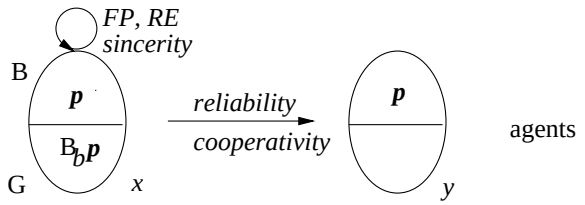


Figure 1: Mental attitudes semantics

In the mental attitudes semantics, communicative acts are defined in terms of the mental state of the BDI agent who issues them. The bridge between the communicative acts and the behavior of agents is provided by the notions of rational effect and feasibility preconditions. The rational effect (RE in Figure 1) is the mental state that the speaker has the goal to bring about in the receiver by issuing a communicative act; the feasibility preconditions (FP) encode the appropriate mental states for issuing a communicative act. From the execution of a communicative acts it can be inferred that the speaker believes the feasibility preconditions and has the goal to achieve the rational effects. This kind of inference can be compared to presupposition accommodation [9]. However, this inference does not guarantee that the speaker actually has these attitudes nor that he would agree that these are the effects of a communicative act. Moreover, RE and FP are concerned only with the speaker, since they do not say anything about the receiver. Thus, assumptions about the reliability and sincerity of the speaker and about the cooperativity of the receiver are necessary to infer that the receiver believes what has been communicated, or wants to comply with a request.

Figure 2 visualizes the same communicative act 'inform' in the social commitment approach. This figure should be read as follows. Again there are a speaker x and a hearer y with their beliefs and goals. Moreover, there are also two commitment stores, represented by the two squares, but not distinguished between propositional and action commitment. The arrows represent that due to agent x informing p, and to avoid a sanction, he may believe it. In this way, agent communication languages based on social commitment constitute an attempt to overcome the mentalistic assumption of FIPA by restricting the analysis to the public level of communication.

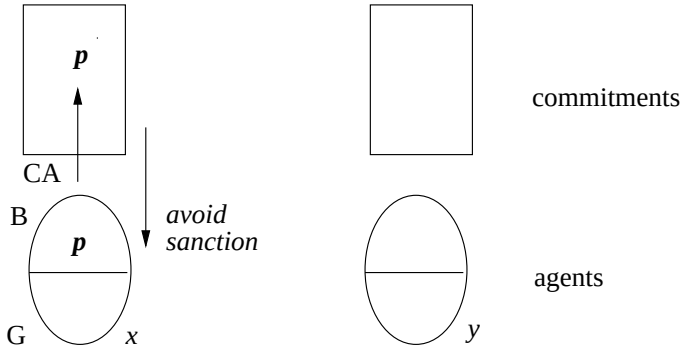


Figure 2: Social commitment semantics

In the social commitments approach, communicative acts are defined in terms of the social commitments they publicly determine for the speaker and the hearer (see Figure 2). However, there are many ways in which this has been made more precise, and the term of commitment has been used in different ways. For example, Walton and Krabbe focus on propositional commitments: "to assert a proposition may amount to becoming committed to subsequently defending the proposition, if one is challenged to do so by another speaker" [10]. However, this kind of propositional commitment is biased towards argumentation dialogue, thus failing to be general enough for other kinds of dialogue. By contrast, Singh is interested in negotiation, and he investigates action commitments, that typically result from commissives and directives. According to Fornara and Colombetti [5], commitment is "a social relationship between the speaker and the hearer".

In Figure 3 we visualize the role-based semantics of 'inform'. The ovals x and y represent two agents playing respectively the roles a and b in the dialogue game. The constitutive rules of the game operate on the mental states of both roles. Only external assumptions like sincerity attribute beliefs to the agents, but they are not part of the game. The basic idea is that communicative acts can be modelled as plan operators with preconditions and effects which can refer to beliefs, goals and intentions, but the mental attitudes they refer to are not the private inaccessible ones of the agents. Rather, the beliefs, goals and intentions to which speech acts refer are attributed to a public image of the participants in the dialogue representing the role they play.

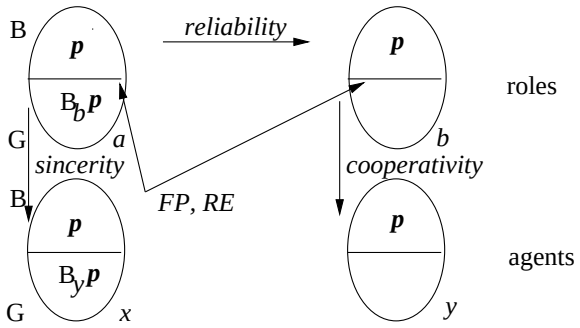


Figure 3: Role-based semantics

Of course, communication among agents is often associated with the roles played in the social structure of the systems, and role names like 'speaker' and 'addressee' or 'buyer' and 'seller' are often mentioned in the definition of agent communications languages. However, usually these terms only serve the function to bind individual agents to the speech acts in the protocol, whereas in our role semantics they are associated with a state which changes during the conversation as a result of the performed speech acts. Technically, attitudes are attributed to agents enacting a role; we just abstract over the agent. So what we are modelling, is similar to the public mental states associated with role enacting agents (REAs) in the model of Dastani *et al.* [3]. A similar solution is also proposed by Gaudou *et al.* [6], where beliefs (but not goals) can be publicly attributed to agents by means of a grounding operator.

3. TOWARDS FORMALIZATION

In the following, we sketch how a role-based semantics can be defined for FIPA and social semantics. With some adjustments of notation, the FIPA definition of the *inform* communicative act is as follows.

inform(a, b, p)
FP: $B(a, p) \wedge \neg B(a, B(b, p) \vee B(b, \neg p))$
RE: $B(b, p)$

We assume that a, b are variables that range over roles, whose attitudes are maintained in public. The inference that can be based on the *feasibility preconditions* (FP) of a speech act, is modeled in the following way:

$$\textit{inform}(a, b, p) \rightarrow B(a, p)$$
$$\textit{inform}(a, b, p) \rightarrow \neg B(a, B(b, p) \vee B(b, \neg p))$$

Because roles' beliefs are maintained in public, these preconditions can indeed be verified.

The inference that is based on the *rational effect* (RE) is as follows. This inference is based on a kind of abduction. Uttering a speech act only makes sense when the rational effect is a goal of the speaker.

$$\textit{inform}(a, b, p) \rightarrow G(a, B(b, p))$$

In the social commitments model, speech acts introduce commitments in the dialogue state or manipulate them. A commitment has a *debtor* and a *creditor*, i.e., respectively, the agent who has the commitment, and the agent to which the commitment is made. A commitment can have different states: unset (i.e., to be confirmed), active (i.e., confirmed), fulfilled, *etc*.

In our translation, a commitment state corresponds to a specific configuration of roles' beliefs and goals. So the commitment to defend a proposition (c.f. [10]), is treated as equivalent to a public belief: the agent must not contradict himself by means of conflicting speech acts or by failing to defend his beliefs.

We focus here on propositional commitment and concessions. Consider a dispute between debtor and creditor about a propositional variable p. A propositional commitment PC is *active* whenever the debtor publicly believes the proposition, while nothing is required for the creditor:

$$PC(\text{active}, a, b, p) \equiv B(a, p)$$

A propositional commitment PC is *fulfilled* when the creditor concedes the proposition:

$$PC(\text{fulfilled}, a, b, p) \equiv \neg B(b, \neg p)$$

Regarding the effect of speech acts in such a dispute, we can say that a *concession*, means that the debtor does not publicly believe (and defend) the opposite:

$$\text{concede}(a, b, p) \rightarrow \neg B(a, \neg p)$$

An 'inform' introduces an active propositional commitment of the speaker and, if it is not challenged, also a concession of the receiver:

$$\text{inform}(a, b, p) \rightarrow B(a, p)$$
$$\text{inform}(a, b, p) \wedge \neg\text{challenge}(b, a, q \rightarrow \neg p, q) \rightarrow \neg B(b, \neg p)$$

The second part of the rule prevents the player of role b from asserting the opposite proposition $\neg p$ in the subsequent dialogue, since this would lead to a contradiction: $\neg B(b, \neg p) \wedge B(b, \neg p)$.

Please note the similarity between the inferences that can be made based on FIPA-style FP and RE, and the inferences that can be made on the basis of propositional commitments. For both the $\text{inform}(a, b, p)$, we get $B(a, p)$ and $\neg B(b, \neg p)$: the difference rests in the effect of challenging or failing to defend beliefs.

Thus, in our framework we are able to model both the mental attitudes and social commitment approaches, but avoiding their major limitations. First, with respect to FIPA, mental attitudes are public, because they are attributed to roles; second, the semantics of speech acts is public too since it consists in public constitutive rules; third, the model is not restricted to cooperative situations, since sanctions can be added to the expectations represented by roles [2].

With respect to social commitment, first, we can consider also cooperative situations were expectations are sufficient; second, as in Walton and Krabbe we can model the notion of concession, but, with respect to them, we can also distinguish between propositional and action commitment by means of the distinction between roles' beliefs and goals. Finally, we can refer to beliefs and goals of roles to express more complex preconditions on speech acts.

4. REFERENCES

[1] G. Boella, J. Hulstjin, and L. van der Torre. A synthesis between mental attitudes and social commitments in agent communication languages. In *Procs. of IAT'05*. IEEE Press, 2005.

[2] G. Boella and L. van der Torre. A game theoretic approach to contracts in multiagent systems. *IEEE Transactions on Systems, Man and Cybernetics - Part C*, 2006.

[3] M. Dastani, V. Dignum, and F. Dignum. Role-assignment in open agent societies. In *Procs. of AAMAS'03*, pages 489–496, New York (NJ), 2003. ACM Press.

[4] FIPA. FIPA ACL specification. Technical Report FIPA00037, Foundation for Intelligent Physical Agents, http://www.fipa.org/specs/fipa00037/SC00037J.html, 2002.

[5] N. Fornara and M. Colombetti. A commitment-based approach to agent communication. *Applied Artificial Intelligence*, 18(9-10):853–866, 2004.

[6] B. Gaudou, A. Herzig, and D. Longin. A logical framework for grounding-based dialogue analysis. In W. van der Hoek, A. Lomuscio, E. de Vink, and M. Wooldridge, editors, *Procs. of LCMAS'05*, 2005.

[7] D. Makinson and L. van der Torre. Input-output logics. *Journal of Philosophical Logic*, 29:383–408, 2000.

[8] Munindhar P. Singh. A social semantics for agent communication languages. In Frank Dignum and Mark Greaves, editors, *Issues in Agent Cmmunication: Proceedings of the IJCAI Workshop on Agent Communication Languages*, LNCS 1916, pages 31 – 45. Springer-Verlag, London, UK, 2000.

[9] Robert C. Stalnaker. Pragmatic presupposition. In M.K. Munitz and P.K. Unger, editors, *Semantics and Philosophy*, pages 197–213. New York University Press, New York, 1974.

[10] Douglas N. Walton and Erik C. Krabbe. *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. State University of New York Press, 1995.

[11] Michael. J. Wooldridge. Semantic issues in the verification of agent communication languages. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(1):9–31, 2000.

Teaching New Teammates

Doran Chakraborty
Department of Mathematics & Computer Science
The University of Tulsa
USA
doran@utulsa.edu

Sandip Sen
Department of Mathematics & Computer Science
The University of Tulsa
USA
sandip@utulsa.edu

ABSTRACT

Knowledge transfer between expert and novice agents is a challenging problem given that the knowledge representation and learning algorithms used by the novice learner can be fundamentally different from and inaccessible to the expert trainer. We are particularly interested in team tasks, robotic or otherwise, where new teammates need to replace currently indisposed team member(s). We are interested in a general knowledge transfer framework where existing team members or experts can train a new agent to follow its role in team coordination by using exemplars of desirable behavior. Each such exemplar presents a team situation and a preferred action. We envisage an iterative training process where the trainer selects more exemplars in the next iteration based on the errors made by the learner in action choices for test exemplars presented in the current iteration. Such an iterative, exemplar based generic knowledge transfer scheme can be used by agents using arbitrary knowledge representation and learning methods. We evaluate the success of training new teammates in the well-known pursuit problem, where some of the current set of expert predators is being replaced by new ones with no *a priori* hunting knowledge. Experimental results demonstrate the robustness of our knowledge transfer scheme with a graceful performance degradation.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Experimentation, Performance

Keywords

Knowledge transfer, coordination, training

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

We have proposed an Agent Teaching Agent (ATA) framework to address the problem of transfer of concept description knowledge between a trainer and a trainee agent. A concept description is a boolean-valued function that classifies input examples into various discrete classifications of the target concept [5]. We assumed that the trainer agent does not have access to the internal knowledge representation of the trainee agent, but can evaluate the latter's concept recognition abilities by asking it to categorize selected exemplars and non-exemplars of the target concept. We focused on the incremental selection of training exemplars by the trainer to expedite the learning of the trainee agent. The ATA framework involves several iterations of training and test set presentations to the trainee, where the trainer chooses the training set of one iteration based on the mistakes made by the trainee on the test set in the previous iterations.

The goal of this paper is to show that the ATA framework can be used to transfer coordination knowledge to other agents, without information about the other agent's internal knowledge representation and learning algorithms. Such coordination knowledge transfer allows teams to recover from failure of some team members as well as deploy "clone" teams in the environment to reduce load on the current team of trainers. We assume that the trainer itself is a learner (a member of the current team), who learns its coordination knowledge from training instances provided by a domain expert, e.g., a coach. We further assume that this training set provided by the expert is no longer available, and the trainer must generate new exemplars from introspection to train new teammates.

To support our claims for the effectiveness of the iterative training process used in ATA, we also compare the performance of ATA-trained trainees with the performance of other trainee agents taught by the teacher agent in one iteration by presenting a single training set. Our results clearly show that the concept description generated by the trainee agent in the latter case is less effective in comparison to the concept description formed by the trainee agent trained using the ATA framework.

2. ATA FRAMEWORK

The trainer agent uses its learning module to acquire the target concept from its interaction with an environment, which may include an expert providing feedback. This learning process produces a target concept description in the knowledge representation used by the trainer agent. The

Train-Agent(Trainer,Trainee,Trainer-knowledge){
Select initial training set N_0 and initial testing set T_0 from Trainer-knowledge
$i \leftarrow 0$
repeat
  trainee agent trains on N_i
  $M_i \subset T_i$, examples misclassified by trainee after training on N_i
  $T_{i+1} \leftarrow T_i \cup newTestInstances(M_i)$
  $N_{i+1} \leftarrow N_i \cup newTrainingInstances(M_i)$
  $i \leftarrow i + 1$
until $|M_i| < threshold$

Figure 1: Algorithm for incremental training.

trainer agent also has a training module which interacts with the trainee and provides successive training and testing sets to train and evaluate the progress in learning of the trainee agent. The trainee learns its own concept description from the set of classified training examples provided by the trainer. It also classifies each of the unclassified test examples provided by the trainer and returns these classified instances to the trainer for evaluation.

We envisage an iterative training procedure in which alternatively the trainer selects a set of training and testing exemplars, the trainee trains using the training set and then classifies the testing set, the trainer observes errors made by the trainee in classifying the instances in the last testing set and accordingly generates the next training and testing sets. This iterative process converges when the error of the trainee falls below a given threshold. We present these iterative training steps in an algorithmic form in Figure 1.

3. DOMAIN AND TRAINING DATA

The domain used for our experimentation is the predator-prey pursuit problem [3] in a continuous world. This problem requires effective coordination by a group of agents to achieve a goal. The goal of the predator in this paper is to capture a prey which is moving in a straight line with half the speed of the predator. The predators who move simultaneously can see each other and the prey, but cannot communicate explicitly to coordinate their moves. To capture the prey, predators must surround it from four sides and be within a threshold distance of the prey. However if they come in close proximity of each other, a collision occurs and each of the colliding parties move a significant number of steps away from the collision point. To avoid such collisions, an expert predator moves a few steps orthogonal to the direction of the prey when it senses another predator in close proximity. The predators must capture the prey within a maximum number of steps, after which the game stops.

For our experimental setup, we have considered C4.5, a decision tree based algorithm [7] as the learning mechanism for the trainer and an instance based learning algorithm, IB2, as the learning mechanism for the trainee. IB2 uses an incremental learning algorithm that stores a subset of the training examples as its classification knowledge [1].

The experts provide the trainers with 17500 instances from various configurations to learn from. The large size of training is required to capture the complexities of coordinating four teammates in continuous space. To derive a rich set of training instances, the experts calculated all possible orientations of the four predators in the four quadrants centered at the prey. A configuration is a vector $< n1, n2, n3, n4 >$ where $n_i \in \{0, 4\}$ and $\sum_{i=1}^{4} n_i = 4$ and n_i is the number of predators in quadrant i with respect to the prey. Several initial predator-prey situations were generated for each such configuration, and the move of the experts from each such situation was noted. A training instance is created for each situation-move pairs by noting polar coordinates of the other predators and the prey, assuming this expert's positions as origin, were the classification of the instance is the move(right, left or ahead) selected by this expert at this situation. A typical training set data is a tuple $(r1,\ 1, r2,\ 2, r3,\ 3, rp,\ \mathrm{p}, M)$, where the different r and values signify the polar quadrants of the other three experts and the prey with respect to the expert who is teaching the trainer. M signifies the move that would be taken by the teacher expert under such an orientation. Once the trainers learn their coordination knowledge, they use the ATA framework to teach a new team member, the trainee.

4. RESULTS

Now we present the results from our experiments. The game is played on a 20×20 grid. We denote the various participants in our experiments by the following notations.

- Expert, E ,
- Trainer agent C: a decision tree based [7] learner,
- Trainee agent I: an instance based learner taught by a C agent in one iteration,
- Trainee agent A: an instance based learner taught by a C agent iteratively using the ATA framework.

Table 1 shows the performance of the different mixes of predators in the predator-prey pursuit game averaged over 1750 games. The first notable observation from Table 1 is the relative performances of the homogeneous groups of trainees who are taught either through the ATA framework or in one iteration. The relevant rows from Table 1 in this case are those with configurations AAAA and IIII. Clearly the AAAA configuration shows a higher capture rate and a faster capture to signify that the trainees who have undergone the learning process through the ATA framework have developed better coordination knowledge than those who have been taught in one iteration. Also the percentage capture rate for the AAAA configuration is fairly close to that of the CCCC configuration, consisting of all trainers, which shows that ATA framework has done a good job in teaching coordination knowledge to the new agents.

Now we shift our attention to heterogeneous predator team compositions. Interestingly, heterogeneous groups have smaller standard deviation compared to homogeneous groups, except for the group of experts. One reason for heterogeneous groups performing better than homogeneous groups can be that the differences in the knowledge of C and A agents can present useful complementarity that actually facilitate coordination. We also ran the Wilcoxon Matched-Pair Signed-Ranks test for pairs of groups where the team compositions differ by one trainee. The results show that the difference in step sizes to capture the prey is statistically significant for all such head-to-head comparisons.

Configurations	%*Capture*	Steps	
		Mean	*STD*
EEEE	100	14.77	6.63
CCCC	86.74	45.49	32.53
CCCA	100	17.85	8.19
CCAA	95.77	24.12	11.39
CAAA	89.54	32.69	18.59
AAAA	81.54	43.34	29.90
IIII	67.42	62.18	50.32

Table 1: Summary performance on an average of 1750 games

5. RELATED WORK

Multiagent learning has been an active area of research in the late 90s [8, 9]. The most relevant work involves one learner telling another agent what portions of the search space to ignore [6], a learner sharing experience [2], problem-solving traces or even learned policies [10] with another concurrent learner. A recent paper discusses a method for two agent to mutually define a concept [11]. Though emphasis is placed on instance selection, in contrast to the current work, there is no pre-existing concept, and hence the learners are peers rather than a trainer-trainee pair. Tan's work [10] of an expert sharing effective problem solving traces with a novice agent and Clouse's work of a trainer suggesting actions to take [4] are perhaps the closest in motivation to the current work, but the iterative nature of teaching, at the heart of the ATA framework, is not addressed by them.

6. CONCLUSIONS

We developed a generic approach for knowledge transfer between experts and novice agents where the knowledge representation and learning algorithms of the experts and the trainees are fundamentally different. We present arguments for the generality of our approach for learning coordination knowledge and evaluate its effectiveness in the pursuit problem with a decision tree learner used as the learning module of the trainer and an instance based learner used as the learning module of the trainee. Initial results are encouraging and demonstrates effective transfer of coordination. The learners trained through the ATA framework exhibit consistently higher performance compared to the learners trained with a single presentation of training data. A team composed entirely of new trainees perform close to the team of trainers. More importantly, heterogeneous teams of trainers and trainees outperform homogeneous teams of either the trainers or the trainees. The results show effective transfer of coordination knowledge to new teammates by existing players using the ATA framework.

We plan to run experiments on a wider set of problem instances with variants of the pursuit problem. We plan to use a support-vector machine based agent as trainer and trainee agents in conjunction with the current instance-based agent and decision-tree based agents. We also plan to explore possible combinations of active learning and ATA approaches to train new teammates. In the future, we would like to run our ATA framework on predators that collude between themselves to evolve a strategy for capturing the prey in optimal number of steps as well as for transferring coordination knowledge in other challenging multiagent problems.

Acknowledgments: This work has been supported in part by an NSF award IIS-0209208 and an NASA EPSCoR RIG.

7. REFERENCES

[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1), 1991.

[2] B. Banerjee, S. Debnath, and S. Sen. Combining multiple perspectives. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 33–40, San Francisco: CA, 2000. Morgan Kaufmann.

[3] M. Benda, V. Jagannathan, and R. Dodhiawalla. On optimal cooperation of knowledge sources. Technical Report BCS-G2010-28, Boeing AI Center, Boeing Computer Services, Bellevue, WA, Aug. 1985.

[4] J. A. Clouse. Learning from an automated training agent. ML-95 Workshop on Agents that Learn From Other Agents, 1995.

[5] T. Mitchell. *Machine Learning*. WCB McGraw-Hill, Boston, MA, 1997.

[6] F. J. Provost and D. N. Hennessy. Scaling up: Distributed machine learning with cooperation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 74–79, Menlo Park, CA, 1996. AAAI Press.

[7] R. J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.

[8] S. Sen and G. Weisß. Learning in multiagent systems. In G. Weiß, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, chapter 6, pages 259–298. MIT Press, 1999.

[9] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3), July 2000.

[10] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, June 1993.

[11] J. Wang and L. Gasser. Mutual online concept learning for multiple agents. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 362–369, New York: NY, 2002. ACM Press.

Learning and evolution

Efficient Agents for Cliff-Edge Environments with a Large Set of Decision Options

Ron Katz, Sarit Kraus
Department of Computer Science and The Gonda Brain Research Center
Bar-Ilan University, Ramat-Gan 52900, Israel, sarit@cs.biu.ac.il

ABSTRACT

This paper proposes an efficient agent for competing in Cliff Edge (CE) environments, such as sealed-bid auctions, dynamic pricing and the ultimatum game. The agent competes in one-shot CE interactions repeatedly, each time against a different human opponent, and its performance is evaluated based on all the interactions in which it participates. The agent, which learns the general pattern of the population's behavior, does not apply any examples of previous interactions in the environment, neither of other competitors nor its own. We propose a generic approach which competes in different CE environments under the same configuration, with no knowledge about the specific rules of each environment. The underlying mechanism of the proposed agent is a new meta-algorithm, Deviated Virtual Learning (DVL), which extends existing methods to efficiently cope with environments comprising a large number of optional decisions at each decision point. Experiments comparing the performance of the proposed algorithm with algorithms taken from the literature, as well as another intuitive meta-algorithm, reveal a significant superiority of the former in average payoff and stability. In addition, the agent performed better than human competitors executing the same task.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms,Economics,Human Factors

Keywords

Reinforcement Learning, Opponent Modeling, Sealed-bid Auctions, Ultimatum Game

1. INTRODUCTION

In many games and economic interactions agents are challenged by the strive of maximizing their profits while preventing the entire deal from falling through. Consider, for example, an agent in an online Web-store that needs to determine the price of a good: Increasing the price would increase its profits as long as the price it charges is not higher than the price the consumer would be willing to pay. A little greedier price would make it lose the whole deal [11]. Similarly, a bidder in a sealed-bid first-price auction [3] attempts to bid the exact amount that is slightly higher than others' biddings. The same goes for the proposer in the ultimatum game (UG) [9], who tries to offer the exact amount equal to the acceptance threshold of his opponent. Higher offers would decrease self profits, while lower offers would yield no profits at all. This situation is somewhat similar to a person standing on the edge of a cliff, trying to see the panoramic view. The closer he gets to edge, the better he can see the view. However, one step too many may cause him to fall off the cliff. Hence, this set of interactions and games are referred to as Cliff-Edge (CE) interactions.

We focus on one-shot CE interactions which are repeatedly competed with different opponents. Such repeated interactions occur, for example in the important application of dynamic pricing, i.e. periodically changing prices according to the reactions of consumers that change over time. This task becomes even more common in the E-commerce world, where communication with the consumers is restricted [11, 4]. Sealed-bid auctions repeated periodically with the same goods are very popular, especially via the internet (see [3, 5]). Similarly, the sequential version of UG with different opponents is under thorough investigation [13, 2, 1]. Thus, designing efficient agents for repeated CE environments is very important for the everyday commercial world.

Moreover, our research deals mainly with interactions between humans and automated agents. This differs from conventional environments of pure computerized agents, where most competitors have large amounts of computational power and act as rational negotiators. In particular, when interacting with humans, the theoretical equilibrium strategy is not necessarily the optimal strategy since human subjects, inherently rationally bounded as well as computationally restricted, commonly do not behave according to perfect equilibrium. Thus, this study joins the important growing research field of interactions between humans and agents (e.g. [6, 12]).

Previous studies of automated agents interacting with humans have been concerned mainly with agents who interact repeatedly with a single human opponent (e.g. [12]). The general approach for such a problem is to model the opponent's behavior, and to adjust a strategy which optimally reacts to the opponent's predicted reaction. In the current study, however, a specific opponent modeling is not relevant, since the agent interacts with each opponent only once. Alternatively, agents who interact with a series of different people, should learn the general pattern of the opponent population.

A possible approach to efficiently adjust to an unknown opponent population, is to simply observe previous samples of interac-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

tions within this population. Several studies have successfully used this approach in order to develop automated agents that can successfully compete with human opponents (e.g. [6]). This approach however, requires a large number of historical examples of human behavior. Moreover, the option of using a historical database is not always possible. For example, in a sealed-bid auction, a bidder usually cannot obtain information about the bids which were offered in previous similar auctions. Therefore, we propose a mechanism to develop an agent which does not depend on examples of previous interactions. Our agent learns while interacting with others, and its performance is evaluated from the first interaction.

Moreover, this study focuses on settings where in each interaction the agent is required to choose an action from a large set of possible options. Most of the previous studies on CE environments focus on settings in which an agent has to choose from at most 10 options during each interaction [13, 11, 1]. A large scale of option settings reflects commercial environments more realistically, where the number of decision options is commonly much higher than 10 (e.g. [4, 5]). Unfortunately, as we show in this study, the performance of existing algorithms choosing from a large set of options such as 100 options is insufficient. The existence of a large set of options demands a construction of a fast and efficient screening procedure. Furthermore, the importance of quickness is emphasized, since we examine short term interactions with only several dozens of opponents.

In this paper we propose a meta-algorithm which extends existing algorithms to efficiently compete in multi-interaction CE environments that consist of a large set of decision options. The general idea of the proposed algorithm is to reinforce options which are slightly greedier than the optimal option currently considered. Thus, in the conflict between exploration and exploitation of current information, we compromise by a small deviation from the current optimal option. A broad description of this method, named Deviated Virtual Learning (DVL), is presented below.

Several approaches for automatically competing in CE environments have been previously proposed [17, 15, 11]. Those approaches are broadly described below. In this paper, we experimentally show that the extension of the DVL meta-algorithm over the basic Reinforcement Learning (RL) algorithm [14], performs significantly better than previous algorithms when competing against human opponents in various CE environments. Moreover, the DVL extension of the RL algorithm (DVRL) is shown to perform better than another intuitive meta-algorithm named Segment-Based, which appears here for the first time. We also found that the DVRL algorithm performs better than human competitors who face the same task.

It is noteworthy that the algorithms discussed here are all generic and suitable for various CE environments, without knowing the specific rules of each environment. Thus, the agent is not aware of whether it is competing in an auction or in the UG. Moreover, in all the simulations presented hereinafter, each algorithm was run with a fixed configuration setting that was not changed from environment to environment.

In the next section, we formally describe the CE environment. In section 3 we present the proposed DVL meta-algorithm. In section 4 we survey other relevant algorithms which appear in previous studies, and present the Segment-based meta-algorithm. We then compare the performance of the DVRL algorithm with the performance of other relevant algorithms, and analyze the results. We conclude and present directions for future work in section 5.

2. THE CE ENVIRONMENTS

The general pattern of one-shot CE interactions considers an agent required to choose an offer *i*, which is an integer $0 \leq i \leq N$, where *N* is the maximal optional choice. Then a positive reward *r* corresponding to the offer *i* is determined, depending on whether the offer passed a certain acceptance threshold set by the opponent (in auctions we refer to the acceptance threshold of the auctioneer, which is the second highest bid proposed). Since the CE set includes various environments, we detail the model of each of the three environments mentioned above: auction, UG and pricing. Other CE environments can be similarly modeled.

- In the sealed-bid first-price auction model an amount *N* is auctioned [1]. The agent is required to choose its bid which is an integer $i, 0 \leq i \leq N$. Given the bid, a reward *r* is determined according to the highest bid τ, among all the bidders (two or more) in the current auction. If $\tau \leq i, r = N - i$ (the amount gained subtracted by the bid amount), otherwise r=0 [2]. In the all-pay version, where all the bidders must pay their bids, even if they have not won the auction [10]: if $\tau \leq i, r = N - i$ otherwise $r = -i$.

- In the UG, the agent should divide an amount of *N* between itself and its opponent. It is required to choose an integer $i, 0 \leq i \leq N$, which is the amount proposed to the opponent. The reward *r* is determined by the opponent's acceptance threshold τ . If $\tau \leq i, r = N - i$ otherwise *r*=0.

- In the pricing task, an agent is required to select a price *i* for an item being sold to a certain consumer (*i* actually represents the profit of the seller, beyond the cost price). In our model, *i* is an integer $0 \leq i \leq N$, where *N* is the maximal reasonable price of the item. After the decision about the price, a reward, *r*, is determined according to the current consumer's acceptance threshold τ. If $\tau \geq i, r = i$ otherwise *r*=0.

In this paper, which considers environments with a large set of decision options, we set *N* to be 100.

To demonstrate the challenge of a competitor in CE environments, let *S(i)* be the corresponding reward of offer *i* if the move succeeds (if the offer is accepted by the consumer in the price setting task, higher than the other bids in the case of an auction, or accepted by the responder in the UG), let *F(i)* be the corresponding reward of offer *i* if the move fails, and let *P(i)* be the probability that an offer *i* will succeed. An efficient agent interacting in multi-interaction CE environments with different opponents must find the optimal offer *i* for the opponent population that maximizes the accumulative utility function:

$$U(i) = S(i)*P(i) + F(i)*(1-P(i))$$

Obviously, there is a trade-off between *S* and *P*: choosing an offer which increases the expected reward, *S*, decreases the probability of success, *P*, and vice versa. Similarly, in the all-pay auction, which is the only case where *F(i)* values are negative and are not equal to zero, choosing an offer which decreases the expected loss, *F*, increases the probability of failure, *(1-P)*.

The naive approach for such problems would be to apply the Monte-Carlo (MC) method (see [14] chapter 5). According to this method each optional proposal or bid amount is examined many times, in a serial or random order, and finally (the later the better) a model for success probabilities (P) distribution of the opponent

[1] In order to evade considerations of value estimations [5] the item auctioned is an amount of money.

[2] For reasons of compatibility with other environment models, equal bid amounts yield a gain for the learning agent.

population is constructed. However, the learning process in this approach is extremely time consuming. Moreover, a large number of actions made by the MC method are totally irrational. An efficient agent should quickly realize, for example, that most responders in the UG would accept a share of 50 percent, and thus a proposal of more than 50 percent, is not advised. Therefore, we would rather use a less naive attitude, in which directed exploration is conducted.

Nevertheless, for a very large number of interactions, the MC method may provide a more precise model of the real success probabilities distribution in the population, than algorithms which use directed exploration. However, in this study we are more concerned with finding fast and efficient approaches for the first several dozens of opponents. Handling the initial interactions, when almost nothing is known about the environment is obviously the most difficult stage. Moreover, considering a larger number of interactions, the severity of the accuracy problem caused by directed exploration, could be reduced by combining a sophisticated algorithm with a naive one. A method such as MC, may replace a more sophisticated algorithm in the progressive interactions, based on previous interactions conducted by the latter.

3. THE PROPOSED APPROACH

In this section we present a detailed description of the proposed meta-algorithm for competing in CE environments. As a meta-algorithm, it extends basic algorithms and improves their performance. We assume that the basic algorithms select their actions according to an evaluation of the expected utility that each decision option would yield, provided it is chosen. The evaluation of the expected utilities is determined according to the results of previous interactions. Hereinafter, we will refer to the database which stores the evaluations of each decision option as the Q-vector [3]. Each algorithm includes its own UPDATE procedure, which determines how to update the Q-vector after observing the successfulness of the latest action. It should be noted that the Q-values stored in those Q-vectors, do not necessarily express the real evaluations of the expected utility, but rather the relative profitability of choosing each option. In addition, we assume each algorithm includes its own SELECT procedure, which determines how to select the next action (apparently according to the current Q-vector). The SELECT procedure moves between exploiting the evaluations stored in the current Q-vector and exploring options with lower current Q-values, in order to improve the correctness of the Q-vector's evaluations. The first action is chosen randomly in all the algorithms presented in this paper.

A reasonable approach for the UPDATE procedure in CE environments is *Virtual Learning* (VL) [16]. According to the VL principle, the agent in UG, for example, treats all the offers (not actually proposed) higher than a successful offer as if they were (virtually) successfully proposed as well. Similarly, it considers all the offers lower than an unsuccessful offer as unsuccessfully proposed. The rationale behind this principle is that the higher the amount proposed to the opponent - the higher the probability of the proposal being accepted [4]. Thus, we should reinforce the Q-values of all the offers higher than the actual offer, provided it was successful, and we should reduce the Q-values of all the offers lower than the actual offer, provided it was unsuccessful. A crucial problem with this approach is an *information asymmetry*[16], which causes an intensified reinforcement of Q-values higher than the optimal option's Q-value. The reason for this phenomenon, as demonstrated in [16], is that whenever the optimal offer is successfully chosen, offers which are higher than the optimal offer are reinforced as well. However, when the optimal offer fails it is punished, while the offers just above suffer no penalty, giving them the appearance of having a much higher probability of success.

In order to overcome this information asymmetry problem, we propose the *Deviated Virtual Learning* principle. This principle can be implemented in various basic algorithms, thus producing a general meta-algorithm, as presented in **Algorithm 1**. According to the DVL principle, we should deviate from the strict rationale underlying the VL principle, and extend the range of offers updated after each interaction. Thus, after a successful interaction, we would raise the Q-values of all the offers higher than the actual offer, as well as a few offers under the actual offer, as described in line 8 in **Algorithm 1**. Similarly, after an offer has failed, we would reduce the Q-values of all the offers lower than the actual offer, as well as a few offers above the actual offer, as described in line 6 [5].

Algorithm 1 THE DVL META-ALGORITHM

Notation: α, β are two integers $0 \leq \alpha < \beta \ll$ N, (where N is the upper bound of possible offers), which denote the deviation rate. The values α and β gradually decrease during the learning process.

```
1: For j=0 to N, Do Q(j)=1
2: For each interaction, Do
3:    Select offer i according to a SELECT procedure
4:    Observing opponent's move, calculate reward
5:    if offer i has failed then
6:       For j=0 to (i + α), Do
            reduce Q(j) according to an UPDATE procedure
7:    else
8:       For j=(i - β) to N, Do raise Q(j) according to an UPDATE procedure
```

As mentioned above, the DVL meta-algorithm can extend any basic algorithm, according to its SELECT and UPDATE procedures. However, we claim that the best algorithm for human environments, as we will demonstrate, is the DVL extension of the Reinforcement Learning [14] algorithm (DVRL), as presented in **Algorithm 2**. The SELECT procedure of the RL version we have used, was to select the offer with the current maximal Q-value (line 3). The UPDATE procedure was to divide the accumulated reward of each offer by the number of previous interactions where offer i was actually or virtually (according to the DVL principle) proposed (lines 6 and 8).

It is important to note that in this study DVRL was run in all the environments with the same configuration of α and β as detailed in **Algorithm 2** (lines 1 and 10).

There are two additional motivations for the deviation principle underlying the DVL approach. First, from the statistical aspect it is not necessarily a mistake to consider an offer which is slightly lower than a successful offer, as successful as well. Using UG terminology, it is quite reasonable to assume that if proposal *i* was accepted (rejected) by an opponent, she would have also accepted (rejected) a slightly lower (higher) proposal. The chance that the proposal would exactly hit the acceptance threshold of the opponent is low, especially for a large set of options. Second, and even

[3] It is noteworthy that despite the name, this paper does not concern the Q-learning algorithm, which handles situations in which the state may be changed at each interaction (see [14] chapter 6.5).

[4] The same rationale stands behind the auctions environment. However, in the pricing environment the opposite is true: all the offers lower than a successful offer are considered successful, since the lower the price of an item the higher the probability of it being purchased by the consumer.

[5] It is worth noting that all the VL-based algorithms in this paper are presented in a version suitable for auctions and for the UG. In the pricing environment, where increasing the price decreases the acceptance probability, the update ranges in lines 6 and 8 should be swapped.

Algorithm 2 THE DVRL ALGORITHM

Notation: n(j) denotes the number of previous interactions where offer j was actually or virtually proposed, s(j) is the corresponding reward for a successful offer j and f(j) is the corresponding reward for offer j when it fails. α, β denote the deviation rate.
We present the configuration used in our environment. However, the initial values of parameters N, α and β (line 1) as well as the update function of α and β (line 10) can be adjusted to other environments as well.

1: N=100, α=10, β=15, t=0 For j=0 to N, **Do** Q(j)=1, n(j)=0
2: For each interaction t, **Do**
3: offer i=$\arg\max_j Q(j)$
4: Observing opponent's move, calculate reward
5: **if** offer i has failed **then**
6: For j=0 to (i + α), **Do** n(j)=n(j)+1, $Q(j) = \frac{Q(j)(n(j)-1)+f(j)}{n(j)}$
7: **else**
8: For j=(i - β) to N, **Do** n(j)=n(j)+1, $Q(j) = \frac{Q(j)(n(j)-1)+s(j)}{n(j)}$
9: t = t+1
10: $\alpha = \frac{10}{\lfloor t/10 \rfloor + 1}$ $\beta = \frac{15}{\lfloor t/10 \rfloor + 1}$

more important, the deviation principle actually outlines a direction of the optimal solution searching, rather than the random trial-and-error approach that underlies conventional methods, such as RL. A DVRL agent that offered 70% of the cake (N) to his UG opponent in the first interaction, and his offer was accepted, would offer 55% (according to our configuration of β) in the next interaction. The agent continues to decrease the offer until it is rejected. A similar pattern of this directed searching was observed in human learning, and is compatible with the directional learning theory (see [8]). According to this theory, if an offer *i* is rejected at interaction *t*, then at interaction *t+1* the proposer will offer his opponent a higher offer, while if offer *i* is accepted at interaction *t*, then at interaction *t+1* the offer will be decreased. This simple method, which was simulated by [2] without yielding any impressive performance, lacks all the data experienced before the previous interaction.[6] In our algorithm, on the other hand, all the previous interactions are taken into account, and the model comes closer to the real distribution of the opponents' population during the learning process. This is the reason for the gradual decrease of α and β values. Another advantage of our approach over the naive directional learning is achieved by setting $\alpha < \beta$, asymmetrically. This property enables fast converging to the optimal solution, by speedily decreasing the proposal amount, and slowly increasing it to the optimal amount.

4. EXPERIMENTAL DESIGN AND ANALYSIS

In this section we examine the performance of the proposed DVRL algorithm competing in various CE environments with human opponents, and compare it to other algorithms' performance. A broad survey of the algorithms examined is presented, followed by a description of the experiment process and the results. After analyzing the results, we compare the performance of DVRL to human negotiators' performance. At the end of this section, we briefly discuss the situation of a small set of decision options' environments.

4.1 Comparative Algorithms description

[6]Simulations of the directional learning method in our environment also does not yield an impressive performance. A similar approach, namely the Derivative-Following strategy, was proposed for the dynamic pricing problem [4]. However, each round (one commerce day) was considered as a set of several interactions with a number of opponents, which is totally summed up by thousands of interactions.

In this section we survey several algorithms which can be used in repeated CE environments. Subsections 4.1.1-4.1.4 describe previously published algorithms, while in 4.1.5 we present a new intuitive meta-algorithm for a large set of options environments.

4.1.1 Roth and Erev's Algorithm (R&E)

Concerning the UG, Roth and Erev [13] designed a human proposer model based upon basic RL [14]. The general idea of the RL method is that each offer is chosen with a probability according to its current Q-value - the larger an offer's Q-value, the higher the probability it will be chosen. After each interaction, provided that the latest offer succeeded, its probability to be chosen again is increased by increasing its Q-value by the reward this offer yields. Roth and Erev showed that despite the fact that UG is characterized by having a significant gap between empirical human behavior and subgame perfect equilibrium predictions, the RL algorithm with a few additional modifications can successfully model typical human empirical behavior in UG. The most noticeable modification, based on the Persistent Local Environment principle, uses a "generalization" parameter which prevents the probability of an option from going to zero if it is adjacent to a successful option. The criterion of adjacency was configured in [13] to be +-1, and thus, if an offer of 4 out of 10 is accepted by the opponent, the agent would reinforce the Q-values of offers 3,4 and 5. Their assumption, which is valid in all of the CE environments, was that an adjacent option of an optimal solution cannot be an extremely bad solution. The detailed algorithm can be found in [13].

4.1.2 Modified version of Zhong, Wu and Kimbrough's (ZWK) RL algorithm

Zhong et al.[17] developed an agent which played UG against various simulated opponents, using a version of RL which is different from R&E's. The UG agent designed by these authors used an ϵ-greedy selection procedure, in which the agent selects, with a probability of (1-ϵ), the offer with the current maximal Q-value, and with a probability of ϵ it uniformly selects an offer for exploration purposes. The advantage of this selection procedure over a selection according to the Q-value's distribution as in R&E, is that it mainly chooses the best option. The disadvantage is that the exploration is not weighted by the estimation of expected utility considerations, as in R&E, but is rather completely random. In order to decrease the effect of this disadvantage, we have modified the selection procedure in our experiment, to take into account the generalization principle of R&E. Thus, when not selecting the offer with the current maximal Q-value, our ZWK algorithm selects an offer giving higher priority to offers which are adjacent to the offer with the current maximal Q-value (**Algorithm 3**, state 4), rather than selecting them uniformly. In addition, since the exploration is much more important in the initial interactions, we decreased the probability for random selection gradually during the learning process. These two modifications were examined and were found to be significantly efficient in our environment. The UPDATE procedure in ZWK's algorithm was also different from that of R&E, by taking into account the previous failures of offer *i*, rather than merely the positive previous rewards (line 6).

4.1.3 Virtual Reinforcement learning (VRL)

Despite the information asymmetry in VL found by [16], Todd and Borgers [15], have shown that the addition of a virtual learning mechanism to the RL algorithm when playing UG, may be very efficient. However, the efficiency is dependent on several configurations of the learning procedure, such as the selection procedure (R&E's or ZWK's) and the Q-values update procedure (reinforce-

Algorithm 3 THE ZWK ALGORITHM

Notation: ϵ denotes the probability for random selection, γ is the probability for selecting an offer which is adjacent to the offer with the current maximal Q-value ($\epsilon + \gamma < 1$), and δ determines the adjacency range ($\delta \ll$ N). The values ϵ and γ are gradually decreased during the learning process.

1: For j=0 to N, **Do** Q(j)=1, n(j)=0
2: For each interaction, **Do**
3: m = $\arg\max_j Q(j)$
4: With a probability of (1- ϵ- γ), offer i = m
With a probability of γ, select offer i uniformly, for i's such that max(0, m - δ) $\leq$ i $\leq$ min(N, m + δ)
With a probability of ϵ, select offer i uniformly, for i's such that 0 $\leq$ i $\leq$ N
5: Observing opponent's move, calculate reward r
6: n(i)=n(i)+1, Q(i) = $\frac{Q(i)(n(i)-1)+r}{n(i)}$

ment of a successful offer by a unit payoff, or by the real amount gained, or rather by the virtual amounts the agent would keep if he hypothetically offered different proposals).

After examining the performance of VRL with all the optional configurations suggested by Todd and Borgers, in **Algorithm 4** we present the version that yields the best performance in our environment. This algorithm includes the same SELECT procedure as in ZWK.

Algorithm 4 THE VRL ALGORITHM

Notation: s(j) denotes the corresponding reward for a successful offer j and f(j) is the corresponding reward for offer j when it fails.

1-5: As in **Algorithm 3**
6: **if** offer i has failed **then**
7: For j=0 to i, **Do** n(j)=n(j)+1, $Q(j) = \frac{Q(j)(n(j)-1)+f(j)}{n(j)}$
8: **else**
9: For j=i to N, **Do** n(j)=n(j)+1, $Q(j) = \frac{Q(j)(n(j)-1)+s(j)}{n(j)}$

4.1.4 Gittins' index strategy

Some researchers have observed CE environments as a special case of the multi-armed bandit problem [11, 1]. In this problem every period a decision maker has to decide on which one of *n* slot machines he wants to play, given each machine has different gain probabilities (unknown a priori). These researchers used Gittins' index strategy [7], an optimal strategy for the multi-armed bandit problem, to model the behavior of proposers in UG and of sellers in the pricing problem, considering each optional proposal or price as an arm. The Gittins' strategy was applied to CE environments by multiplying the Bernoulli reward process index value of each option (see [7] p. 222) by its reward. However, these CE problems are different from the classical multi-armed bandit problem since the arms are independent, while in CE environments the success probability of an offer is gradually influenced by the size of the offer. This difference may become more critical when there is a large number of options rather than the 10 optional arms problems considered in these papers. Thus, Brenner and Vriend [2] tried to adjust to the interdependent arms in UG by adding the VL mechanism to the Gittins' strategy. In the following section we present the performance of both the basic Gittins' strategy (GS) (as in [11, 1]) and the virtual Gittins' strategy (VG) (as in [2]) in our environment.

4.1.5 Segment-based meta-algorithm

This algorithm, suggested here for the first time is an intuitive meta-algorithm which can extend any of the former basic methods. This algorithm is specifically designed for environments with a large number of decision options. The idea of the segment-based approach is to divide all the options into segments, and to activate the learning method in the initial interactions on these segments, rather than on specific discrete options (**Algorithm 5**, state 3). After several interactions the resolution can be increased by focusing on smaller segments. This process continues gradually towards the last segmentation hierarchy with the smallest segments, i.e. specific discrete options, where final fine tuning is performed (end of state 3). This algorithm is based on the assumption of locality, i.e. adjacent options yield similar average profits. The advantage of this approach is the gradual filtering of the optimal solution. A common problem in conventional RL, for instance, is that an option might have a high Q-value in a progressive stage of the learning process, although it is far from the optimal option. With the proposed meta-algorithm, this situation is prevented already in the preliminary stages of learning, by weakening the Q-value of the entire range surrounding this non-beneficial solution.

Algorithm 5 THE SEGMENT-BASED META-ALGORITHM

Notation: H denotes the number of segmentation hierarchies configured by the user according to the size of the options set (setting this parameter, the order of magnitude of the expected number of interactions must be considered), S_h is the size of each segment in the hth segmentation hierarchy, Q_h are the Q-values vectors corresponding to the hth segmentation hierarchy, and T_h is the serial number of the last interaction of the hth segmentation hierarchy.

1: For h=1 to H, **Do** For j=0 to $\lceil \frac{N}{S_h} \rceil$, **Do** Q_h(j)=1,
2: For each interaction, **Do**
3: For interactions $0 - T_1$: c = 1
Select segment j from the 1st segmentation hierarchy according to a SELECT procedure, using vector Q_1
Select offer i uniformly from segment j
For interactions $T_{x-1} - T_x$: c = x
Select segment j from the xth segmentation hierarchy according to a SELECT procedure, using vector Q_x
Select offer i uniformly from segment j
...
For interactions $T_{H-1} - T_H$: c = H
Select offer i according to a SELECT procedure, using vector Q_H
4: Observing opponent's move, calculate reward
5: For h=c to H, **Do** update Q_h according to an UPDATE procedure

4.2 Experiment Description

In order to evaluate the performance of the proposed algorithm, and to compare it with other algorithms, we experimentally examined agents interacting with human opponents in 4 different CE environments, as follows:
1. UG, where the players had to divide 100 new Israeli Shekels (NIS, where 1 U.S. \$ $\approx$ 4.5 NIS), i.e. N=100.
2. A variant of UG in which the responder determines only whether the proposer gets his share. The responder himself always receives his own allocation [9]. In this version the responder's decision is influenced by fairness and vindictiveness considerations, rather than profitability as in the original UG version. Here also N=100.
3. Sealed-bid 2-bidders auction. The auction was for 100 NIS.
4. All-pay sealed-bid auction for 100 NIS, where all the bidders must pay their biddings[10]. In this version risk taking considerations is added to the bidder's decision.

The examination of different environments which activate different personality characteristics of human opponents, guarantees generality of the results. Dynamic pricing was not experimentally examined, since no specific item can be unquestionably desired by

all of the participants. The following algorithms were examined:
1. R&E, as described in section 4.1.1.
2. ZWK, as specified in section 4.1.2.
3. Virtual RL (VRL), as specified in section 4.1.3.
4. Basic Gittins' index strategy (GS), as described in section 4.1.4.
5. Virtual Gittins' index strategy (VG), as described in 4.1.4.
6. Segment-based extension of the ZWK version of RL (SZWK).
7. Segment-based extension of VRL. (SVRL).
8. DVRL, as specified in section 3.
9. Segment-based extension of VG (SVG).
10. Deviated virtual extension of GS (DVG).

The first five algorithms were suggested by others and adapted to our environment as detailed above, while the last five algorithms are based on the two meta-algorithms we developed [7].

It is important to mention that each algorithm was run with fixed parameter configurations (such as the ϵ value's decreasing rate, and deviation rates α, β in the DVL algorithms) for all the different environments. Though specific configurations for each environment could yield better performance, we preferred the generality of the algorithms over a variety of CE environments, ensuring that no environment specific characterization would be used.

In the first experiment human participants were used as responders in the UG games, and as bidders in the auctions. Each person participated once in each of the 4 environments. The automated agents, on the other hand, interacted serially against different human opponents. Evaluating agents that were designed for human involved environments by examining their performance with real human data is necessary. As mentioned above, human competitors do not obey subgame perfect equilibrium, and thus their behavior cannot be a priori simulated. Additionally, human behavior cannot be accurately statistically modeled, especially in small populations as in this case. Another benefit from empirical experiments is the ability to provide a concrete algorithm, namely DVRL (with a concrete configuration of parameters), which successfully competed against human opponents. Thus, this agent can be immediately applied in real applications, at least as a starting point.

In the first stage of the experiment we surveyed 34 students (17 males and 17 females) participating in four CE environments. The participants were students at Bar Ilan University, aged 20-28 (average 23.75), who were not experts in negotiation strategies nor in economic theories directly relevant to the experiment (e.g. game theory, decision theory). Each of the participants, who sat in an isolated room at a computer work-station, interacted in the 4 CE environments, one after the other.

In the auctions the participants were required to propose a bid, which could be any integer from 0 to 100 NIS. The winner gained a virtual 100 NIS. In the ultimatum games, where 100 NIS was the full amount to be shared, the participants were required to respond to various proposals, which were artificially provided in order to determine the minimal acceptance amount of the players [8](though the participants were told that the proposals were provided by other people). Data from a few participants who were inconsistent in their decisions were excluded. At the end of the experiment, each participant was paid between 15 to 30 NIS, proportionally to her earnings in the interactions in which she participated.

After extracting the bids from people in the auctions, as well as their acceptance thresholds in the ultimatum games, we constructed sets of opponents' reactions for each environment. The sizes of the sets were 34 for the auctions, 28 for the UG and 27 for the UG variant, due to the exclusion of inconsistent players. At this stage we examined the performance of each of the ten algorithms detailed above, which were run serially against the sets of opponents' thresholds. Thus, each algorithm had one interaction with each of the human opponents in each of the four environments, without knowing in advance the number of interactions. Since there is importance to the order of the opponents, we constructed 200 random permutations of the human decisions series, for each environment, and compared the average payoffs of the different algorithms for each permutation. In addition, these algorithms were run against an artificial series of 50 auction opponents, constructed randomly according to a normal distribution N(71,10). In this manner, we wanted to examine the performance of the algorithms with a theoretical population which distributes normally, though there is no evidence of such a distribution in any CE environment.

4.3 Experimental results

Table 1 presents the average payoffs for each algorithm. The average payoffs integrate the data of all the 200 permutations. Due to the fact that the algorithms' random factors cause variation in the results we ran each algorithm repeatedly for 50 times for each permutation,. The standard deviations of the 50 results of each algorithm with each permutation are also presented in order to measure the stability and self-consistence of each algorithm.

4.3.1 Payoff analysis

The results show that the DVRL algorithm was almost always the most profitable algorithm. A non-parametric Friedman test revealed significant differences in the ranking of the algorithms ($p < 0.001$). Further pairwise Wilcoxon tests showed that the most efficient algorithm was significantly the DVRL algorithm, except for the auction, where DVG was significantly better though only slightly, and for the normal distribution auction, where no significant difference was found between DVRL and SVG.

Moreover, the extension of both DVL and Segment-based meta-algorithms almost always improved the payoff. Pairwise Wilcoxon tests revealed that the algorithms SVRL, DVRL, SVG and DVG always achieved higher payoffs than the first five algorithms ($p < 0.001$). The SVRL method always achieved the lowest payoff of the leading quartet (SVRL, DVRL, SVG and DVG) ($p < 0.001$), except for the UG, where no significant difference was found between SVRL and DVG. The internal ranking of SVG and DVG was inconsistent, with no significant differences in the all-pay auction and the UG variant. As for the SZWK algorithm, we found that its performance was always significantly higher ($p < 0.001$) than the basic ZWK's algorithm, except for the all-pay auction, where no significant difference was found between these methods.

Virtual learning was revealed as an efficient method, as can be concluded by the general superiority of VRL over ZWK (except for the UG variant), of VG over GS (except for the UG variant, and for UG, where no difference was found) and of SVRL over SZWK (except for the auction). A non-virtual version of SVG (a segment based extension of GS), which is not presented because of space considerations, was similarly overpowered by SVG.

4.3.2 Results explanations

In order to understand the origin of the advantage of DVRL over DVG, SVRL and over SVG, we should observe **Figure 1**. This plot describes the average payoff of each algorithm - SVRL, DVRL, DVG and SVG - (the y-axis), during the interactions (the x-axis).

[7] A combination of the 2 meta-algorithms is not presented in this paper since its performance was experimentally worse than DVRL's. Moreover, the implementation of the combined agent is much more complicated than DVRL's, since it includes a large number of configurational parameters.

[8] This method of responders' behavior examination is widely accepted in UG literature [1].

Environment		R&E	ZWK	VRL	GS	VG	SZWK	SVRL	DVRL	SVG	DVG
UG	payoff	35.55	39	41.3	37.43	37.44	40.13	41.59	**43.28**	42.35	41.79
	SD	5.56	4.2	4.03	4.02	3.90	3.86	3.47	2.3	2.17	2.12
UG variant	payoff	28.93	33.99	32.41	31.88	30.44	35.23	36.37	**39.24**	37.42	37.23
	SD	5.86	5.01	5.6	5.31	5.96	4.85	4.19	1.99	2.68	2.30
Auction	payoff	13.35	15.95	16.18	15.89	16.52	17.35	17.05	19.06	17.69	**19.37**
	SD	3.21	3.81	3.64	4.12	3.57	3.31	2.87	1.47	3.15	1.69
All-pay auction	payoff	-3.27	-1.88	-1.09	-2.14	-1.11	-1.99	-0.26	**2.91**	1.65	1.5
	SD	6.07	5.2	5.14	5.22	4.43	6.12	5.14	2.6	3.08	2.81
Normal distribution auction	payoff	7.83	11.66	12.39	11.08	12.63	13.03	14.14	**14.88**	14.82	14.74
	SD	2.12	3.18	2.94	4.16	2.87	2.87	1.72	0.57	1.09	0.84

Table 1: Average payoff and standard deviation of various algorithms against human opponents

The plot, which enables easier tracking of the learning process, presents only the data of the all-pay auction. However, the data of the other environments were quite similar, thus we can refer to the all-pay auction as a representative case. The left side of the plot shows a noticeable preliminary advantage of the DVRL method. Particularly, the payoff of DVRL is higher than the others until the 12th round, where it becomes proximately equal to DVG. This preliminary advantage can be attributed to the low extent of reinforcement in the RL's Q-values update procedure, compared to the extensive reinforcement conducted by Gittins' indices. For example, the Q-value of an offer which succeeded 2 out of 3 interactions, would be reinforced in the RL method by 66% of its reward, while using Gittins' method it would be reinforced by 91%. An offer which was successful only once out of 5 interactions, would be reinforced in the RL approach by 20% of its reward, whereas while using Gittins' method it would be reinforced by 47%. Thus, offers which are extremely far from the optimal will be filtered more quickly with the pedant RL methods.

The pedantry of RL is also the reason for the relative success of the SVRL in the first learning stages. However, in SVRL this pedantry becomes an obstacle in the progressive learning stages: Optimal segments, where the probability of success is usually not 100%, were sometimes rejected after a few interactions, and were hardly focused in later stages[9]. However, this problem in the progressive learning stages, was much less harmful in the DVRL method. The reason for this is that, in contrast to SVRL, (optimal) options with currently low Q-values will not necessarily be abandoned for long. In addition, according to the DVL update protocol, (optimal) options are also reinforced by the success of options which are less risky. Thus, the DVRL becomes less sensitive to over filtering of the pedant RL method.

This difference in sensitivity between the segment-based and the DVL approaches underlies the superiority of DVG over the VSBG algorithm, in the progressive learning stages, as noticeable on the right side of **Figure 1** (from the 16th round on). However, the difference is much less significant than the difference between the RL-based algorithms, since the Gittins approach is much less pedant. Therefore, the SVG is not critically harmed by the abandonment of low valued options. Moreover, the fast convergence to an optimal (large) segment which usually occurred in SVG, provided a superiority over the DVG algorithm, in the initial interactions.

Another finding in our experiment was the general improvement caused by virtual learning. This could be explained by the fast

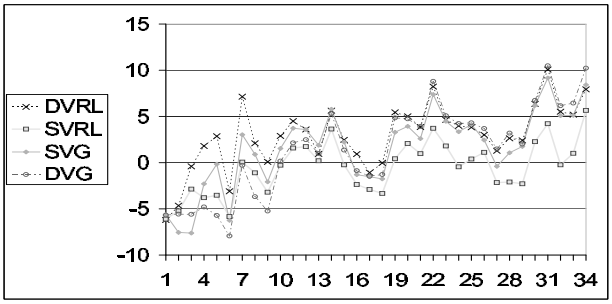


Figure 1: Average payoff during the all-pay auction

updating of a large set of offers after every interaction, with the VL method. Thus, offers which are significantly irrelevant are filtered within a few interactions, while it is easy to find a non virtual learner that proposes 90% in UG after the 20th interaction, for example. This significant advantage offsets the disadvantage in accuracy, caused by the information asymmetry in the virtual updating process mentioned above.

In order to measure the stability and self-consistence of each algorithm we compared the standard deviations of the payoff results of 50 runs with each permutation of the ten algorithms. Observing the standard deviations in **Table 1**, it can be concluded that the standard deviation of DVRL was always the least ($p < 0.001$), except for the UG, where both DVG and SVG were lower than DVRL. Since low values of standard deviation guarantee more certain and stable results, using the proposed DVRL algorithm is preferable from this aspect as well. This property may be attributed to the fact that the DVL meta-algorithm includes no random decision, except for the first interaction. In addition, the fast screening in the DVRL as well as the lower sensitivity to over-filtering of the optimal solution discussed earlier, provide a stable and consistent pattern of solution search. For similar reasons, the relative ranking of other algorithms in the deviations measures were compatible with the relative ranking in the payoffs measures.

4.3.3 Comparison with human performance

In this section we compare the performance of the DVRL algorithm with the performance of human competitors. For this purpose we asked our participants to compete iteratively against a series of other human opponents, exactly in the same manner the automated agents had been competing. In this manner we wanted to reveal whether the DVRL agent can outperform human natural intuitions and life experience in this context. Note that the automated agent was run with the same parameters for all the environments, and no domain specific knowledge had been applied to the agents.

We began our examination after collecting a series of acceptance thresholds in the ultimatum games from the first 21 participants, as

[9]Extending the duration of the first stages, where a few number of large segments are observed, would have reduced this problem. However, since we considered environments with a total number of 50 interactions at most, the average total payoff would not have risen by extending the duration of the first stages, since it would often harm the later interactions, where the fine tuning is managed. The final stage of fine tuning is critically important, especially in the context of CE environments.

Environment (# of human participants)	**Human**	**DVRL**
UG (13)	49.56 (3.41)	52.59
UG variant (13)	40.8 (7.45)	44.36
Auction (13)	15.74 (4.02)	19.01
All-pay Auction (13)	-0.18 (3.53)	1.4
Normal distribution auction (15)	11.79 (2.78)	14.35

Table 2: Average payoff (SD) of humans and of a DVRL agent competing with human opponents.

well as their biddings in the auctions (whereas in previous sections the series included all the 34 participants). In order to enlarge the size of the series we inserted each opponent twice, creating a 42-round auction, a 34-round UG and a 36-round UG variant (when using the original series without the enlargement, the results were not qualitatively different). Similarly, we constructed an artificial series of 40 biddings in an auction normally distributed, as before. The consequent participants were asked to interact with those series, functioning as proposers in the ultimatum games and as bidders in the auctions. After each decision, the participants were informed of the success of their offer by the "current" opponent. Actually, these proposers were expected to learn the acceptance thresholds distribution of the responders' population, exactly as the agents learn. All the learners, including the automated learner, competed with the same series of opponents.

The average payoffs of the human learners, as well as the payoffs of the DVRL agent, in the exact same scenarios, are presented in **Table 2**. The results show a clear advantage of the DVRL agent over the human learners in all the environments. Although this statement cannot be proved statistically, since we ran only one permutation from each environment, the results show that the DVRL average payoff was usually about one standard deviation above the human's average payoff.

4.3.4 A small set of decision options environments

In order to check the performance of our meta-algorithms in a small set of options, we performed the exact same simulations with merely 10 decision options. The agents could choose an integer from 0 to 10 in each interaction, and the original values of the opponents' vectors were divided by 10 and then rounded to the closest integer. The results, not presented here due to space restrictions, showed a significant advantage of the DVRL over the other algorithms, both in payoffs and in standard deviations, almost in all cases. Moreover, the internal hierarchy among the algorithms was similar to the 100-options version, and we found a general significant advantage of both meta-algorithms over the basic algorithms. Nonetheless, the gap between the performance of the basic algorithms and their extensions was smaller in the 10-options version.

5. CONCLUSION AND FUTURE WORK

We presented a new meta-algorithm, namely DVL, which extends existing methods for efficiently competing in multi-interaction one-shot CE environments with different human opponents. Our experimental findings show that the DVL extension of an RL basic algorithm performs significantly better than humans and other algorithms surveyed in this paper, especially in environments which contain a large set of optional decisions at each decision point. Moreover, our results show that both DVL and the segment-based meta algorithm, also presented here for the first time, almost always significantly improve the performance of the basic algorithms they extend.

In future work we intend to design agents which efficiently compete in CE environments for a high number of interactions. As mentioned above, in this study we were principally concerned with adaptable agents, which compete in no more than several dozens of repeated interactions. However, the DVRL algorithm does not necessarily grant efficiency for environments which last for hundreds of interactions. The reason is mainly due to the fact that the directed search for an optimal option, partially neglects the exploration of other options. Thus, we would like to explore the "damage" DVRL and other algorithms cause in the long run, and to suggest efficient approaches for interacting in CE environments for long durations.

In addition, we intend to extend the approaches discussed in this paper to more sophisticated classes of interactions. Particularly, we would like to examine the repeated version of CE interactions, in which several negotiation rounds can be conducted against each opponent. When competing repeatedly against the same opponent, a specific modeling of the current opponent must be done, in addition to the generic modeling of the population. Moreover, in contrast to the one-shot version, a current move may influence the future behavior of the opponent, a fact that must be taken into account. Therefore, we intend to design an agent that develops several optional models of typical opponents in the population, and matches the appropriate model to each opponent with which it interacts.

Acknowledgements: This work is supported in part by NSF IIS0222914. Kraus is affiliated with UMIACS. In memory of Matat Rosenfeld-Adler.

6. REFERENCES

[1] P. Bourgine and B. Leloup. May learning explain the ultimatum game paradox ? Technical Report GRID Working Paper No. 00-03, Ecole Polytechnique, 2000.

[2] T. Brenner and N. Vriend. On the behavior of proposers in ultimatum games. *J. Econ. Behav. Organ.* Forthcoming.

[3] A. Byde, C. Preist, and N. Jennings. Decision procedures for multiple auctions. In *AAMAS*, pages 613–620, 2002.

[4] J. DiMicco, A. Greenwald, and P. Maes. Dynamic pricing strategies under a finite time horizon. In *EC'01*, 2001.

[5] S. Fatima, M. Wooldridge, and N. Jennings. Sequential auctions for objects with common and private values. In *AAMAS*, pages 635–642, 2005.

[6] Y. Gal, A. Pfeffer, F. Marzo, and B. Grosz. Learning social preferences in games. In *AAAI*, pages 226–231, 2004.

[7] J. Gittins. *Multiarmed Bandits Allocation Indices*. Wiley, New York, 1989.

[8] B. Grosskopf. Reinforcement and directional learning in the ultimatum game with responder competition. *Experimental Economics*, 6(2):141–158, 2003.

[9] W. Guth and S. Huck. From ultimatum bargaining to dictatorship - an experimental study of four games varying in veto power. *Metroeconomica*, 48(3):262–279, 1997.

[10] V. Krishna and J. Morgan. An analysis of the war of attrition and the all-pay auction. *J. Econ. Theory*, 72:343–362, 1997.

[11] B. Leloup and L. Deveaux. Dynamic pricing on the internet: Theory and simulations. *JECR*, 1(3):265–276, 2001.

[12] L. Niklasson, H. Engstrom, and U. Johansson. An adaptive rock, scissors and paper player based on a tapped delay neural network. In *ADCOG21*, pages 130–136, 2001.

[13] A. Roth and I. Erev. Learning in extensive form games: Experimental data and simple dynamic models in the intermediate term. *Games Econ. Behav.*, 8:164–212, 1995.

[14] R. Sutton and A. Barto. *An Introduction to Reinforcement Learning*. MIT Press, 1998.

[15] P. Todd and B. Borges. Designing socially intelligent agents for the ultimatum game. In K. Dautenhahn, editor, *Socially intelligent agents-Papers from the 1997 Fall Symposium*, pages 134–136. AAAI Press, Menlo Park, CA, 1997.

[16] N. Vreind. Will reasoning improve learning? *Econ. Lett.*, 55(1):9–18, 1997.

[17] F. Zhong, D. Wu, and S. Kimbrough. Cooperative agent systems: Artificial agents play the ultimatum game. *Journal of Group Decision and Negotiation*, 11(6):433–447, 2002.

A Novel Method for Automatic Strategy Acquisition in N-player Non-zero-sum Games

S. Phelps
Dept. of Computer Science
University of Liverpool
Liverpool L69 3BX
United Kingdom
sphelps@csc.liv.ac.uk

M. Marcinkiewicz
Dept. of Computer Science
Graduate Center
City University of New York
365, 5th Avenue
New York, NY 10016
USA
marek@sci.brooklyn.cuny.edu

S. Parsons
Dept. of Computer Science
City University of New York
365, 5th Avenue
New York, NY 10016, USA
parsons@sci.brooklyn.cuny.edu

ABSTRACT

We present a novel method for automatically acquiring strategies for the double auction by combining evolutionary optimization together with a principled game-theoretic analysis. Previous studies in this domain have used standard co-evolutionary algorithms, often with the goal of searching for the "best" trading strategy. However, we argue that such algorithms are often ineffective for this type of game because they fail to embody an appropriate game-theoretic *solution-concept*, and it is unclear, what, if anything, they are optimizing. In this paper, we adopt a more appropriate criterion for success from evolutionary game-theory based on the likely adoption-rate of a given strategy in a large population of traders, and accordingly we are able to demonstrate that our evolved strategy performs well.

Keywords

auctions and electronic markets, multi-agent evolution, adaptation and learning, game theoretic foundations of agent systems

1. INTRODUCTION

The automatic discovery of game-playing strategies has long been considered a central problem in Artificial Intelligence. The standard technique in evolutionary computing for discovering new strategies is *co*-evolution, in which the fitness of each individual in an evolving population of strategies is assessed relative to other individuals in that population by computing the payoffs obtained when the selected individuals interact. Co-evolution can sometimes result in *arms-races*, in which the complexity and robustness of strategies in the population increases as they counter-adapt to adaptations in their opponents.

Often, however, co-evolutionary learning can fail to converge on robust strategies. In this paper we explore some of the limitations of current co-evolutionary algorithms, and we introduce a novel method for automated strategy-acquisition by combining a game-theoretic analysis together with an evolutionary search involving a single non-coevolving population. The novel aspect of our method is that rather than using payoffs to individual strategies as our fitness function, we instead use the size of their basin of attraction under an evolutionary game-theoretic analysis. We apply our method to the double auction market [5]: a domain which has long served as a benchmark problem in understanding strategic-interactions in multi-agent systems.

The outline of this paper is as follows. In Sections 2 and 3 we give a brief introduction to game theory and evolutionary game theory within the context of automated strategy-acquisition. In Section 4 we review some of the weaknesses of standard co-evolutionary algorithms from the perspective of game-theory. In Section 5 we give a brief overview of a technique called heuristic-strategy approximation, also known as empirical game theory, on which our method is based, and discuss how it is able to overcome some of the limitations of co-evolutionary algorithms when analysing strategic interactions amongst *existing* well-known strategies for a given game. In Section 6 we perform a heuristic-strategy analysis on a variant of the double auction market called a clearing-house. In Section 7 we describe our novel contribution to this domain; viz, how we are able to combine heuristic-strategy analysis and evolutionary search in order to automatically acquire *new* strategies for the double auction game, and finally in sections 8 and 9 we discuss our results.

2. NASH EQUILIBRIUM

The failure of certain types of co-evolutionary algorithms to converge on robust strategies in certain scenarios is well known, and has many possible causes; for example, the population may enter a limit cycle if strategies learnt in earlier generations are able to exploit current opponents and current opponents have "forgotten" how to beat the revived living fossil. Whilst many effective techniques have been developed to overcome these problems, there remains, however, a deeper problem which is only beginning to be addressed successfully. In some games, such as Chess, we can safely bet that if player A consistently beats player B, and player B consistently beats player C, then player A is likely to beat

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

player C. Since the dominance relationship is transitive, we can build meaningful *rating systems* for objectively ranking players in terms of ability, and the use of such ranking systems can be used to assess the "external" fitness of strategies evolved using a co-evolutionary process and ensure that the population is evolving toward better and better strategies. In many other games, however, the dominance graph is highly intransitive, making it impossible to rank strategies on a single scale. In such games, it makes little sense to talk about "best", or even "good", strategies since even if a given strategy beats a large number of opponent strategies there will always be many opponents that are able to beat it. The best strategy to play in such a game is always dependent on the strategies adopted by one's opponents.

Game theory provides us with a powerful concept for reasoning about the best strategy to adopt in such circumstances: the notion of a *Nash equilibrium*. A set of strategies for a given game is a Nash equilibrium if, and only if, no player can improve their payoff by unilaterally switching to an alternative strategy.

If there is no dominant strategy[1] for the game, then we should play the strategy that gives us the best payoff based on what we believe our opponents will play. If we assume our opponents are payoff maximisers, then we know that they will play a Nash strategy set by *reductio ad absurdum*; if they did not play Nash then by definition at least one of them could do better by changing their strategy, and hence they would not be maximising their payoff. This is very powerful concept, since although not every game has a dominant strategy, every finite game posseses at least one *equilibrium* solution. Additionally, if we know the entire set of strategies and payoffs, we can deterministically compute the Nash strategies. If only a single equilibrium exists for a given game, this means that, in theory at least, we can always compute the "appropriate" strategy for a given game.

Note, however, that the Nash strategy is not always the *best* strategy to play in all circumstances. For 2-player zero-sum games, one can show that the Nash strategy is not exploitable. However, if our opponents do not play their Nash strategy, then there may be other non-Nash strategies that are better at exploiting off-equilibrium players. Additionally, many equilibria may exist and in n-player non-constant-sum games it may be necessary for agents to *coordinate* on the same equilibrium if their strategy is to remain secure against exploitation; if we were to play a Nash strategy from one equilibrium whilst our opponents play a strategy from an alternative equilibrium we may well find that our payoff is significantly lower than if we had coordinated on the same equilibrium as our opponents.

3. BEYOND NASH EQUILIBRIUM

Standard game theory does not tell us which of the many possible Nash strategies our opponents are likely to play. *Evolutionary* game theory [10] and its variants attack this problem by positing that, rather than computing the Nash strategies for a game using brute-force and then selecting one of these to play, our opponents are more likely to gradually adjust their strategy over time in response to to repeated observations of their own and others' payoffs. One approach to evolutionary game-theory uses the *replicator dynamics* equation to specify the frequency with which different pure strategies should be played depending on our opponent's strategy:

$$\dot{m}_j = [u(e_j, \vec{m}) - u(\vec{m}, \vec{m})]\, m_j \tag{1}$$

where $\vec{m}$ is a mixed-strategy vector, $u(\vec{m}, \vec{m})$ is the mean payoff when all players play $\vec{m}$, and $u(e_j, \vec{m})$ is the average payoff to pure strategy j when all players play $\vec{m}$, and $\dot{m}_j$ is the first derivative of m_j with respect to time. Strategies that gain above-average payoff become more likely to be played, and this equation models a simple *co-evolutionary* process of mimicry learning, in which agents switch to strategies that appear to be more successful.

For any initial mixed-strategy we can find the eventual outcome from this co-evolutionary process by solving $\dot{m}_j = 0$ for all j to find the final mixed-strategy of the converged population. This model has the attractive properties that: (i) all Nash equilibria of the game are stationary points under the replicator dynamics; and (ii) all focal points of the replicator dynamics are Nash equilibria of the evolutionary game.

Thus the Nash equilibrium solutions are embedded in the stationary points of the direction field of the dynamics specified by equation 1. Although not all stationary points are Nash equilibria, by overlaying a dynamic model of learning on the equilibria we can see which solutions are more likely to be discovered by *boundedly-rational* agents. Those Nash equilibria that are stationary points at which a larger range of initial states will end up, are equilibria that are more likely to be reached (assuming an initial distribution that is uniform).

This is all well and good in theory, but the model is of limited practical use since many interesting real-world games are *multi-state*[2]. Such games can be transformed into normal-form games, but only by introducing an intractably large number of pure strategies, making the payoff matrix impossible to compute.

4. CO-EVOLUTION

But what if we were to approximate the replicator dynamics by using an evolutionary search over the strategy space? Rather than considering an infinite population consisting of a mixture of all possible pure strategies, we use a small finite population of randomly sampled strategies to approximate the game. By introducing mutation and cross-over, we can search hitherto unexplored regions of the strategy space. Might such a process converge to some kind of approximation of a true Nash equilibrium? Indeed, this is one way of interpreting existing co-evolutionary algorithms; fitness-proportionate selection plays a similar role to the replicator dynamics equation in ensuring that successful strategies propagate, and genetic operators allow them to search over novel sets of strategies. There are a number of problems with such approaches from a game-theoretic perspective, however, which we shall discuss in turn.

Firstly, the proportion of the population playing different strategies serves a dual role in a co-evolutionary algorithm [3]. On the one hand, the proportion of the population playing a given strategy represents the probability of playing

[1]A strategy which is always the best one to adopt no matter what any opponent does.

[2]The payoff for a given move at any stage of the game depends on the history of play.

that pure strategy in a mixed-strategy Nash equilibrium. On the other hand, evolutionary search requires diversity in the population in order to be effective. This suggests that if we are searching for Nash equilibria involving mixed-strategies where one of the pure strategy components has a high frequency, corresponding to a co-evolutionary search where a high percentage of the population is adopting the same strategy, then we may be in danger of over-constraining our search as we approach a solution.

Secondly and relatedly, although the final set of strategies in the converged population may be best responses to each other, there is no guarantee that the final mix of strategies cannot be invaded by other yet-to-be-countered strategies in the search space, or strategies that became extinct in earlier generations because they performed poorly against an earlier strategy mix that differed from the final converged strategy mix. Genetic operators such as mutation or crossover will be poor at searching for novel strategies that could potentially invade the newly established equilibrium because of the above problem. If these conditions hold, then the final mix of strategies is implausible as a true Nash equilibrium or ESS, since there will be unsearched strategies that could potentially break the equilibrium by obtaining better payoffs for certain players. We might, nevertheless, be satisfied with the final mix of strategies as an approximation to a true Nash equilibrium on the grounds that if our co-evolutionary algorithm is unable to find equilibrium-breaking strategies, then no other algorithm will be able to do so. However, as discussed above, we expect *a priori* that co-evolutionary algorithms will be particularly *poor* at searching for novel strategies once they have discovered a (partial) equilibrium.

Finally, co-evolutionary algorithms employ a number of different selection methods, not all of which yield population dynamics that converge on game-theoretic equilibria [4].

These problems have led researchers in co-evolutionary computing to design new algorithms employing game theoretic solution concepts [2]. In particular, [3] describe a game-theoretic search technique for acquiring approximations of Nash strategies in large symmetric 2-player constant-sum games with type-independent payoffs. In this paper, we address n-player non-constant-sum multi-state games with type-dependent payoffs. In such games, playing our Nash strategy (or an approximation thereof) does not guarantee us security against exploitation, thus if there are multiple equilibria, it may be more appropriate to play a *best-response* to the strategies that we infer are in play.

5. EMPIRICAL GAME-THEORY

Reeves et al. [9] and Walsh et al. [11] obviate many of the problems of standard co-evolutionary algorithms by restricting attention to small representative sample of "heuristic" strategies that are known to be commonly played in a given multi-state game. For many games, unsurprisingly none of the strategies commonly in use is dominant over the others. Given the lack of a dominant strategy, it is then natural to ask if there are mixtures of these "pure" strategies that constitute game-theoretic equilibria.

For small numbers of players and heuristic strategies, we can construct a relatively small normal-form payoff matrix which is amenable to game-theoretic analysis. This *heuristic* payoff matrix is calibrated by running many iterations of the game; variations in payoffs due to different player-types (eg black or white, buyer or seller) or stochastic environmental factors (eg PRNG seed) are averaged over many samples of type information resulting in a single mean payoff to each player for each entry in the payoff matrix. Players' types are assumed to be drawn independently from the same distribution, and an agent's choice of strategy is assumed to be independent of its type, which allows the payoff matrix to be further compressed, since we simply need to specify the number of agents playing each strategy to determine the expected payoff to each agent. Thus for a game with k strategies, we represent entries in the heuristic payoff matrix as vectors of the form

$$\vec{p} = (p_1, \ldots p_k)$$

where p_i specifies the number of agents who are playing the ith strategy. Each entry $p \in P$ is mapped onto an outcome vector $q \in Q$ of the form

$$\vec{q} = (q_1, \ldots q_k)$$

where q_i specifies the expected payoff to the ith strategy. For a game with n agents, the number of entries in the payoff matrix is given by

$$s = \frac{(n+k-1)!}{n!(k-1)!} \tag{2}$$

For small n and small k this results in payoff matrices of manageable size; for $k = 3$ and $n = 6$, 8, and 10 we have $s = 28$, 45, and 66 respectively.

Once the payoff matrix has been computed we can subject it to a rigorous game-theoretic analysis, search for Nash equilibria solutions and apply different models of learning and evolution, such as the replicator dynamics model, in order to analyse the dynamics of adjustment to equilibrium.

The equilibria solutions that are thus obtained are not rigorous Nash equilibria for the full multi-state game; there is always the possibility that an unconsidered strategy could invade the equilibrium. Nevertheless, heuristic strategy equilibria are more plausible as models of real world game playing than those obtained using a standard co-evolutionary search precisely because they *restrict* attention to strategies that are commonly known and are in common use. We can therefore be confident that no commonly known strategy for the game at hand will break our equilibrium, and thus the equilibrium stands at least some chance of persisting in the short term future.

Of course, once an equilibrium is established, the designers of a particular strategy may not be satisfied with their strategy's adoption-rate in the game-playing population at large. As [11] suggest, the designers of, for example, a particular trading strategy in a market game may have financial incentives such as patent rights to increase their "market-share" – that is, the proportion of players using their strategy, or, in game-theoretic terms, the probability of their pure strategy being played in a mixed-strategy equilibrium with a large basin of attraction. They go on to propose a simple methodology for performing such optimization using manual design methods. A promising-looking candidate strategy is chosen for perturbation analysis; a new, perturbed, version of the original heuristic payoff matrix is computed in which the payoffs of the candidate strategy are increased by a small fixed percentage, thus modelling a hypothetical tweak to the strategy that yields in a small increase in payoffs. The replicator-dynamics direction field is then replotted to establish whether the hypothetically-optimized strategy is able to

to achieve a high adoption rate in the population. Strategy designers can then concentrate their efforts on improving those strategies that become strong attractors with a small increase in payoffs.

In this paper, we extend this technique by using a genetic-algorithm (GA) to *automatically* optimize candidate strategies by searching for a hitherto-unknown best-response — or, to use more appropriate nomenclature, a *better-response* — to an existing mix of heuristic strategies. Rather than using a standard co-evolutionary algorithm to perform the optimization, we use a single-population GA where the fitness of an individual strategy is computed from the heuristic-strategy payoff matrix according to the basin size it yields under the replicator dynamics.

6. THE DOUBLE-AUCTION

We apply our method to the acquisition of strategies for the *double auction* [5]. The double auction is a generalisation of more commonly-known single-sided auctions, such as the English ascending auction, which involve a single seller trading with multiple buyers. In a double auction, we allow multiple traders on both sides of the market; as well as soliciting offers to buy a good from buyers, that is bids, we also solicit offers to sell a good from sellers, so called *asks*. Variants of the double auction are commonly used in many real-world market places such as stock exchanges in scenarios where supply and demand are highly dynamic. Whilst single-sided auctions are well-understood from a game theoretic perspective, double-sided auctions remain intractable to a full game-theoretic analysis especially when there are relatively few traders on each side of the market. Thus much analysis of this game has focused on using heuristic methods to explore viable bidding strategies.

In previous work [8], we used a heuristic-strategy analysis to analyse two variants of the double auction market mechanism populated with a mix of heuristic strategies, and were able to find approximate game-theoretic equilibrium solutions. In this paper, we use the same basic framework, but we focus on the *clearing-house* double auction (CH) [5] with uniform pricing, in which all agents are polled for their offers before transactions take place, and all transactions are then executed at the same market-clearing price. In this paper, we consider only the following three heuristic-strategies. Future work will use a more representative (and larger) set of heuristic-strategies to optimize against.

- The truth-telling strategy (TT), whereby agents submit offers equal to their valuation for the resource being traded (in a strategy-proof market, TT will be a dominant strategy);
- The Roth-Erev strategy (RE) – a strategy based on reinforcement learning, described in [1] and calibrated with the parameters specified in [7]; and
- The Gjerstad-Dickhaut strategy (GD) [6], whereby agents estimate the probability of any bid being accepted based on historical market data and then bid to maximize expected profit.

Since all mixed-strategy vectors lie in the unit-simplex, for $k = 3$ strategies we can project the unit-simplex onto a two dimensional space and then plot the switching between strategies that occurs under the dynamics of equation 2. Figure 1 shows the direction-field of the replicator-dynamics equation for these three heuristic strategies, showing that we have two equilibrium solutions. Firstly, we see that GD is a best-response to itself, and hence is a pure-strategy equilibrium. We also see it has a very large *basin of attraction*; for any randomly-sampled initial configuration of the population most of the flows end up in the bottom-right-hand-corner. Additionally, there is a second mixed-strategy equilibria at the coordinates (0.88, 0.12, 0) in the simplex corresponding to an 88% mix of TT and a 12% mix of RE, however the attractor for this equilibrium is much smaller than the pure-strategy GD equilibrium; only 6% of random starts terminate here vs 94% for pure GD. Hence, according to this analysis, we expect most of the population of traders to adopt the GD strategy.

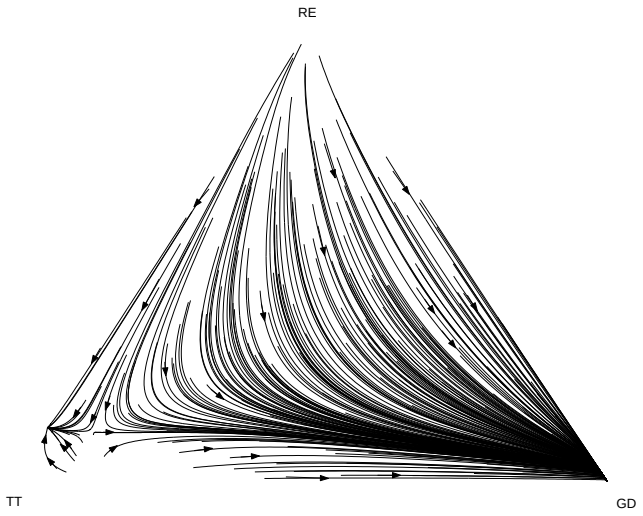


Figure 1: The original replicator dynamics direction field for a 12-agent clearing-house auction with the original unoptimized Roth-Erev strategy (labelled RE).

How much confidence can we give to this analysis given that the payoffs used to construct the direction-field plot were estimated based on only 2×10^3 samples of each game? One approach to answering this question is to conduct a sensitivity analysis; we perturb the mean payoffs for each strategy in the matrix by a small percentage to see if our equilibria analysis is robust to errors in the payoff estimates. Figure 2 shows the direction-field plot after we perform a perturbation where we remove 2.5% of the payoffs from the TT and GD strategies and assign +5% payoffs to the RE strategy. This results in a qualitatively different set of equilibria; the RE strategy becomes a best-response to itself with a large basin of attraction (61%), and thus we conclude that our equilibrium analysis is sensitive to small errors in payoff estimates, and that our original prediction of widespread adoption of GD may not occur if we have underestimated the payoffs to RE.

If we observe a mixture of all three strategies in actual play, however, the perturbation analysis also suggests that we could bring about widespread defection to RE if were able to tweak the strategy by improving its payoff slightly; *the perturbation analysis points to RE as a candidate for potential optimization.*

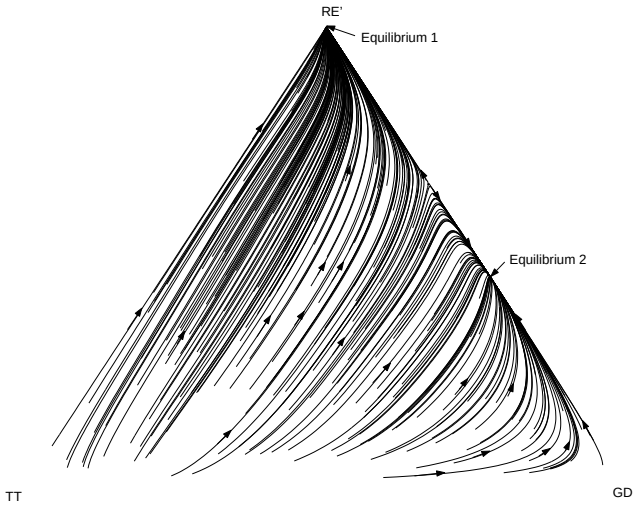


Figure 2: Replicator dynamics direction field for a 12-agent clearing-house auction perturbed with +5% payoffs to the Roth-Erev strategy (labeled RE')

7. STRATEGY ACQUISITION

In the previous section we described how we used heuristic-strategy approximation to identify a candidate strategy for optimization. We also introduced an intriguing metric for ranking strategies on a single fully-ordered scale: viz, the size of the strategy's basin of attraction under the replicator dynamics. In this section we use this metric to perform a heuristic search of a space of strategies closely related to the RE strategy. In the following we define the space of strategies that we search, and the details of the search algorithm.

The RE strategy uses re-inforcement learning (RL) to choose from n possible markups over the agent's limit price based on a reward signal computed as a function of profits earned in the previous round of bidding. Agents bid or ask at price p

$$p = l \pm mo \tag{3}$$

where l is the agent's limit price, o is the output from the learning algorithm and m is a scaling parameter. Additionally, the Roth-Erev learning algorithm itself has several free parameters: the recency parameter r, the experimentation parameter x, and an initialisation parameter $s1$. In addition to the original Roth-Erev (RE) algorithm, there are several other learning-algorithms that that have successfully been used for RL strategies in ACE. We search over three additional possibilities: stateless Q-learning (SQ), modifications to RE used by [7] (NPT) and a control algorithm which selects a uniformly random action regardless of reward signal (DR). SQ has free parameters: the discount-rate g, experimentation ϵ, and a learning-rate p.

Individuals in this search space were represented as a 50-bit string, where:

- bits 1-8 coded for parameter m in the range $(1, 10)$;
- bits 9-16 coded for the parameters ϵ or x in the range $(0, 1)$;
- bits 17-24 coded for parameter n in the range $(2, 258)$;
- bits 25-32 coded for parameters g or r in the range $(0, 1)$;
- bits 33-40 coded for parameter $s1$ in the range $(1, 1.5 \times 10^4)$;
- bits 41-42 coded for the choice of learning algorithm amongst RE, NPT, SQ or DR; and
- Bits 43-50 coded for parameter p in the range $(0, 1)$.

We used a genetic-algorithm (GA) to search this space of strategies, where the fitness of each individual strategy in the search space was computed by estimating its basin size under the replicator dynamics under interaction with our existing three strategies: GD, TT and RE. Basin size was estimated using the same brute-force methods described in Section 5, but since we recompute all entries in the heuristic-payoff matrix in support of each candidate strategy, we used lower sample sizes in order to facilitate evaluation of many strategies; the sample size for the number of games played for each entry in the heuristic payoff matrix was increased as a function of the generation number: $10 + int(100 \ln(g + 1))$ allowing the search-algorithm to quickly find high-fitness regions of the search-space in earlier generations and reducing noise and allowing more refinement of solutions in later generations. We used a constant number of replicator-dynamics trajectories — 50 — in order to estimate the basin size from the payoff matrix once it had been recomputed for our candidate strategy.

We chose a GA as our search algorithm principally because of its ability to cope with the additional noise that the lower sample size introduced into the objective function Our GA was configured with a population size of 100, with single-point cross-over, a cross-over rate of 1, a mutation-rate of 10^{-4} and fitness-proportionate selection. The GA was run for 32 generations, which took approximately 1800 CPU hours.

8. RESULTS

Figure 5 shows the mean fitness of the GA population for each generation. As can be seen, there is still a large amount of variance in fitness values in later generations. However, inspection of a random sample of strategies from each generation revealed a partial convergence of phenotype, but with significant fluctuations in fitness values due to small sample sizes (see above). Most notably, the fittest individual at generation 32 had also appeared intermittently as the fittest individual five times in the previous 10 generations, and thus we took this as the output from the search.

Our optimised strategy uses the stateless Q-learning algorithm with the following parameters:

$$\begin{aligned} m &= 1.210937 \\ n &= 6 \\ \epsilon &= 0.18359375 \\ g &= 0.4140625 \\ p &= 0.1875 \end{aligned}$$

The notable feature of this strategy is the small number of possible markups (n), and the narrow range of the markups

$[1, nm]$ as compared with the distribution of valuation distribution widths. This feature was shared by all of the top five strategies in the last ten generations, and is another factor that indicated convergence of the search.

We proceeded to analyze our specimen strategy under a full heuristic-strategy analysis using 10^4 samples of the game for each of the 455 entries in the payoff matrix. With the current version of our simulator[3], we are able to complete this analysis in less than twenty four hours using a dual-processor 3.6Ghz Xeon workstation.

Figure 6 shows twenty trajectories of the replicator dynamics plotted as a time series for each strategy, and shows the interaction between the new, optimised strategy, OS, together with the existing strategies: GD, TT and RE.

Taking 10^3 randomly sampled initial mixed-strategies, we calculate that there are two attractors:

$$\begin{aligned} A &= (0.0, 0.0, 1.0, 0.0) \\ B &= (0.67, 0.32, 0.0, 0.0) \end{aligned}$$

over (OS, TT, GD, RE). Attractor A captures only 3% of trajectories, whereas attractor B captures virtually the entire four-dimensional simplex (97%). Although this basin is very large, our optimized strategy shares this equilibrium with the truth-telling strategy (TT), giving us a final total market share to $0.67 \times 0.97 = 65\%$. This compares favourably with a market-share of 32% for truth-telling and 3% for GD. The original RE strategy is dominated by our optimised strategy. Figures 3 and 4 show the direction field for two of the 3-strategy combinations involving our optimised strategy: (OS, TT, GD) and (OS, GD, RE) respectively.

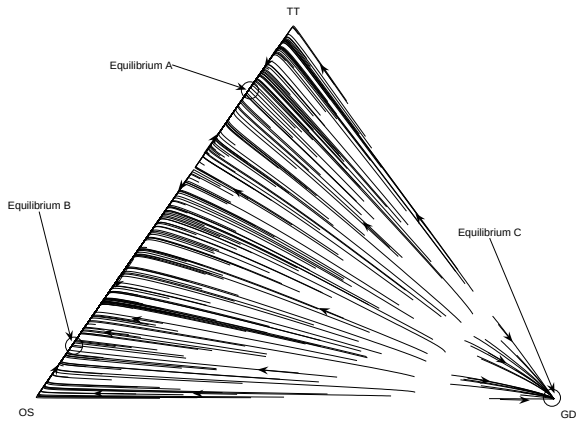


Figure 3: Replicator dynamics direction field for a 12-agent clearing-house auction showing interaction between optimised strategy (OS) verses TT and GD

9. DISCUSSION

It is somewhat remarkable that our somewhat simplistic optimised strategy is able to gain defectors from a highly sophisticated strategy like GD, whilst at the same time truth-telling is able to retain a share of followers in a population predominated by *OS*ers (TT appears to be *parasitic* on OS). What accounts for the ability of small OS mixes to invade high-probability mixes of a sophisticated adaptive strategy (GD), whilst remaining vulnerable to invasion by a low-probability mix of a non-adaptive strategy (TT)?

Our current hypothesis is that OS is able to exploit a flaw in the way that we construct valuation distributions. As discussed earlier, we use the same method of assigning valuations as in [11]; that is, for each run of the game, the lower-bound, b, of the valuation distribution is selected uniformly at random from the range $[61, 160]$ and the upper-bound b' is similarly drawn from $[b + 60, b + 209]$. For that run of the game, each agent's valuation is then drawn uniformly from $[b, b']$. However, we hypothesise that this results in a statistical correlation between the meta-bounds and the average slope of truthful supply and demand schedules— that is, given these distribution parameters there is insufficient variance in the difference between valuations of traders who are neighbours on the supply or demand curve. Since we are using a uniform-price k=0.5 clearing rule, the mechanism is vulnerable to price-manipulation from the least efficient trades; the buyer with the lowest matched bid, and the seller with the highest matched ask can potential manipulate the final clearing price - *provided that they do not overstate their value claim to the extent that it impinges on the 2nd-lowest matched bid, or the 2nd-highest matched ask.* Given sufficient variance in valuations between the lowest and the 2nd-lowest matched offers (call this random variable the "match delta"), this vulnerability is not easily exploited. However, in a market with a small number of traders and a narrow distribution for the match delta there is an opportunity to trade at small margin above truth if you find yourself with a valuation close to the equilibrium price. This is precisely the behaviour of the strategies that we observe to be predominant in the later generations of our GA- they all use a small number of possible markups, each of them small in comparison to the possible valuation bounds. The reinforcement-learning component of the strategy is then able to fine-tune the markup depending on where the trader finds themselves on the supply or demand curve after valuations are drawn. If it is far away from the equilibrium-price it can adjust its margin close to zero, whilst if it is near the equilibrium-price it can find a small margin that does not impinge on its nearest-neighbour. This hypothesis is also consistent with parasitic truth-telling; it is easy to see that truth-telling is a best-response for a 2nd-lowest matched bidder to a lowest matched bidder playing OS. Further work needs to be carried out to verify this hypothesis, but early indications are that the efficacy of our optimised strategy is highly sensitive to changes in the valuation meta-bounds.

Thus it appears that our evolved strategy over-fits to what is potentially an artifact. Does this invalidate our approach? On the contrary, we believe that it demonstrates that highly sophisticated strategies can possess an Achilles heel in certain situations, and that our method for strategy acquisition can find and exploit these vulnerabilities. GD performs robustly given a large variance in match delta, but given some partial prior knowledge about valuation distributions in a small market, then there may exist a more effective, and simpler strategy. In the particular case that valuation distributions are constructed as above, then the strategy OS will perform remarkably well.

10. SUMMARY

[3] `http://freshmeat.net/projects/jasa`

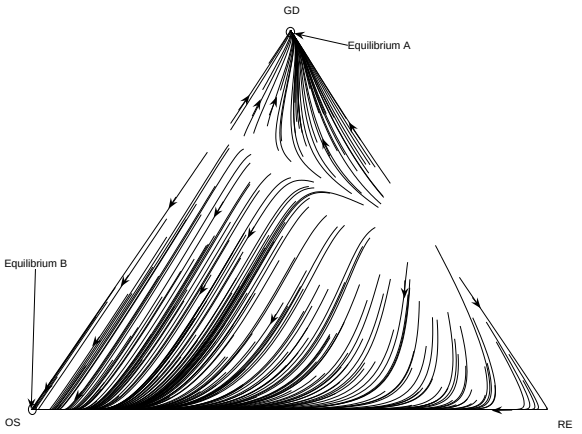


Figure 4: Replicator dynamics direction field for a 12-agent clearing-house auction showing interaction between optimised strategy (OS) verses GD and the original Roth-Erev strategy (RE)

In this paper we have described a novel method for strategy acquisition for n-player non-zero-sum games, and we have successfully applied our method to the acquisition of a trading strategy for the double-auction market; a recognised benchmark problem for this domain. Our method combines two existing methodologies in a very simple way: empirical game-theory and heuristic search (in this case, using a genetic-algorithm). Our key contribution to this domain is the use of the basin-size metric to rank strategies. Although we have not provided a thorough analytical verification of our method, we believe that at this stage its novel features over and above those of empirical game-theory are simple enough that a successful application of our method to well-known benchmark problem suffices to demonstrate the potential for future research.

11. FURTHER WORK

The main weaknesses of our current approach stem from: a) potential weaknesses in the underlying empirical game-theory methodology, and b) the lack of a customised search algorithm tailored specifically for this domain.

As regards a), we recognise that empirical game-theory is an emerging field. As such, advances and refinements are continually being made to address some of its earlier weaknesses. For example, Walsh et al. [12] describe a refinement to the method which allows for a more principled approach for determining optimal sample size, and [13] describe a further approximation technique that allows for a greatly reduced number of strategy profiles to be considered. These and other refinements will be incorporated into our approach in future work. One of the key acknowledged weaknesses of empirical game-theory is the somewhat arbitrary selection of initial heuristic-strategies. However, it is our hope that the search method outlined in this paper will become the basis of an iterative approach to extending an initial set of manually chosen heuristic-strategies by populating the mix with additional heuristic-strategies that are discovered through heuristic-search.

As regards b), designing such a search algorithm is future work. But note that *despite* the lack of a custom-designed search algorithm, we are still able to use a general-purpose heuristic-search method to successfully acquire an interesting novel strategy for our benchmark problem.

Acknowledgements

This work is supported by NSF grant IIS-0329037 — "Tools and Techniques for Automated Mechanism Design" and EPSRC grant GR/T10671/01 — "Market based Control of Complex Computational Systems".

12. ADDITIONAL AUTHORS

Additional author: P. McBurney (Dept. of Comp. Sci., Univ. of Liverpool, email: `peter@csc.liv.ac.uk`).

13. REFERENCES

[1] I. Erev and A. E. Roth. Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. *American Economic Review*, 88(4):848–881, 1998.

[2] S. Ficici. *Solution Concepts in Coevolutionary Algorithms*. PhD thesis, Brandeis University, May 2004.

[3] S. Ficici and J. Pollack. A game-theoretic memory mechanism for coevolution. In *LNCS 2723*, pages 286–297. Springer-Verlag, 2003.

[4] S. G. Ficici and J. B. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In *PPSN VI*. Springer Verlag, 16-20 2000.

[5] D. Friedman and J. Rust. *The Double Auction Market*. Westview, 1991.

[6] S. Gjerstad and J. Dickhaut. Price formation in double auctions. *Games and Economic Behaviour*, 22:1–29, 1998.

[7] J. Nicolaisen, V. Petrov, and L. Tesfatsion. Market power and efficiency in a computational electricity market with discriminatory double-auction pricing. *IEEE Trans. on Evol. Computation*, 5(5):504–523, October 2001.

[8] S. Phelps, S. Parsons, and P. McBurney. An evolutionary game-theoretic comparision of two double auction market designs. In *Agent-Mediated Electronic Commerce VI*. Springer-Verlag, 2004.

[9] D. Reeves, J. MacKie-Mason, M. Wellman, and A. Osepayshvili. Exploring bidding strategies for market-based shceduling. *Decision Support Systems*, 2005.

[10] J. Smith. *Evolution and the theory of games*. Cambridge University Press, 1982.

[11] W. Walsh, R. Das, G. Tesauro, and J. Kephart. Analyzing complex strategic interactions in multi-agent games. In *AAAI-02 Workshop on Game Theoretic and Decision Theoretic Agents*, 2002.

[12] W. Walsh, D. Parkes, and R. Das. Choosing samples to compute heuristic-strategy nash equilibrium. In *Agent-Mediated Electronic Commerce V*, 2003.

[13] M. P. Wellman, D. M. Reeves, K. M. Lockner, and R. Suri. Searching for walverine. In *Proceedings of the IJCAI-05 Workshop on Trading Agent Design and Analysis*, 2005.

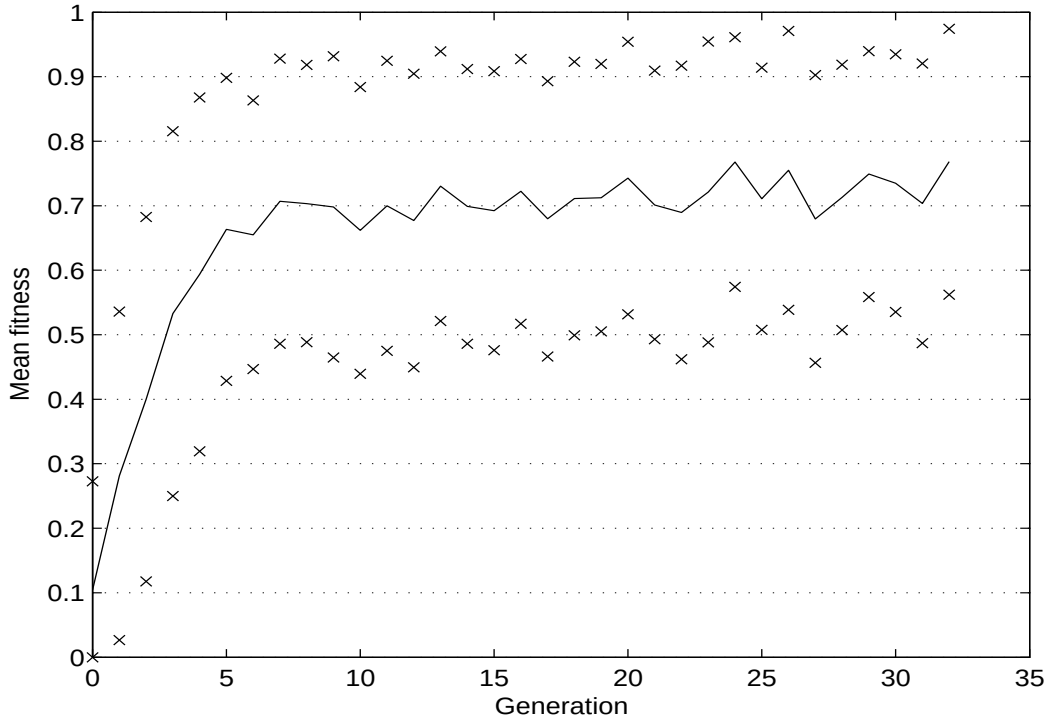


Figure 5: Mean fitness of the GA population with one stdev

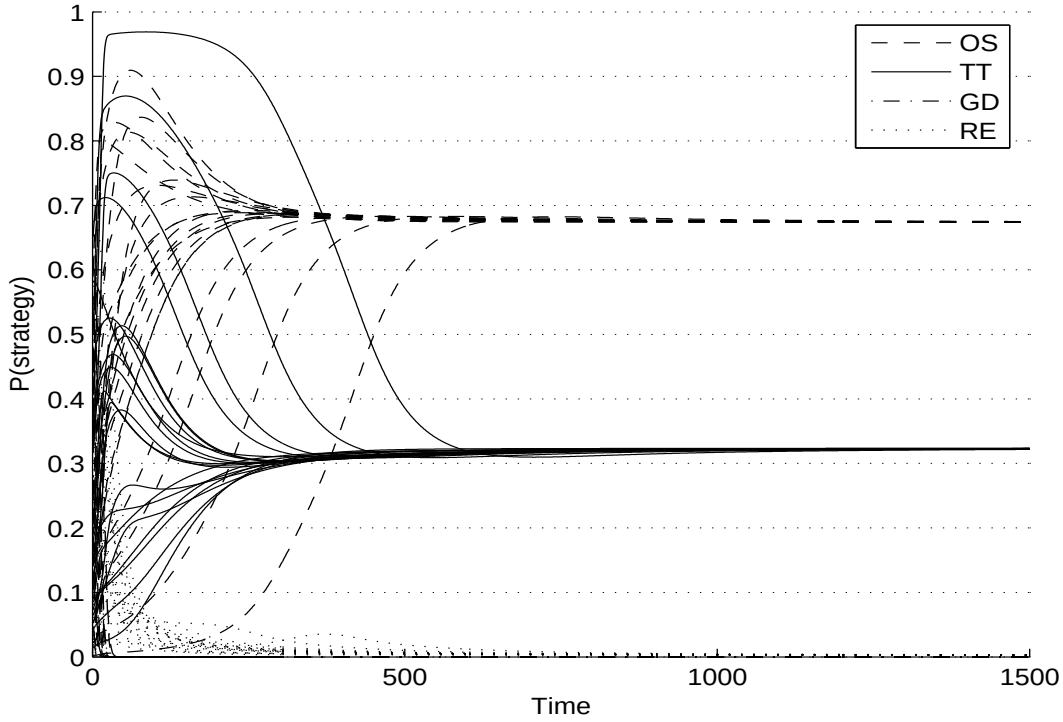


Figure 6: Replicator dynamics time series plot for a 12-agent clearing-house auction showing interaction between optimised strategy (OS) verses GD, TT and the original Roth-Erev strategy (RE)

Robust Game Play Against Unknown Opponents

Nathan Sturtevant
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
nathanst@cs.ualberta.ca

Michael Bowling
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
bowling@cs.ualberta.ca

ABSTRACT

A standard assumption of search in two-player games is that the opponent has the same evaluation function or utility for possible game outcomes. While some work has been done to try to better exploit weak opponents, it has only been a minor component of high-performance game playing programs such as Chinook or Deep Blue. However, we demonstrate that in games with more than two players, opponent modeling is a necessary component for ensuring high-quality play against unknown opponents. Thus, we propose a new algorithm, soft-max^n, which can help accommodate differences in opponent styles. Finally, we show an inference mechanism that can be used with soft-max^n to infer the playing style of our opponents.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*Games*

General Terms

Algorithms

Keywords

multi-player games, search, opponent modeling

1. INTRODUCTION AND BACKGROUND

The minimax decision rule is well-known for its performance guarantees in perfect-information game trees. If we have a perfect evaluation function, minimax gives a lower-bound guarantee on our score, regardless of our opponent. This is different than being maximally exploitive, however, because minimax will not necessarily take maximal advantage of a weak opponent.

In two-player zero-sum games, different algorithms have been suggested with the intent of using opponent modeling to take better advantage of opponent weaknesses [2, 3, 7]. While these algorithms work in theory, they require an *a priori* model of the opponent, and the improvement does not offset the loss of deeper search afforded by alpha-beta pruning in two-player games.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

In games with more than two players or teams of players we face a different situation. Game theory does not provide tight lower-bounds on performance in multi-player[1] games as in two-player zero-sum games, and existing algorithms for play demonstrate great weaknesses when playing new or unmodeled opponents. And, while better pruning techniques have recently been developed [14, 15], they are still not as effective as alpha-beta in two-player games.

This paper examines the role of opponent models for search in large multi-player perfect-information game trees. We make two novel contributions: a more robust algorithm for playing n-player games and a representation and method of inferring opponent preferences in these games.

This paper proceeds as follows: we will first discuss previous algorithms for opponent modeling in more detail, as well as the max^n algorithm [10] for playing multi-player games. We will then introduce a sample domain, the card game Spades, and demonstrate that an incorrect opponent model, such as assuming the opponent is using the same evaluation function, can be quite harmful. We introduce a new algorithm, soft-max^n, which can better accommodate differences in opponent styles. Finally, given this algorithm, we show how we can infer models of our opponents through observations of their play.

1.1 Related Work

Opponent modeling has been studied in the context of two-player, zero-sum, deterministic, perfect-information games for many years. Some of the earliest work on opponent modeling [8, 6] began with the recursive problem of opponent modeling. If I do not assume that my opponent is identical to me, I must have a model of my opponent. Similarly, if my opponent is also trying to model me, I will also need a model of my opponent's model of me, and so on. At first glance this seems to be a recursive nightmare. But, algorithms like M* [2] are capable of handling a number of opponent models and using them recursively. PrOM [3] expanded on these ideas, but still did not find measurably large success, even given a perfect opponent model to begin with.

In some sense, because most of this work is in the context of two-player, zero-sum games, it is not surprising that it has met with limited success in practice. This is the domain with the strongest theoretical algorithm, minimax, and thus has the least need for opponent modeling. We demonstrate here that in multi-player games there is a need for good opponent modeling. This same observation has been made in the context of poker [1, 13], although the approach in that game has been to model the opponent's strategy

[1]We use the phrase *multi-player* to specifically refer to situations with three or more players, as distinguished from the phrase *two-player*, although most of our statements with regard to multi-player games also apply to two-player non-zero-sum games.

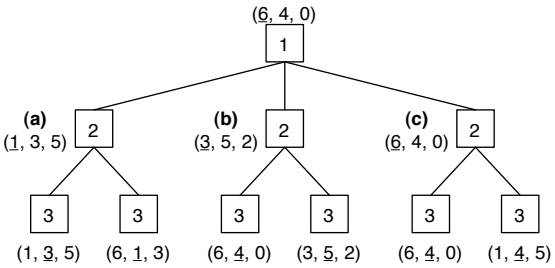


Figure 1: Example max^n tree.

rather than their evaluation function. In many cases the number of outcomes to be ordered is small compared to the domain's state space, making modeling of preferences more tractable.

1.2 Max^n

The max^n algorithm is used for calculating plays in perfect information game trees. In a max^n tree with n players, the leaves of the tree are n-tuples, where the ith element in the tuple is the ith player's score or utility for that position. At the interior nodes in the game tree, the max^n value of a node where player i is to move is the max^n value of the child of that node for which the ith component is maximum. Ties are traditionally broken toward the left, to maximize pruning. At the leaves of a game tree an exact or heuristic evaluation function can be applied to calculate the n-tuples that are backed up in the game tree.

We demonstrate this in Figure 1. In this tree there are three players. The player to move is labeled inside each node. At node (a), Player 2 is to move. Player 2 can get a score of 3 by moving to the left, and a score of 1 by moving to the right. So, Player 2 will choose the left branch, and the max^n value of node (a) is (1, 3, 5). Player 2 acts similarly at node (b) selecting the right branch, and at node (c) breaks the tie to the left, selecting the left branch. At node the root of the tree, Player 1 chooses the move at node (c), because 6 is greater than the 1 or 3 available at nodes (a) and (b).

1.3 Max^n Theoretical Properties

In order to better motivate the theoretical reasoning for our work, we briefly touch on one aspect of the game theory behind the minimax and max^n algorithms. The minimax algorithm works by computing the best response to any strategy a perfect opponent could play. The value of this strategy is called the minimax value or an equilibrium value. While there may be many strategies that allow a player to achieve this value in a two-player zero-sum game, each of these strategies has the same value.

Max^n calculates equilibrium values and strategies similar to minimax. It is simple to see that max^n returns a line of play which is an equilibrium, because at each point in the game tree players are always taking the move with the maximum score possible. Thus, no player can get a better result by unilaterally changing their strategy. See [10] for a full proof.

In a multi-player game there can be many different equilibrium strategies, but unlike in two-player zero-sum games, they are not guaranteed to all have the same value. Even more importantly, if your opponent is playing a different equilibrium strategy than you are, there is no guarantee that you will be in equilibrium, or that you will even receive the minimum equilibrium value that you expected to receive.[2]

[2]We could get a guaranteed lower bound on our score by reducing the n-player game to a two-player game, assuming all of our opponents are seeking to minimize our score. This approach is overly pessimistic, assuming that not only will opponents arbitrarily sacrifice their own score to reduce ours, but are also entirely coordinated in doing so.

Even in two-player games there are cases where there is a need to distinguish between different strategies with the same equilibrium value. Perhaps the biggest example is from the game of Checkers. In one situation the program CHINOOK was able to prove that a line of play was a draw, when humans mistakenly thought it was a win for the computer. Thus, it appeared to be a bug when CHINOOK made a move that led to a clear draw. In practice CHINOOK just lacked the ability to select between different equilibrium strategies to choose the one that was most likely to lead to an opponent mistake [12].

In the case of Checkers, the result of playing the minimax strategy without attention to an opponent model simply means that the program isn't maximally exploiting a weak opponent. Against a perfect opponent such differences don't matter. However, the concept of a perfect opponent in a multi-player game is ill-defined. In the next section we demonstrate the effects of playing a multi-player game when we have incorrect assumptions or models about our opponents' strategies.

2. SPADES

Most of the work in this paper is not specific to any particular game. But, because the game of Spades is well-known, and it is easy to create small, concrete examples from this game, we will use it as the primary domain for examples and experiments. There are many games similar to Spades which have similar properties, which we will not cover here, including Oh Hell! and Sergeant Major. The complete rules of Spades is found in [4], but many more games are described in detail at http://www.pagat.com/.

Spades is a card game for two or more players. In the 4-player version players play in teams, while in the 3-player version each player is on their own, which is what we focus on. We will only cover a subset of the rules of Spades here. A game is split into many hands. Within each hand the basic unit of play is a trick. At the beginning of a hand, players must bid how many tricks they think they can take in that hand. At the end of each hand they receive a score based on how many tricks they actually took. The goal of the game is to be the first player to reach a pre-determined cumulative score, often 300.

If players make their bid exactly, they receive a score of $10 \times bid$. Any tricks taken over their bid are called overtricks. If a player takes any overtricks they count for 1 point each, but each time a player accumulate 10 overtricks, they lose 100 points. Finally, if a player misses their bid, they lose $10 \times bid$. So, if a player bids 3 and takes 3, they get 30 points. If they bid 3 and take 5, they get 32 points. If they bid 3 and take 2, they get -30 points. Thus, the goal of the game is to make your bid without taking too many overtricks. In this work we consider the perfect-information version of Spades where we can see our opponents cards.

The perfect-information version of Spades is a complex and challenging problem in itself. If we wish to play the imperfect-information version of Spades, Monte-Carlo sampling can be used to turn a perfect-information player into a imperfect-information player, such as was done in Bridge [5].

To demonstrate the effect opponent modeling has on quality of play, we consider two player types, one player who tries to maximize the number of tricks they take in a hand (ignoring the bid), which we call MT. The other player type we consider attempts to minimize the number of overtricks they take, which we call mOT. Then, we varied these players by either giving them a model of

Table 1: The six ways to arrange two player types, A and B, in a three-player game.

	Seat 1	Seat 2	Seat 3
1	A	A	B
2	A	B	A
3	A	B	B
4	B	A	A
5	B	A	B
6	B	B	A

Table 2: The effect of opponent models on play.

	Players A v. B	Player A Score	Player A %Win	Player B Score	Player B %Win
(a)	mOT_{MT} v. MT_{mOT}	248.6	74.7	163.8	25.3
(b)	mOT_{MT} v. MT_{MT}	235.4	59.0	199.2	41.0
(c)	mOT_{mOT} v. MT_{mOT}	198.2	53.5	191.4	46.5
(d)	mOT_{mOT} v. MT_{MT}	178.2	44.0	207.3	56.0
(e)	mOT_{MT} v. mOT_{mOT}	212.4	37.2	250.8	62.8
(f)	MT_{mOT} v. MT_{MT}	157.3	36.0	218.4	64.0

their opponent, which we designate in the subscript. So MT_{MT} is a player who is attempting to maximize tricks, and assumes the same about all opponents. MT_{mOT} is also trying to maximize tricks, but assumes that all opponents are trying to minimize their overtricks. Players do not do any further recursive opponent modeling.

We arranged the experiments as follows. For each combination of the two player types and opponent models we played 100 games, each of which ended after any player reached 300 points. In each hand players were dealt seven cards from a standard 52 card deck and searched the game trees in their entirety. A rule-based system was used to determine player bids, and all players used the same system. Because there are three players in each game, each game was repeated with the same cards six times to account for different arrangements of players at the table (See Table 2). Although the specific results will vary based on the number of cards, the score limit for ending games, and the rule-system used for bidding, similar trends are found in the data regardless of these parameters.

We report the results of play in Table 2. Each row shows the average score and percentage of wins for the two listed player types averaged over the 600 games played[3] (100 games for each of the six combinations of player types). In each game two of the players will be of the same type and thus that type will have an increased chance of winning the game. This is offset by the fact that in the other half of the games that type will only be one of the three players and so have a lower chance of winning. In summary, if the player types were equal, they would be expected to win half of the games. Similarly, even if a player type wins 100% of the games, its average score would likely be less than 300 because in half of the games one of the losing players must be of the same type as the winner and presumably have a score less than 300.

By the rules of the game, we would expect the player who is minimizing overtricks to do best, and in general they do. This is particularly noticeable when they have a correct model of their opponent as shown in rows (a) and (b).[4] When both players have

[3]All results reported in this paper have 95% confidence of being within 7.2 points or less, which is sufficient to support our claims.

[4]Due to the nature of our experimental setup, one-third of a players' opponents will actually be of their own type. Hence, when we say player A has a correct model of its opponent, such as in row (a), we mean that at least two-thirds of its opponents are correctly modeled.

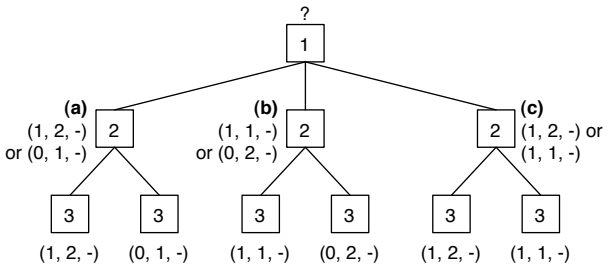


Figure 2: Example search tree in Spades.

correct models of each other (a), the mOT player wins 74.7% of the games. If the MT player does not have a correct model of the mOT player (b), the win rate drops to 59.0%. However, if the mOT player also has an incorrect model, the win rate drops to 44.0% and the preferred evaluation strategy can actually lose. This is a high price to pay for an incorrect opponent model. Furthermore, we cannot just assume our opponent is trying to maximize tricks, because we will do quite poorly against another player who is trying to minimize his overtricks as shown in row (e).

Lastly, row (f) shows the same results of the importance of correct modeling when both players try to maximize their tricks. In this case the average scores are much lower than in other games. This occurs because both players are using poor strategies, which increases the variance and thus lowers the average score. Similar results, not reported here, have been seen in other card games.

Note that because we are only doing one level of opponent modeling, it is important that we both have a correct model of our opponent and that our opponent has a correct model of us. Otherwise our predictions of our opponents' behavior will be incorrect. This explains why the results from row (a) are different from row (b). In row (b), the MT_{MT} player will behave differently than the mOT_{MT} player predicted since it expects to be facing an MT opponent. These results indicate that errors in second-level modeling effects (i.e., my model of my opponent's model of me) are important, but of a smaller magnitude than first-level modeling errors.

3. Soft-Max^n

We propose a new algorithm, soft-max^n, which is more robust to unknown opponents. We first present the algorithm intuition in the context of our results in Spades, followed by a more formal presentation of the algorithm.

First let us briefly return to Figure 1. At node (c) there is a tie for Player 2, so either move can be chosen, resulting in a score of 6 or a score of 1 for Player 1. Because max^n only backs a single value up the tree, there is no way for Player 1 at the root to know the risk of taking move (c). Because a value that is a tie for one player is not necessarily a tie for the others, there is a risk any time we encounter a tie in the game tree. The first inclination may be to remove all ties in the game tree. If all players have a publicly known total ordering on possible outcomes in a game (i.e., there are no ties), the max^n algorithm will work "perfectly", in that this problem will never arise. In practice, however, players themselves may only have a partial ordering on outcomes. When there is a tie in the partial ordering, there may be no way to predict how the opponent will break that tie.

Even worse, consider Player 2's decisions in Figure 2, where there are no ties. In this example, both Player 1 and 2 have bid 1

The only situation where a player has a correct model for all of its opponents is player B in rows (e) and (f).

Table 3: Dominance relations in max^n sets.

(a)	(3, 4, 3)
(b)	(2, 1, 7), (4, 2, 4)
(c)	(4, 1, 5), (3, 2, 5)
(d)	(10, 0, 0), (0, 10, 0), (0, 0, 10)

trick, and we will ignore Player 3 for the time being. Consider the possible strategies for Player 2. If Player 2 is trying to maximize tricks, at nodes (a) and (c) the outcome (1, 2, -) is chosen, which allows Player 1 to make their bid, and at node (b) (0, 2, -) is chosen. If Player 2 is trying to minimize overtricks, (1, 1, -) will be chosen at nodes (b) and (c), allowing Player 1 to make their bid, but at node (a), Player 2 will choose (0, 1, -).

If we think Player 2 is trying to maximize tricks, move (a) and (c) will have the same value. Barring additional information in the tie-breaking rules, there is the possibility we will make the wrong move to (a), missing our bid where moving to (c) would have guaranteed our bid. We get similar results if we model Player 2 as trying to minimize overtricks.

It should be clear that there is a tension between two issues. We want to avoid ties in our game trees, because they introduce extra uncertainty in whether we are backing up the correct max^n value. However, avoiding ties requires detailed knowledge of our opponent's preferences, i.e., very specific opponent models. We've already shown, though, that overly specific models can result in poor performance when the opponent's play does not correspond to the chosen model. For example, in Spades it is safe to assume the opponent prefers making their bid to missing it, but it is not safe to assume an opponent will necessarily avoid taking overtricks. Our solution, then, is to use generic, less presumptuous, opponent models, but more intelligently handle the resulting increase in the number of ties. We will first describe the soft-max^n algorithm and its alternate tie-breaking mechanism. In the next section we will describe our formalization of generic opponent models.

3.1 Max^n Sets

Instead of passing a single max^n value up the game tree, the soft-max^n algorithm backs up sets of max^n values. A max^n set s contains max^n values $v_1, \ldots, v_k$ where each value v_i is a standard max^n tuple. We combine multiple max^n sets with the *union* operation. The union of two sets is a new set containing all max^n values that appear in each individual set.

We compare sets using a dominance relationship. A max^n set s_1 strictly dominates another max^n set s_2 with respect to some player i if and only if,

$$\forall v_1 \in s_1 \; \forall v_2 \in s_2 \quad v_1[i] > v_2[i].$$

Similarly the max^n set s_1 weakly dominates s_2 relative to Player i if and only if,

$$\begin{aligned} &\forall v_1 \in s_1 \; \forall v_2 \in s_2 \qquad v_1[i] \geq v_2[i] \quad \text{and} \\ &\exists v_1 \in s_1 \; \exists v_2 \in s_2 \qquad v_1[i] > v_2[i]. \end{aligned}$$

We demonstrate these concepts with the values in Table 3. From the perspective of Player 1, (c) weakly dominates (a), because Player 1's scores in (c), 4 and 3, are at least as large as the score of 3 in (a). From the perspective of Player 2, (a) strictly dominates both (b) and (c), while from Player 3's perspective (b) strictly dominates (a), but does not dominate (c). If each player has a minimum score of 0 and a maximum score of 10, the set (d) can neither dominate another set or be strictly dominated by another set.

3.2 The Soft-Max^n Algorithm

Given the definition of a max^n set and dominance relation, we define soft-max^n as played by player i as follows:

1. At a leaf node, the max^n set is a heuristic or exact evaluation function.
2. At an internal node in the tree, the max^n set of values for that node is the union of all sets from its children that are not strictly dominated with respect to player j.
3. At the root of the tree, player i can use any decision rule to select the best set from among the non-dominated moves.

At the root of the tree, there are any number of ways to decide which move to make. For instance, depending on the game state, we may prefer to maximize potential score, minimize risk, or play to receive a guaranteed outcome. For all examples in this paper, we take the move which has the max^n set with the highest average value. There are obvious arguments why this may be the incorrect approach as there's no reason to assume all elements of the set are equally likely, but it is a simple approach that works in practice, as we demonstrate in the next section. We are currently exploring more principled options for making decisions.

We can illustrate soft-max^n using the tree in Figure 2. Recall that we now use only a generic opponent model for each player, which means that we will only distinguish between outcomes where our opponent are choosing to miss or make their bid. Player 1 and 2 each bid one trick. Since Player 2 always takes at least one trick, instead of returning one or the other result at (a), (b), and (c), we return a max^n set containing both child values at these nodes. At the root, Player 1 can, for instance, calculate the worst-case (or average-case) score on each path. Player 1 is guaranteed to take one trick on branch (c), so unlike (a) and (b), this move can be made safely. Thus, under soft-max^n Player 1 can make a more informed decision about which move to make.

The max^n algorithm has been shown to calculate one of many possible equilibria in a game. By comparison, Soft-max^n calculates all equilibria in the game.

LEMMA 1. *Soft-max^n calculates a superset of all pure-strategy equilibria values at every node in a game tree.*

Proof. We provide a proof by induction. First consider the leaves of a game tree. There are only single values in the leaves, so they must be equilibria for those sub-trees. Next, assume that at the kth level in the tree we have a max^n set containing a superset of all equilibria for that subtree. We show by contradiction that at the $(k+1)$th level of the tree we still have a superset of equilibria in our max^n set.

Assume there is a node, n, at the $(k+1)$th level for which soft-max^n does not return a max^n set which contains a superset of all equilibria. This means there is some child c whose max^n set contains an equilibrium not backed up into the parent. But, this can only happen if all values in the max^n set of c are strictly dominated by another child of n. If this is the case, then any individual value in c will always be dominated, and thus no value in c can be part of any equilibrium strategy. This is a contradiction, concluding the proof. □

We can be certain that all equilibria of the game are included in the soft-max^n sets. We cannot guarantee however that additional values will not be included. Informally, the extra values calculated by soft-max^n will be possible outcomes in the game, if we allow players to change their strategies at nodes in the middle of a game where they are indifferent to the possible outcomes.

3.3 Effect on Quality of Play

To demonstrate the effectiveness of soft-max^n, we repeated the experiments reported in Table 2, but this time the player trying to minimize overtricks uses soft-max^n and a generic opponent model. That is, the opponents are only assumed to be trying to make their bid. We employed the same methodology as in the previous experiments, except that now mOT$_g$ refers to the fact that the mOT player has this generic opponent model. The new results are in Table 3.3.

The new "% Gain" column is the increase in win percentage by using a generic model instead of an incorrect model. The generic model results in a substantial increase in the worst-case performance of playing an unknown opponent. In every case except against another mOT player that has modeled it, it is now winning at least half the games. Thus, if we are playing a game where we are unsure of the strategies that our opponents are using, we are better off using a generic model than making the wrong assumption about how our opponents play. The new "% Loss" column shows the further increase in win percentage that could be attained by using a *correct* opponent model rather than the general one. This demonstrates there is still room for improvement if the player can learn a model of its unknown opponent through play. In the next section we present a formalization of opponent models and then show how a specific opponent model can be inferred from opponent decisions.

4. OPPONENT MODELING

In the previous section we showed the improvement gained by using a generic model of the opponent. In this section we formalize a notion of an opponent model and how generic models can be constructed. A simple way to distinguish between two players is based on their evaluation function they use at the leaves of a search tree, that is, their relative utility of the different possible outcomes in the game. If an opponent has a total ordering for the possible outcomes in a game, we can use this information to simulate the decisions they would make in a game. In practice, a player may be ambivalent between two similar states. Thus, we model an opponent with a partial ordering defining their preferences over the possible outcomes in the game.

Formally, an *opponent model*, is a directed graph where the vertices are possible outcomes of the game. Edges are then used to encode the preference relationship. We define an opponent model, O, only by its edge set. In particular, $(u, v) \in O$ (*i.e.*, is an edge in the graph) if and only if u is preferred to v by the player being modeled. The graph thus provides a partial ordering over game outcomes. We assume the graph is closed under transitivity. In other words, if $(u, v) \in O$ and $(v, w) \in O$ then $(u, w) \in O$. Since preferences are necessarily transitive, we can always add edges into any model to form its transitive closure.

Two example opponent models are shown for two-trick Spades in Figure 3. There are six possible ways the two tricks can be taken in the game, which are shown on the left side of the figure. If Player 1 bids one, the models for maximizing tricks and minimizing overtricks are shown on the right. If the first player is trying to maximize tricks, they will prefer outcome 6 where they get two tricks, to all other outcomes. Outcomes 4 and 5 where they get one trick, would be preferred to outcomes 1, 2 and 3, where they get none. If Player 1 instead wants to minimize overtricks, they will have a slightly different model. Player 1 would prefer outcomes 4 and 5 to all other outcomes, and outcome 6 to outcomes 1, 2, and 3. For many domains, like Spades, the number of possible outcomes is relatively small and can be enumerated in a graph. In domains where this is not possible, the graph can be represented and reasoned over implicitly.

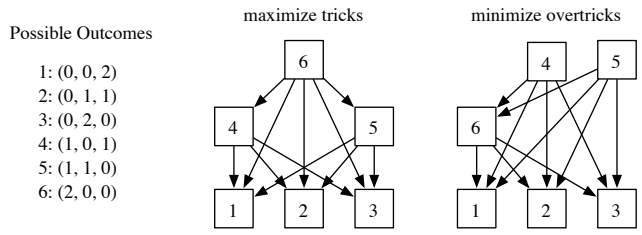


Figure 3: Two example opponent models for a tiny version of Spades.

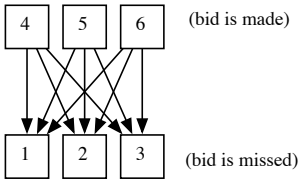


Figure 4: The *generalization* of the models from Figure 3.

It is likely that a player will not have a specific model of his opponent. The player, though, may have a set of candidate models, $O_1, \ldots, O_k$, believing that one of these models accurately captures the opponent's preferences. We saw in Table 2 that an incorrect model can have devastating effects on the quality of play. We also saw that soft-max^n with very generic assumptions about the opponents' preferences can greatly improve play. This suggests that given a set of candidate models we want to construct the *generalization* of these models. This generalization should (i) be as specific as possible, and (ii) be consistent with all of the candidate models. We can achieve this generalization by simple intersection:

$$G(O_1, \ldots, O_k) = \bigcap_{i=1}^{k} O_i.$$

Hence, $(u, v) \in G(O_1, \ldots, O_k)$ if and only if $\forall i\, (u, v) \in O_i$. In other words, u is preferred to v in the generalization if and only if u is preferred to v in all of the candidate models.

In Figure 4 we demonstrate the generalization of the models from Figure 3. In this case, the generalization of the two candidate models results in a model where the opponent prefers outcomes in which they make their bid, but it does not tell us anything beyond that.

4.1 Opponent Modeling and Soft-Max^n

Given an opponent model, it is easy to make use of it in soft-max^n by simply defining a dominance rule on max^n sets. Let s_1 and s_2 be max^n sets. We say s_1 strictly dominates s_2 under opponent model O, if and only if,

$$\forall u_1 \in s_1\ \forall u_2 \in s_2 \quad (u_1, u_2) \in O,$$

and weakly dominates, if and only if,

$$\begin{aligned} &\forall u_1 \in s_1\ \forall u_2 \in s_2 \qquad (u_2, u_1) \notin O \quad \text{and} \\ &\exists u_1 \in s_1\ \exists u_2 \in s_2 \qquad (u_1, u_2) \in O. \end{aligned}$$

These definitions of domination can now be used in the soft-max^n search for any given opponent model. Given a set of candidate opponent models, we can also construct the generalization of these models and use that model with soft-max^n.

Table 4: The effect of generalized opponent models on play.

	Players	Player A				Player B	
	A v. B	Score	%Win	%Gain	%Loss	Score	%Win
(a)	mOT_g v. MT_{mOT}	241.7	68.5	15.0	6.8	178.2	31.5
(b)	mOT_g v. MT_{MT}	218.2	53.5	9.5	5.5	206.1	46.5
(c)	mOT_g v. mOT_{MT}	242.2	54.8	4.8	8.0	228.7	45.2
(d)	mOT_g v. mOT_{mOT}	230.6	46.0	8.8	4.0	243.8	54.0

In domains where we cannot explicitly enumerate all possible outcomes in a game, we can instead use a functional model to provide the same information that we are getting from our explicit models in Spades.

4.2 Inferring Opponent Models

Without a specific opponent model we have shown that effective play can be obtained by generalizing a set of candidate models. During the course of a game, though, we actually observe our opponents' decisions. This opens up the possibility of inferring models of our opponents' preferences from observations of their choices. This is no easy task, as it is not generally possible from observations to distinguish between a player's preference over outcomes and simple indifference. Any decision of the opponent can always be explained as some tie-breaking mechanism on outcomes over which the player has no preference. Inference *ex nihilo* may be ill-defined, but inference based on a set of candidate models can still be done.

Given a set of candidate opponent models and a game with decisions by the opponents, we can identify which models are consistent with their actual decisions, and more importantly which are not. This will not identify which model best captures their preferences, but it can be used to eliminate models which certainly do not capture their preferences.[5] Eliminating these models will make the generalization of the remaining candidates more specific to the actual opponent. For example, near the end of a hand of Spades one may observe the opponent play their final cards so as to avoid taking a second overtrick, which they could have guaranteed taking. From this one can infer that the maximizing tricks model is inconsistent and eliminate it from the candidates. If maximizing tricks and minimizing overtricks were the only two candidates, one would then be certain the opponent minimizes overtricks. In later hands, this more specific model of the opponent can be used.

In order to perform this consistency check we need to reconstruct the actual decisions an opponent faced during the game. We can do this by running soft-max^n with generalized models for all of the players, including ourselves. Hence for any opponent decision we will have max^n sets, which we know contain the possible outcomes the opponent was deciding between. Let s_1 and s_2 be two sets the opponent decided between, where the actual decision was the action associated with s_1. This decision is definitely inconsistent with opponent model O if,

$$\forall u_1 \in s_1 \; \forall u_2 \in s_2 \quad (u_2, u_1) \in O.$$

We can perform this consistency check for every candidate model, for every opponent, for every decision that they made. For each opponent, we eliminate inconsistent models and recompute the generalization of the remaining models to be used in future soft-max^n searches against this opponent.

[5]This is essentially version-space learning [11] where the hypotheses are partial orderings and the training data are decisions the agent has made.

LEMMA 2. *If our opponent is described by some opponent model O_i and it is common knowledge that all players' preferences are consistent with the generalized model $G(O_1, \ldots, O_k)$, then our inference is sound, i.e., we will not eliminate O_i.*

Proof. By Lemma 1 we know that soft-max^n's values for each node in the tree is a superset of all equilibria consistent with the generalized opponent model. Let the opponent's decision for some node in the tree be between set s_1' and s_2', where s_1' was the set that was chosen. Let s_1 and s_2 be the soft-max^n sets for this node. Due to the common knowledge assumption, the soft-max^n sets must be supersets, i.e., $s_1' \subseteq s_1$ and $s_2' \subseteq s_2$. For purposes of contradiction, let s_2 strictly dominate s_1 under O_i, thus making the opponent's decision inconsistent with the model. By the definition of strict domination and the superset relation, since s_2 strictly dominates s_1 under O_i, then s_2' must strictly dominate s_1 and therefore s_2' must strictly dominate s_1'. Thus all outcomes in s_2' are preferred to s_1', and the opponent could not make this decision. □

4.3 Inference Experiments

To show the effectiveness of inference in Spades, we created 3 player types, which we refer to as MT, mOT and ML. As before, the MT player prefers outcomes where more tricks are taken. The mOT player tries to avoid overtricks. The ML player first tries to make their bid. But, among the set of hands where the bid is made or lost, the ML player tries to maximize the number of opponents that miss their bid.

In these games, one player was trying to infer the types of the other players only by observing their decisions. There are 9 ways to combine the 3 opponent types for the two opponents not doing inference, so we played 100 hands for each of these combinations for a total of 900 hands. Over these hands we were able to make 423 inferences about opponent types, although several times we made the same inference about player types multiple times in the same hand.

Because ML can look like both MT and mOT at times, it was the hardest to distinguish. We were only able to rule it out for a player in 51 of the overall hands. We were able to rule out mOT and MT, though, 106 and 103 times respectively. Combined, we successfully made an inference about an opponent in about one in six hands. Correct opponent inferences allow us to use a more specific model in soft-max^n resulting in 4–8% improvements in win percentage based on the results in Tables 2 and 3.3. The average number of hands per game in these results was 15, so this simple notion of consistency is fast enough to allow us to make game-altering inferences about our initially unknown opponents.

4.4 Generalizing Domains

Given that we have based so much of our discussion and experimentation on the game of Spades, it may not be immediately obvious how these results can be generalized. Spades makes a good example because we can easily divide the space of outcomes into two parts – those outcomes in which a player makes their bid, and

those outcomes in which they don't. Additionally, it is easy to enumerate all outcomes that can ever occur in a Spades game, and then reason over those outcomes.

But, consider the general case where each player has a utility in the range $\{0...1\}$. In this case there are, in theory, an infinite number of possible outcomes which we would like to reason over. In practice, however, we can reduce the number of possible outcomes using a clustering algorithm, or even more simply, just by dividing the space of possible utilities into discrete divisions. We can then assume that opponents are indifferent between outcomes within each cluster or division.

This is the opposite approach from what is normally taken in two-player zero-sum games. In these games more accurate utilities, often from increased depth of search, have been directly correlated with better play. But, in multi-player games we are seeing the opposite effect. That is, the more accurately we try to model our opponent's utilities, the more likely we are to make a mistake in our modeling. Thus, we want more generic models of our opponent that can tolerate small errors.

Thus, the benefits of soft-max^n may be even more apparent in domains where the number of possible outcomes is larger. Recent work in 4-player chess [9] suggests that max^n does not perform well in this domain because it does not consider minor collaborations which could lead to the max^n player losing the game very quickly. It would be interesting to see if soft-max^n with a generic opponent model could overcome these shortcomings.

5. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed a new algorithm, soft-max^n, for playing multi-player games. Soft-max^n avoids many of the drawbacks of the max^n algorithm, particularly in the face of unknown opponents. Additionally, we have shown how we can use soft-max^n to make inferences about the types of opponents we are playing.

This work forms a very promising foundation for future work in multi-player game-playing. There are still two outstanding issues that must be addressed before soft-max^n can result in human competitive play. The first is to develop less brittle inference. Although we prove the soundness of our technique, human play will not likely correspond to any of our preselected opponent models. In fact, it may not be consistent with *any* opponent model, as it may depend upon external, unobserved circumstances. Hence, our inference could actually eliminate all candidate models in the course of a few hands, leaving no opponent model available for use with soft-max^n. We are currently exploring the replacement of our symbolic reasoning with a more probabilistic approach to mitigate this problem. The second issue is the extension of soft-max^n to imperfect information, which is a common feature of multi-player games. Although Monte-Carlo sampling will allow us to handle the uncertainty during play, the effect of imperfect information on inference still needs to be addressed.

Finally, we are also working to understand the trade-off between the deeper search allowed by max^n pruning algorithms, and the more informed search performed by soft-max^n. It appears that both approaches have their strengths, depending on the situation at hand. Future work is still needed to better understand and classify the relative strengths of each approach.

6. ACKNOWLEDGMENTS

We wish to thank Darse Billings, Martin Müller, Jonathan Schaeffer and Akihiro Kishimoto for their ideas and discussion in relation to this research. This work was partially funded by the Alberta Ingenuity Centre for Machine Learning (AICML) and by Alberta's Informatics Circle of Research Excellence (iCORE).

7. REFERENCES

[1] D. Billings, A. Davidson, T. Schauenberg, N. Burch, M. Bowling, R. Holte, J. Schaeffer, and D. Szafron. Game tree search with adaptation in stochastic imperfect information games. In *Computers and Games*, 2004.

[2] D. Carmel and S. Markovitch. Incorporating opponent models into adversary search. In *AAAI-96*, pages 120–125, Aug. 4–8 1996.

[3] H. H. L. M. Donkers, J. W. H. M. Uiterwijk, and H. J. van den Herik. Probabilistic opponent-model search. *Inf. Sci.*, 135(3-4):123–149, 2001.

[4] A. H. M. (Editor), G. Mott-Smith, and P. D. Morehead. *Hoyle's Rules of Games, Third Revised and Updated Edition*. Signet Book, 2001.

[5] M. L. Ginsberg. GIB: Imperfect information in a computationally challenging game. *Journal of Artificial Intelligence Research*, 14:303–358, 2001.

[6] H. Iida, J. W. H. M. Uiterwijk, H. J. van den Herik, and I. S. Herschberg. Potential applications of opponent-model search. part 1, the domain of applicability. *ICCA Journal*, 16(4):201–208, 1993.

[7] P. J. Jansen. *Using Knowledge About the Opponent in Game-tree Search*. PhD thesis, Computer Science Department, Carnegie Mellon University, 1992.

[8] R. E. Korf. Generalized game trees. In *IJCAI-89*, pages 328–333, 1989.

[9] U. Lorenz and T. Tscheuschner. Player modeling, search algorithms and strategies in multi player games. In *Advances in Computer Games*, 2005.

[10] C. Luckhardt and K. Irani. An algorithmic solution of N-person games. In *AAAI-86*, volume 1, pages 158–162, 1986.

[11] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[12] J. Schaeffer, J. Culberson, N. Treloar, B. Knight, P. Lu, and D. Szafron. A world championship caliber checkers program. *Artificial Intelligence*, 53(2-3):273–290, 1992.

[13] F. Southey, M. Bowling, B. Larson, C. Piccione, N. Burch, D. Billings, and C. Rayner. Bayes' bluff: Opponent modelling in poker. In *Proceedings of the Twenty-First Conferece on Uncertainty in Artificial Intelligence*, pages 550–558, 2005.

[14] N. R. Sturtevant. Last-branch and speculative pruning algorithms for max^n. In *IJCAI-03*, pages 669–678, 2003.

[15] N. R. Sturtevant. Leaf-value tables for pruning non-zero sum games. In *IJCAI-05*, pages 317–323, 2005.

Probabilistic Policy Reuse in a Reinforcement Learning Agent

Fernando Fernández
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
fernando@cs.cmu.edu

Manuela Veloso
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
veloso@cs.cmu.edu

ABSTRACT

We contribute Policy Reuse as a technique to improve a reinforcement learning agent with guidance from past learned similar policies. Our method relies on using the past policies as a probabilistic bias where the learning agent faces three choices: the exploitation of the ongoing learned policy, the exploration of random unexplored actions, and the exploitation of past policies. We introduce the algorithm and its major components: an exploration strategy to include the new reuse bias, and a similarity function to estimate the similarity of past policies with respect to a new one. We provide empirical results demonstrating that Policy Reuse improves the learning performance over different strategies that learn without reuse. Interestingly and almost as a side effect, Policy Reuse also identifies classes of similar policies revealing a basis of *core* policies of the domain. We demonstrate that such a basis can be built incrementally, contributing the learning of the structure of a domain.

1. INTRODUCTION

We introduce Policy Reuse as a technique for reinforcement learning guided by past policies. Policy Reuse balances among exploitation of the ongoing learned policy, exploration of random actions, and exploration of the past policies. The exploration versus exploitation tradeoff defines whether to explore unseen parts of the space or to exploit the knowledge already acquired during the learning process. Several strategies, such as ϵ-greedy, Boltzmann or directed exploration [16], have been developed to balance this tradeoff. However these strategies only exploit knowledge obtained in the current learning process.

Several methods aim at improving learning by introducing additional knowledge into the exploration choices. Advice rules [7] define the actions to be preferred in different sets of states. The user provides the advice rules. Different knowledge sources can be used, for example through learning policies by imitation [10]. Other algorithms introduce previous knowledge through macro-actions or sub-policies. Macro-actions can be used to learn new action policies in Semi-Markov Decision Processes (SMDPs) [18]. Options can also be used in SMDPs by learning their behavior on line [13].

Hierarchical RL uses different abstraction levels to organize subtasks [3], and some approaches are able to learn such a hierarchy [6]. The methods for learning hierarchies or options capture the structure of the domain. Some related algorithms are SKILL [17], which discovers partially defined policies that arise in the context of multiple tasks in the same domain, and L-Cut, which discovers subgoals and corresponding sub-policies [12]. Sub-policies can suboptimally solve a task with computable bounds [1].

Transfer learning, as knowledge reuse across different learning task, can be performed by initializing the Q-values of a new episode with previously learned Q-values [2, 8]. However if the source and target tasks are very different, transfer learning requires expert knowledge to decide on the feasibility of the transfer, and on the mapping between actions and states from the source and target tasks [14, 15]. Value function transfer is an alternative but it is restricted to previous learning processes performed also through a value function.

In this paper we contribute Policy Reuse, a reinforcement learning method in which learned policies are saved and reused for similar tasks. The main algorithm of Policy Reuse is the PRQ-Learning algorithm, a method to probabilistically reuse a set of past policies that solve different tasks within the same domain. The PRQ-Learning algorithm has two main ideas. First, we introduce the π-reuse exploration strategy, which is able to probabilistically bias the exploration to include a given predefined past policy; and second, we introduce the similarity function that allows the estimation of the usefulness of past policies with respect to learning a new task.

We further present the PLPR algorithm, an incremental method to build a library of policies. When solving a new problem by policy reuse, the PLPR algorithm determines how different the learned policy is from the past policies as a function of the effectiveness of the reuse. If the past and new policies are "sufficiently" different, PLPR decides to add the new policy to the library of policies. Otherwise, it does not. PLPR is therefore capable of identifying a set of "core" policies that need to be saved to solve any new task in the domain within a threshold of similarity, δ. Given δ, our algorithm identifies a set of "δ-core-policies," as the basis or learned structure of the domain. Thus our method

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

to build the Policy Library has a novel "side-effect" in terms of learning the structure of the domain, i.e., the basis or the core policies of the domain.

In summary, Policy Reuse contributes to the overall goal of a lifelong reinforcement learning agent, as (i) it provides a mechanism to reuse past policies; (ii) it incrementally builds a policy library; and (iii) it learns an abstract domain structure in terms of core policies of the domain.

The paper is organized as follows. Section 2 introduces Policy Reuse for a reinforcement learning agent. Section 3 defines the π-reuse exploration strategy and presents experiment results in a large robot navigation domain. Section 4 formalizes the similarity metric among policies and introduces the PRQ-Learning algorithm. Section 5 presents the PLPR algorithm. Lastly, Section 6 draws conclusions.

2. POLICY REUSE IN REINFORCEMENT LEARNING

Reinforcement Learning problems are typically formalized using Markov Decision Processes (MDPs). An MDP is a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} >$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{T}$ is a stochastic state transition function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \Re$, and $\mathcal{R}$ is a stochastic reward function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \Re$. RL assumes that $\mathcal{T}$ and $\mathcal{R}$ are unknown.

We focus in RL domains where different *tasks* can be solved. The MDP's formalism is not expressive enough to represent all the concepts involved in knowledge transfer [11], so we define domain and task separately to handle different tasks executed in the same domain. We introduce a task as a specific reward function, while the other concepts, $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{T}$ stay constant for all the tasks in the same domain. We characterize a domain, $\mathcal{D}$, as a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{T} >$. We define a task, Ω, as a tuple $< \mathcal{D}, \mathcal{R}_\Omega >$, where $\mathcal{D}$ is a domain as previously defined, and $\mathcal{R}_\Omega$ is the stochastic and unknown reward function.

Definition 1. *A Domain $\mathcal{D}$ is a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{T} >$, where $\mathcal{S}$ is the set of all states; $\mathcal{A}$ is the set of all actions; and $\mathcal{T}$ is a state transition function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \Re$.*

Definition 2. *A task Ω is a tuple $< \mathcal{D}, \mathcal{R}_\Omega >$, where $\mathcal{D}$ is a domain; and $\mathcal{R}_\Omega$ is the reward function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \Re$.*

We assume that we are solving episodic tasks with absorbing goal states. Thus if s_i is a goal state, the probability of transitioning to the same state is 1 ($\mathcal{T}(s_i, a, s_i) = 1$), hence the transition probability to a different state is 0 ($\mathcal{T}(s_i, a, s_j) = 0$ for $s_i \neq s_j$), and the immediate reward is 0 ($\mathcal{R}(s_i, a) = 0$, for all $a \in \mathcal{A}$).

A trial or episode starts by locating the learning agent in a random position in the environment. Each episode finishes when the agent reaches a goal state or when it executes a maximum number of steps, H. The agent's goal is to maximize the expected average reinforcement per episode, W, as defined in equation 1:

$$W = \frac{1}{K} \sum_{k=0}^{K} \sum_{h=0}^{H} \gamma^h r_{k,h} \quad (1)$$

where γ ($0 \leq \gamma \leq 1$) reduces the importance of future rewards, and $r_{k,h}$ defines the immediate reward obtained in the step h of the episode k, in a total of K episodes.

An action policy, Π, is a function $\Pi : \mathcal{S} \rightarrow \mathcal{A}$ that defines how the agent behaves. If the action policy was created to solve a defined task, Ω, we call that action policy Π_Ω. The gain, or average expected reward, received when executing an action policy Π in the task Ω is called W_Ω^Π. Lastly, an optimal action policy for solving the task Ω is called Π_Ω^*. The action policy Π_Ω^* is optimal if $W_\Omega^{\Pi_\Omega^*} \geq W_\Omega^\Pi$, for all policy Π in the space of all possible policies when $K \rightarrow \infty$. Action policies can be represented using the action-value function, $Q^\Pi(s, a)$, which defines for each state $s \in \mathcal{S}$, $a \in \mathcal{A}$, the expected reward that will be obtained if the agent starts to act from s, executing a, and after it follows the policy Π. So, the RL problem is mapped to learning the function $Q^\Pi(s, a)$ that maximizes the expected gain. The learning can be performed using different algorithms, such as Q-Learning [19].

The goal of Policy Reuse is to use different policies, which solve different tasks, to bias the exploration process of the learning of the action policy of another similar task in the same domain. We call Policy Library to the set of past policies, as defined next.

Definition 3. *A Policy Library, L, is a set of n policies $\{\Pi_1, \ldots, \Pi_n\}$. Each policy $\Pi_i \in L$ solves a task $\Omega_i =< \mathcal{D}, \mathcal{R}_{\Omega_i} >$, i.e., each policy solves a task in the same domain.*

The previous definition does not restrict the characteristics of the tasks (they may be repeated), nor the characteristics of the policies (they may be sub-optimal). The scope of Policy Reuse is summarized as: we want to solve the task Ω, i.e., learn Π_Ω^*; we have previously solved the set of tasks $\{\Omega_1, \ldots, \Omega_n\}$ with n policies stored as a Policy Library, $L = \{\Pi_1, \ldots, \Pi_n\}$; how can we use the policy library, L, to learn the new policy, Π_Ω^*?

Policy Reuse answers this question by adding the past policies into a learning episode as an probabilistic exploration bias. We define an exploration strategy able to bias the exploration process towards the policies of the Policy Library, and a method to estimate the utility of reusing each of them and to decide whether to reuse them or not. Furthermore, Policy Reuse provides an efficient method to construct the Policy Library. We now detail the Policy Reuse approach.

3. REUSING A PAST POLICY

We now describe how one policy that solves a particular task can be used to bias the learning of an action policy for a similar task. Formally, we want to solve a new task Ω, i.e., learn Π_Ω^*. We assume that we are given a Policy Library, say $L = \{\Pi_1, \ldots, \Pi_n\}$ and that there is an oracle that returns Π_{past} as the most similar past policy to Π_Ω^*. We present how our algorithm reuses the policy Π_{past}.

Section 4 later defines a similarity metric between policies for different tasks, and introduces how to automatically estimate the policy to reuse.

3.1 The π-reuse Exploration Strategy

The π-reuse strategy is an exploration strategy able to bias a new learning process with a past policy. Let Π_{past} be the past policy to reuse and Π_{new} the new policy to be learned. We assume that we are using a direct RL method to learn the action policy, so we are learning the related Q function. Any RL algorithm can be used to learn the Q function, with the only requirement that it can learn off-policy, i.e., it can learn a policy while executing a different one, as Q-Learning does [19].

The goal of π-reuse is to balance random exploration, exploitation of the past policy, and exploitation of the new policy, as represented in Equation 2.

$$a = \begin{cases} \Pi_{past}(s) & \text{w/prob. } \psi \\ \epsilon - greedy(\Pi_{new}(s)) & \text{w/prob. } (1-\psi) \end{cases} \quad (2)$$

The π-reuse strategy follows the past policy with probability ψ, and it exploits the new policy with probability of $1-\psi$. As random exploration is always required, it exploits the new policy with an ϵ-greedy strategy.

Table 1 shows a procedure describing the π-reuse strategy integrated with the Q-Learning algorithm. The procedure gets as an input the past policy Π_{past}, the number of episodes K, the maximum number of steps per episode H, and the ψ parameter. An additional υ parameter is added to decay the value of ψ in each step of the learning episode. The procedure outputs the Q function, the policy, and the average gain obtained in the execution, W, which will play an important role in similarity assessment, as the next sections present. The variable ψ_h keeps the value of $\upsilon^h\psi$ in each step of each episode.

π-reuse ($\Pi_{past}, K, H, \psi, \upsilon$).
Initialize $Q^{\Pi_{new}}(s,a) = 0$, $\forall s \in \mathcal{S}, a \in \mathcal{A}$
For $k = 0$ to $K-1$
Set the initial state, s, randomly.
Set $\psi_1 \leftarrow \psi$
for $h = 1$ to H
With a probability of ψ_h, $a = \Pi_{past}(s)$
With a probability of $1-\psi_h$, $a = \epsilon$-greedy$(\Pi_{new}(s))$
Receive the next state s', and reward, $r_{k,h}$
Update $Q^{\Pi_{new}}(s,a)$, and therefore, Π_{new}:
$Q^{\Pi_{new}}(s,a) \leftarrow (1-\alpha)Q(s,a)^{\Pi_{new}} +$
$\alpha[r + \gamma \max_{a'} Q^{\Pi_{new}}(s',a')]$
Set $\psi_{h+1} \leftarrow \psi_h \upsilon$
Set $s \leftarrow s'$
$W = \frac{1}{K}\sum_{k=0}^{K}\sum_{h=0}^{H}\gamma^h r_{k,h}$
Return W, $Q^{\Pi_{new}}(s,a)$ and Π_{new}

Table 1: π-reuse Exploration Strategy.

3.2 Experiments

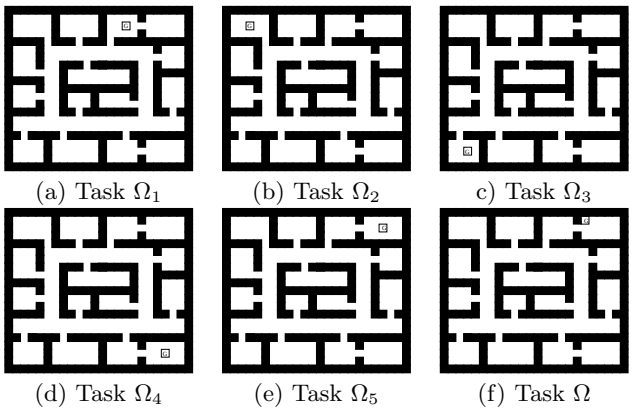
We describe the experiments performed to demonstrate the usefulness of the π-reuse exploration strategy. We first describe the experimental domain used.

3.2.1 Robot Navigation Domain

We use a grid-based robot navigational domain (see Figure 1) with multiple rooms. The environment is represented by walls, free positions and goal areas, all of them of size 1×1. The whole domain is $N \times M$ (24×21 in our case). The actions that the robot can execute are "North," "East," "South," and "West", all of size one. The final position after executing an action is noised by adding to the new position a random value that follows a uniform distribution in the range $(-0.20, 0.20)$.

Walls block the robot's motion, i.e., when the robot tries to execute an action that would crash it into a wall, the action keeps the robot in its original position.

The robot knows its location in the space through continuous coordinates (x, y). We assume that we have the optimal uniform discretization of the state space (which consists of 24×21 regions) [1]. The goal in this domain is to reach the area marked with 'G', in a maximum of H actions. When the robot reaches it, it is considered a successful episode, and it receives a reward of 1. Otherwise, it receives a reward of 0.



(a) Task Ω_1 (b) Task Ω_2 c) Task Ω_3

(d) Task Ω_4 (e) Task Ω_5 (f) Task Ω

Figure 1: Grid-based Office Domain

Figure 1 shows 6 different tasks, Ω_1, Ω_2, Ω_3, Ω_4, Ω_5 and Ω, given that the goal states, and therefore, the reward functions, are different. All these tasks are used in the experiments described in the next sections.

We choose the robot navigation domain for experimentation because it has been widely used in transfer learning papers (e.g., [17, 8, 11]). Transfer learning in more complex domains, as the Keepaway task in robot soccer, requires a mapping between tasks that use different state and action spaces [15, 14]. Such mapping requires a considerable amount of expert knowledge. Our ongoing research line includes the study of how our policy reuse algorithm can be extended to include transfer learning knowledge.

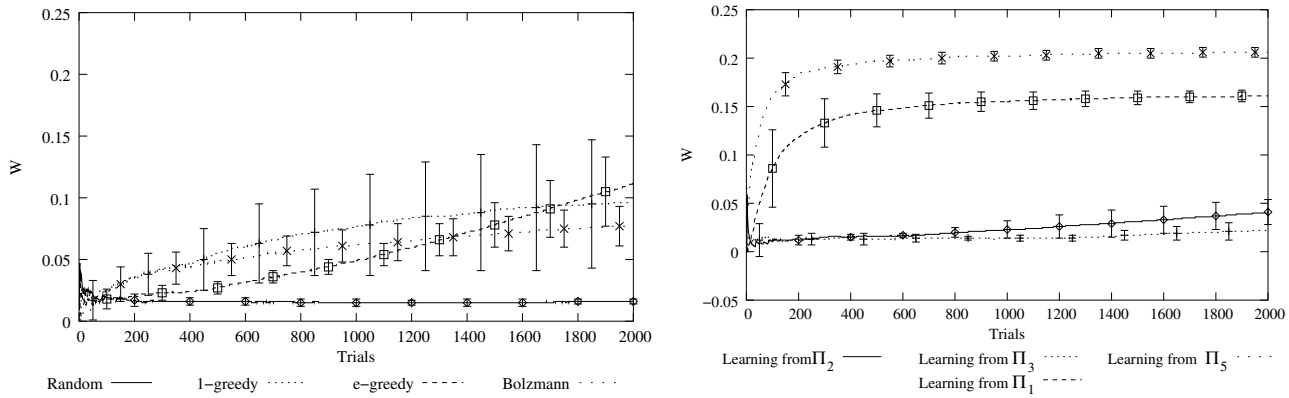
3.2.2 Results

We describe the experimental results of applying different exploration strategies for learning the task Ω, shown in Figure 1(f). Learning has been performed using the Q-Learning algorithm, for fixed parameters of $\gamma = 0.95$ and $\alpha = 0.05$, which empirically have demonstrated to be accurate for learning in this domain.

We analyze the learning performance in executing $K = 2000$ episodes. Each episode consists of following the defined strategy until the goal is achieved or until the maximum number of steps ($H = 100$) is reached. The x axis and the y axis show the episode or trial number and the average gain obtained respectively. A value of 0.2 for the episode 200 means that the average gain obtained in the first 200 episodes is 0.2. The results provided are the average of ten executions. Error bars provide the standard deviation in the ten executions.

The learning process has been first executed following different exploration strategies that do not use any past policy. Specifically, we have used four different strategies: (i) random; (ii) completely greedy; (iii) an ϵ-greedy (i.e., with probability ϵ follows the greedy strategy, and with probability $(1-\epsilon)$ acts randomly), with an initial value of $\epsilon = 0$, which is incremented by 0.0005 in each episode; (iv) the

[1] Different methods for function approximation have been successfully applied on this domain [4]. We have simplified the state space representation to a uniform discretization to focus on the study of Policy Reuse.



(a) Learning from Scratch (b) Learning with π-Reuse

Figure 2: Results of the learning process for different exploration strategies.

Boltzmann strategy ($P(a_j) = \frac{e^{\tau Q(s,a_j)}}{\sum_{p=1}^{n} e^{\tau Q(s,a_p)}}$), initializing $\tau = 0$, and increasing it by 5 in each learning episode.

Figure 2(a) shows the results. When acting randomly, the average gain in learning is almost 0, given that acting randomly is a very poor strategy. However when a greedy behavior is introduced, (strategy 1-greedy), the curve shows a slow increment, achieving values of almost 0.1. The curve obtained by the Boltzmann strategy does not offer significant improvements. The ϵ-greedy strategy seems to compute an accurate policy in the initial episodes, and it corresponds to the highest average gain at the end of the learning.

We then execute the learning with policy reuse. Figure 2(b) shows the learning curves of different learning processes performed with the π-reuse exploration strategy. In each of them, a different policy has been reused. In the first one, the policy reused is Π_5 ($\Pi_{past} = \Pi_5$), which is optimal to solve the task Ω_5. Such task has its goal into the same room as the goal of Ω. In the second case, $\Pi_{past} = \Pi_1$, which solved Ω_1. In the last two cases, $\Pi_{past} = \Pi_2$ and Π_3 respectively, whose associated tasks (Ω_2 and Ω_3) are very different when compared with Ω. All the reused policies are optimal for their respective tasks.

The parameters used in the Q-Learning update equation are the same as above ($\gamma = 0.95$ and $\alpha = 0.05$). The parameter setting for the π-reuse exploration strategy are $\psi = 1$, $\upsilon = 0.95$, and $\epsilon = 1 - \psi_h$. [2]

Figure 2(b) shows how, when biasing the exploration process for learning the task Ω with the policies Π_1 and Π_5, the obtained gain significantly increases within the first few episodes of the execution. For instance, when reusing Π_5, in only 100 iterations the average gain is higher than 0.15, and after 400 iterations the value stays around 0.2. When reusing Π_1, the gain is higher than 0.1 after only 200 episodes, and after 500 episodes it stays around 0.15. In both cases, the standard deviation is high in the initial episodes, but it approaches 0 in subsequent episodes.

However when the learning is biased with a very different policy, as Π_2 and Π_3, the average gain shown in Figure 2(a) is below 0.05, so the learning process is even worse than when learning from scratch.

The results show that reusing a past policy provides a bias in the exploration process which can speed up the learning when compared with learning from scratch. The improvement depends on whether the reused policy solves a task similar to the one we are currently learning. A similarity metric between tasks is hard to define even for a well structured domain.

Interestingly, we introduce a similarity metric between policies based on the usefuness of reuse. For instance, the results in Figure 2(b) show that the learning curves provide us information on the similarity between policies. In that figure, the gain obtained for each of the past policies can be understood as: (i) an estimation of how similar the policy reused is to the one we are currently learning; and (ii) an estimation of how useful the policy reused is in order to learn the new policy. Actually, the gain obtained by each past policy can be used to rank the similarity of the past policies with respect to the new one. In this case, the most similar policy to Π_Ω is Π_5, followed by Π_1, Π_2 and Π_3.

Furthermore, the gain estimated above can be computed very fast, and in only 25 episodes, the gain of reusing the policy Π_5 significantly outperforms the gain of reusing the other policies.

4. REUSING A LIBRARY OF POLICIES

We describe now the PRQ-learning algorithm for the efficient reuse of the policies stored in a Policy Library. We describe a similarity function that estimates the usefulness of reusing a specific policy for learning a new one.

4.1 A Similarity Function Between Policies

The exploration strategy π-reuse, as defined in Table 1, returns the learned policy Π_{new}, and the average gain obtained in its learning process, W. Let W_i be the gain obtained while executing the π-reuse exploration strategy, reusing the past policy Π_i. We can use such value to measure the usefulness of reusing the policy Π_i to learn the new policy Π_{new}. The next definitions formalize this idea.

Definition 4. *Given a policy Π_i that solves a task $\Omega_i =< \mathcal{D}, R_i >$, and a new task $\Omega =< \mathcal{D}, R_\Omega >$, the Reuse Gain of*

[2]This parameter setting has been chosen after an informal experimentation, so different values of the parameters could provide better results. However they are good enough to evaluate the algorithm and to move on the next steps of the research on Policy Reuse. Additional information about the properties of this parameter setting can be found in [5].

the policy Π_i on the task Ω, W_i, is the gain obtained when applying the π-reuse exploration strategy with the policy Π_i to learn the policy Π.

Then the most useful policy to reuse, Π_k, from a Library Policy, $L = \{\Pi_1, \ldots, \Pi_n\}$, is the one that maximizes the Reuse Gain when learning such a task, as defined in equation 3:

$$\Pi_k = \arg_{\Pi_i} \max(W_i), i = 1, \ldots, n \quad (3)$$

To solve this equation we need to compute the Reuse Gain for all the past policies. Interestingly, such a gain can be estimated on-line at the same time that the new policy is computed. This idea is formalized in the PRQ-Learning algorithm.

4.2 The PRQ-Learning Algorithm

The goal of the PRQ-learning algorithm is to solve a task Ω, i.e. to learn an action policy Π_Ω. We have a Policy Library $L = \{\Pi_1, \ldots, \Pi_n\}$ composed of n past optimal policies that solve n different tasks respectively. Then two main questions need to be answered: (i) given the set of policies $\{\Pi_\Omega, \Pi_1, \ldots, \Pi_n\}$, which consists of the policies in the Policy Library plus the ongoing learned policy, what policy is exploited? (ii) once a policy is selected, what exploration/exploitation strategy is followed?

The answer to the first question is as follows: let W_i be the Reuse Gain of the policy policy Π_i on the task Ω. Also, let W_Ω be the average reward that is received when following the policy Π_Ω greedily. The solution we introduce consists of following a softmax strategy using the values W_Ω and W_i, as defined in equation 4, with a temperature parameter τ. This value is also computed for Π_0, which we assume to be Π_Ω. Equation 4 provides a way to decide whether to exploit the past policies or the new one.

$$P(\Pi_j) = \frac{e^{\tau W_j}}{\sum_{p=0}^{n} e^{\tau W_p}} \quad (4)$$

The answer to the second question (what exploration strategy to follow once a policy is chosen) is an heuristic that depends on the selected policy. If the policy chosen is Π_Ω, the algorithm follows a completely greedy strategy. However, if the policy chosen is Π_i (for $i = 1, \ldots, n$), the π-reuse action selection strategy, defined in previous section, is followed instead. In this way, the Reuse Gain of each of the past policies can be estimated on-line with the learning of the new policy. Thus, the values required in Equation 4 are continuously updated each time a policy is used.

All these ideas are formalized in the PRQ-Learning algorithm (Policy Reuse in Q-Learning) shown in Table 2. The algorithm gets as input: a new task to solve Ω; the policy library L; the temperature parameter of the softmax policy selection equation τ, and a decay parameter $\Delta\tau$; and a set of previously defined parameters: $K, H, \psi, \upsilon, \gamma, \alpha$.

The algorithm initializes the new Q function to 0, as well as the estimated reuse gain of the policies in the library. Then the algorithm executes the K episodes iteratively. In each episode, the algorithm decides which policy to follow. In the first iteration, all the policies have the same probability to be chosen, given that all W_i values are initialized to 0. Once a policy is chosen, the algorithm uses it to solve the task, updating the Reuse Gain for such a policy with the reward obtained in the episode, and therefore, updating the probability to follow each policy. The policy being learned can also be chosen, although in the initial steps it behaves as a random policy, given that the Q values are initialized to 0. While new updates are performed over the Q function, it becomes more accurate, and receives higher rewards when executed. After executing several episodes, it is expected that the new policy obtains higher gains than reusing the past policies, so it will be chosen most of the time.

PRQL($\Omega, L, \tau, \Delta\tau, K, H, \psi, \upsilon, \gamma, \alpha$)

- Given:
  1. A new task Ω we want to solve
  2. A Policy Library $L = \{\Pi_1, \ldots, \Pi_n\}$
  3. An initial value of the temperature parameter, τ, and an incremental size, $\Delta\tau$, for the Boltzmann policy selection strategy
  4. A maximum number of episodes to execute, K
  5. A maximum number of steps per episode, H
  6. The parameters ψ and υ for the π-exploration strategy
  7. The parameters γ and α for the Q-learning update equation
- Initialize:
  1. $Q_\Omega(s, a) = 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$
  2. Initialize W_Ω to 0
  3. Initialize W_i to 0
  4. Initialize the number of episodes where policy Π_Ω has been chosen, $U_\Omega = 0$
  5. Initialize the number of episodes where policy Π_i has been chosen, $U_i = 0, \forall i = 1, \ldots, n$
- For $k = 1$ to K do
  - Choose an action policy, Π_k, assigning to each policy the probability of being selected computed by the following equation (equation 4):
    $$P(\Pi_j) = \frac{e^{\tau W_j}}{\sum_{p=0}^{n} e^{\tau W_p}}$$
    where W_0 is set to W_Ω
  - Execute the learning episode k
    - If $\Pi_k = \Pi_\Omega$, execute a Q-Learning episode following a fully greedy strategy
    - Otherwise, use the π-reuse exploration strategy to reuse Π_k, i.e. call π-reuse($\Pi_k, 1, H, \psi, \upsilon$)
    - In any case, receive the reward obtained in that episode, say R, and the updated Q function, $Q_\Omega(s, a)$
  - Set $W_k = \frac{W_k U_k + R}{U_k + 1}$
  - Set $U_k = U_k + 1$
  - Set $\tau = \tau + \Delta\tau$
- Return the policy derived from $Q_\Omega(s, a)$

Table 2: PRQ-Learning.

4.3 Experiments

We present the experiments performed with the PRQ-Learning algorithm. We demonstrate three main claims: Firstly, that the performance can be improved if we can bias the exploration with past policies, even if we have several and we do not know apriori which one is the most similar and/or useful to reuse. Second, that it is possible to determine which is the most useful policy simultaneously while learning the new policy. And third, that a balance between

exploring, exploiting past policies, and exploiting the new policy being learned can be successfully achieved.

We use the PRQ-Learning algorithm for learning the task Ω, defined in Figure 1(f). We assume that we have 3 different libraries of policies, so we distinguish three different cases. In the first one, the policy library is $L_1 = \{\Pi_2, \Pi_3, \Pi_4\}$, assuming that the tasks Ω_2, Ω_3 and Ω_4, defined in Figure 1(b), (c) and (d) respectively, were previously solved. All these tasks are very different from the one we want to solve, so their policies are not supposed to be very useful in learning the new one. In the second case, Π_1 is added, so $L_2 = \{\Pi_1, \Pi_2, \Pi_3, \Pi_4\}$. The third case uses the Policy Library $L_3 = \{\Pi_2, \Pi_3, \Pi_4, \Pi_5\}$

The PRQ-Learning algorithm is executed for the three cases. The learning curves are shown in Figure 3. The parameters used are the same used in Section 3.2. The only new parameters are the ones of the Boltzmann policy selection strategy, $\tau = 0$, and $\Delta\tau = 0.05$, obtained empirically.

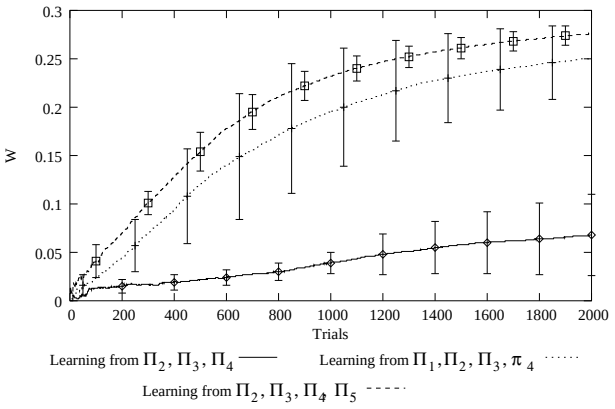


Figure 3: Learning curve when learning the task of Figure 1(f) reusing different sets of policies.

Figure 3 shows two main conclusions. Firstly, when a very similar policy is included in the set of policies to be reused, the improvement on learning is very high. For instance, when reusing Π_1 and Π_5, the average gain is greater than 0.1 in only 500 iterations, and more than 0.25 at the end of the episode. Secondly, when no similar policy is available, the learning curve is similar to the results obtained when learning from scratch with the 1-greedy strategy (which is the strategy followed by PRQ-Learning for the new policy, as defined by the PRQ-Learning algorithm). This demonstrates that the PRQ-learning algorithm has discovered that reusing the past policies is not useful, so it follows the best strategy available, which is to the 1-greedy strategy with the new policy.

In summary, we can say that the PRQ-learning algorithm has demonstrated to successfully reuse a predefined set of policies. The remaining issue consists of investigating how to acquire such a set of policies. The next section focuses on how an agent can build a library of policies.

5. BUILDING A LIBRARY OF POLICIES

We now describe the *PLPR* algorithm (Policy Library through Policy Reuse), an algorithm to build a library of policies. The algorithm is based on an incremental learning of policies that solve different tasks. The tasks to be solved are not known apriori, and are sequentially given. Otherwise, a method to learn them in parallel could be applied [9].

5.1 The PLPR Algorithm

The PLPR algorithm works as follows. Initially the Policy Library is empty, $PL = \emptyset$. When the first task Ω_1 is solved, the corresponding learned policy Π_1 is learned without reuse. Π_1 is added to the Policy Library and $PL = \{\Pi_1\}$. When a second task needs to be solved, the PRQ-Learning algorithm is applied reusing Π_1. Π_2 is learned. The algorithm makes a decision on whether to add Π_2 to the Policy Library or not. This decision is based on how similar Π_1 is to Π_2, following the similarity function we now introduce.

Definition 5. *Given a policy, Π_i that solves a task $\Omega_i =< \mathcal{D}, R_i >$, a new task $\Omega =< \mathcal{D}, R_\Omega >$, and its respective optimal policy, Π, Π is δ-similar to Π_i (for $0 \leq \delta \leq 1$) if $W_i > \delta W^*_\Omega$, where W_i is the Reuse Gain of Π_i on task Ω and W^*_Ω is the average gain obtained in Ω when an optimal policy is followed.*

The interesting aspect of this concept is that for any optimal policy Π, if we know a past policy which is δ-similar to it, we also know that such optimal policy can be easily learned just by applying the π-reuse algorithm with the past policy. The gain obtained in the learning process (the reuse gain) will be at least δ times the maximum gain in such a task. From this definition, we can formalize the concept of δ-similarity with respect to a Policy Library, L.

Definition 6. *Given a Policy Library, $L = \{\Pi_1, \ldots, \Pi_n\}$ in a domain $\mathcal{D}$, a new task $\Omega =< \mathcal{D}, R_\Omega >$, and its respective optimal policy, Π, Π is δ-similar with respect to L iff $\exists \Pi_i$ such as Π is δ-similar to Π_i, for $i = 1, \ldots, n$.*

If the algorithm finds that a policy Π is δ-similar with respect to a Policy Library L, then it knows that the policy Π can be easily learned by reusing the policies in L. Table 3 presents the PLPR algorithm, which is executed each time that a new task needs to be solved. It gets as an input the Policy Library and the new task to solve, and outputs the learned policy and the updated Policy Library.

PLPR Algorithm

- Given:
  1. A Policy Library, L, composed of n policies, $\{\Pi_1, \ldots, \Pi_n\}$
  2. A new task Ω we want to solve
  3. A δ parameter
- Execute the PRQ-Learning algorithm reusing L. Receive from this execution Π_Ω, W_Ω and W_{max}, where:
  - Π_Ω is the learned policy
  - W_Ω is the average gain obtained when the policy Π_Ω was followed
  - $W_{max} = \max W_i$, for $i = 1, \ldots, n$
- Update PL using the following equation:

$$L = \begin{cases} L \cup \{\Pi_\Omega\} & \text{if } W_{max} < \delta W_\Omega \\ L & otherwise \end{cases} \quad (5)$$

Table 3: PLPR Algorithm.

Equation 5 is the update equation for the Policy Library. It requires the computation of the most similar policy in the

library, which is the policy Π_j such as $j = \arg_i \max W_i$, for $i = 1, \ldots, n$. W_{max} is the gain obtained by reusing such a policy. The new policy learned is inserted in the library if W_{max} is lower than δ times the gain obtained by using the new policy (W_Ω), where $\delta \in [0, 1]$ defines the similarity threshold, i.e., whether the new policy is δ-similar with respect to the Policy Library.

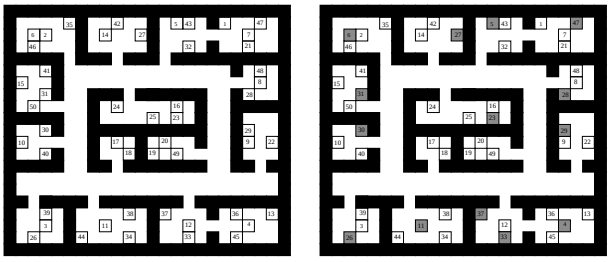
The parameter δ hence plays an important role. If $\delta = 0$, the Policy Library stores only the first policy learned, given that the average gain obtained by reusing it will be greater than zero in most cases, due to the positive rewards obtained by chance. If $\delta = 1$, most of the policies learned are inserted, as $W_{max} < W_\Omega$, given that W_Ω is maximum if the optimal policy has been learned. Different δ values in the range $(0, 1)$ provide different sizes of the library. Thus δ defines the "resolution" of the library.

The PLPR algorithm has an interesting side effect in terms of learning the structure of the domain. The Policy Library is initialized to empty, and a new policy is included only if it is different *enough* (depending on the δ threshold) from the previously stored ones. When the policies stored are *fully representative* of the domain, no more policies are stored. Therefore the obtained library can be considered as the *Basis-Library* of the domain, and the stored policies can be considered as the *core-policies* of such domain.

5.2 Experiments

We present the experiments performed to learn a Basis-Library in the navigation domain (Section 3). We perform a task consisting of the executing $K = 2000$ episodes. Like in the previous experiments, each episode consists of a sequence of actions until the goal is achieved or until the maximum number of actions, $H = 100$, is reached.

A number, 50, of different tasks are sequentially performed, each of them with a different goal area, located in different positions of the different rooms of the domain, as shown in Figure 4(a). The figure does not represent a unique task with 50 different goals, but the 50 different goal areas of the 50 different tasks. The results provided are the average of 10 different executions, in which the 50 different tasks are sequentially performed following a random order. In these experiments, we use the same parameter settings as earlier (see Section 4.3).



(a) 50 different goal areas (b) Core-Policies

Figure 4: Office Domain.

The first element to study is the size of the Policy Library built while performing the tasks with the PLPR algorithm, i.e., the number of core-policies stored in the Policy Library. Figure 5 shows in the y axis the size of the Policy Library, and in the x axis, the number of tasks performed up to that moment. As introduced above, when $\delta = 0$, only 1 policy is stored. When $\delta = 0.25$, the number of core-policies is around 14. Interestingly, this is very close to the number of rooms in the domain (15). While increasing δ, the number of core-policies increases and when $\delta = 1$, almost all the learned policies are stored.

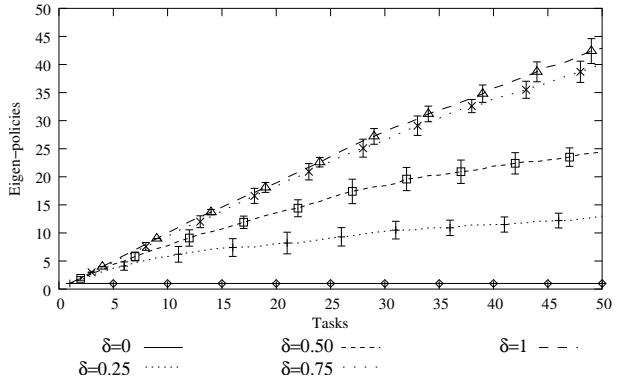


Figure 5: Number of eigen-policies obtained.

Figure 4(b) shows an example of the core-policies obtained in one of the executions with $\delta = 0.25$. The figure represents the resulting Policy Library composed of 14 core-policies. In the figure, we assume that a policy is represented by the goal area of the task that it solves. A core-policy is represented also by a shaded goal area. The figure demonstrates that one and only one core-policy has been learned for most of the rooms. The algorithm has discovered that if two different tasks are given two goal areas in the same room, their respective policies are very similar, so only one of them needs to be stored in the Policy Library. We therefore observe that the PLPR algorithm learns the *structure* of the domain as represented by the core-policies.

Figure 6 shows the average gain obtained when performing the 50 different tasks with the PLPR algorithm, for the different values of δ. In most of the cases, $\delta = 0.25, 0.50, 0.75$ and 1, the average gain increases up to more than 0.2, and no significant differences exist between them. Only in the case of $\delta = 0$, the average gain stays low, around 0.16, given that $\delta = 0$ generates a Policy Library with only one policy (the first one learned).

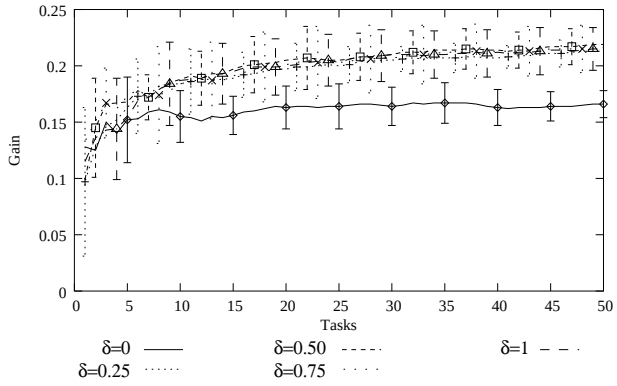


Figure 6: Results of PLPR.

For comparison purposes, we executed the same learning process with different exploration strategies as learning from scratch. In those cases, while new policies are learned from scratch, the average gain obtained stabilizes around 0.12 for all the strategies, without very significant differences. Therefore, Policy Reuse can almost reach a 100% gain in the performance of the 50 tasks over the results obtained when the 50 tasks are learned from scratch.

6. CONCLUSIONS

We have introduced different algorithms to address the main challenges of policy reuse in a reinforcement learning agent. First, the PRQ-Learning algorithm allows to probabilistically bias an exploration learning process by reusing a Policy Library; the algorithm significantly improves the learning performance over exploration strategies that learn from scratch. Second, the PLPR algorithm incrementally builds the Policy Library; the library is built at the same time as new policies are learned and past policies are reused. And last, our method to build the Policy Library allows the learning of the structure of the domain in terms of a set of core-policies.

Future ongoing research includes the extension of Policy Reuse to reuse across different representational frameworks, including different agents or domains. We envision multiple agents sharing and reusing policies previously learned to improve the solvability horizon of individual non-reuse based learning agents.

7. ACKNOWLEDGMENTS

This research was conducted while the first author was visiting Carnegie Mellon from the Universidad Carlos III de Madrid, supported by a generous grant from the Spanish Ministry of Education and Fullbright. He was partially sponsored also by the Ministry of Education and Science project number TIN2005-08945-C06-05 and by CAM-UC3M project number UC3M-INF-05-016. The second author was partially sponsored by Rockwell Scientific Co., LLC under subcontract no. B4U528968 and prime contract no. W911W6-04-C-0058 with the US Army, and by BBNT Solutions, LLC under contract no. FA8760-04-C-0002 with the US Air Force. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the sponsoring institutions, the U.S. Government or any other entity.

8. REFERENCES

[1] M. Bowling and M. Veloso. Bounding the suboptimality of reusing subproblems. In *Proceedings of IJCAI-99*, 1999.

[2] J. Carroll and T. Peterson. Fixed vs. dynamic sub-transfer in reinforcement learning. In *Proceedings of the International Conference on Machine Learning and Applications*, 2002.

[3] T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[4] F. Fernández and D. Borrajo. On determinism handling while learning reduced state space representations. In *Proceedings of the European Conference on Artificial Intelligence (ECAI 2002)*, Lyon (France), July 2002.

[5] F. Fernández and M. Veloso. Exploration and policy reuse. Technical Report CMU-CS-05-172, School of Computer Science, Carnegie Mellon University, 2005.

[6] B. Hengst. Discovering hierarchy in reinforcement learning with HEXQ. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.

[7] R. Maclin, J. Shavlik, L. Torrey, T. Walker, and E. Wild. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, July 2005. To appear.

[8] M. G. Madden and T. Howley. Transfer of experience between reinforcement learning environments with progressive difficulty. *Artificial Intelligence Review*, 21:375–398, 2004.

[9] R. B. Ollington and P. W. Vamplew. Concurrent Q-Learning: Reinforcement learning for dynamic goals and environments. *International Journal of Intelligent Systems*, 20:1037–1052, 2005.

[10] B. Price and C. Boutilier. Accelerating reinforcement learning through implicit imitation. *Journal of Artificial Intelligence Research*, 19:569–629, 2003.

[11] A. A. Sherstov and P. Stone. Improving action selection in MDP's via knowledge transfer. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, July 2005.

[12] Ö. Şimşek, A. P. Wolfe, and A. G. Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, 2005.

[13] R. S. Sutton, D. Precup, and S. Singh. Intra-option learning about temporally abstract actions. In *Proceedings of the Internacional Conference on Machine Learning (ICML'98)*, 1998.

[14] M. E. Taylor and P. Stone. Behavior transfer for value-function-based reinforcement learning. In *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, July 2005. To appear.

[15] M. E. Taylor, P. Stone, and Y. Liu. Value functions for RL-based behavior transfer: A comparative study. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, July 2005. To appear.

[16] S. Thrun. Efficient exploration in reinforcement learning. Technical Report C,I-CS-92-102, Carnegie Mellon University, January 1992.

[17] S. Thrun and A. Schwartz. Finding structure in reinforcement learning. In *Advances in Neural Information Processing Systems 7*. MIT Press., 1995.

[18] W. T. B. Uther. *Tree Based Hierarchical Reinforcement Learning*. PhD thesis, Carnegie Mellon University, August 2002.

[19] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

Learning from Induced Changes in Opponent (Re)Actions in Multi-Agent Games

P.J. 't Hoen hoen@cwi.nl
S.M. Bohte sbohte@cwi.nl
J.A. La Poutré hlp@cwi.nl
CWI, The Netherlands Centre for Mathematics and Computer Science
Kruislaan 413, NL-1098 SJ Amsterdam, The Netherlands

ABSTRACT

Multi-agent learning is a growing area of research. An important topic is to formulate how an agent can learn a good policy in the face of adaptive, competitive opponents. Most research has focused on extensions of single agent learning techniques originally designed for agents in more static environments. These techniques however fail to incorporate a notion of the effect of own previous actions on the development of the policy of the other agents in the system. We argue that incorporation of this property is beneficial in competitive settings. In this paper, we present a novel algorithm to capture this notion, and present experimental results to validate our claims.

General Terms

Multi-Agent Learning

Keywords

Multi-Agent RL, Opponent Modeling, Learning Reactions to Actions

1. INTRODUCTION

Acting intelligently in a dynamic environment shared with other competing agents is a hard task, as daily life as well as a host of work on learning in multi-agent settings demonstrates. Without the presence of a teacher, and with only the observed actions and rewards to learn from, the framework of Reinforcement Learning (RL) (Sutton & Barto, 1998) is an obvious choice. Conceptually, the focus moves from the realm of Markov Decision Problems to that of Game Theory (GT) and stochastic games (Shoham, Powers, & Grenager, 2004).

Recent work has proposed a number of extensions from the single agent setting to the multi-agent domain. State-of-the-art Multi-Agent Reinforcement Learning (MARL) algorithms improve on single-agent RL approaches by incorporating models of the *current* opponent agent's behavior. This allows an agent to model the current "environment" including opponents, against which it has to optimize its own behavior by playing a best-response. In a multi-agent setting, the policy that maximizes payoff will depend on the *changing* actions of adversaries. Importantly, the adaptations of adversaries are likely to be reactions to *ones own* actions. This point is largely ignored in current (MARL) algorithms. Such strategic reasoning can be paramount when an agent is repeatedly interacting with the same opponents, and earlier actions influence the future behavior of these learning adversaries.

The Prisoner's Dilemma (PD) is illustrative in that in repeated play the optimal policy differs from blindly playing the myopic best response, and that it can be profitable to have a good estimation of the opponents reactions to ones own play. In one single game, two competing agents can each choose from actions cooperate or defect ({C,D}), and the maximum *joint* payoff is achieved when both agents choose to Cooperate. However, when the other agents plays Cooperate, an agent obtains an even larger payoff from playing Defect. From a game theoretical perspective then, for the single shot game {D,D} is the dominant strategy and a Nash-Equilibrium (see Section 2).

In iterated play of the PD (iPD) game, there is no longer a clear dominant strategy, as both agents can achieve a higher aggregated *and* individual reward by cooperating and playing the joint action {C,C}, *provided* there is some strategic incentive to rarely unilaterally defect. That is: defecting as a strategy is discouraged by the likely *negative future* reaction of the opponent (see also the famous Folk Theorem discussed in Section 2).

It is important to notice that just modeling the *current* behavior of the opponent in iPD can lead to {D,D} outcome in repeated play when the reaction of the opponent to ones own play of defection is not taken into account. An agent that does not take into account the changing behavior of the opponent may estimate that a cooperating opponent may be exploited, and ignores the fact that defection will result in an opponent that also defects. When repeatedly playing a game against an adversary, the question is which moves one should play to maximize the individual reward given the *dependent* adaptive behavior of the adversary.

We argue that current state-of-the-art MARL algorithms (M. H. Bowling & Veloso, 2002; M. Bowling, 2004; Tesauro, 2003; Conitzer & Sandholm, 2003; Greenwald & Hall, 2003; Banerjee & Peng, 2004; Weinberg & Rosenschein, 2004) (see also Shoham et al. (2004) for a more in depth discussion) do not incorporate a sufficient notion of the longer term im-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

pact of their actions on the dynamics of the environment. This environment includes other adaptive agents that react to moves of their opponents as well. For example, current MARL algorithms that play the iPD will universally converge to the worst possible outcome of {D,D} in self-play[1](Crandall & Goodrich, 2005). The one exception to the full defection converge outcome in repeated play of the iPD we are aware of is a recent paper by Crandall and Goodrich (2005), which we will discuss further below.

In this paper, we present a novel MARL framework called Strategic Opponent Policy Modeling (StrOPM) that goes beyond current Myopic Opponent Policy Modeling (MOPM) MARL approaches. StrOPM, as most other MARL's, takes into account the current (estimated) policy of the opponents and the rewards for actions. Importantly, our algorithm also estimates how the policies of the opponents change over time due to actions played by StrOPM. The policy of the agent using the StrOPM algorithm is adjusted to increase the expected future reward by taking into consideration 1) the immediate and future rewards of actions 2) the estimated policy of the opponent, and 3), most importantly, the impact of the chosen actions on the future policy of the opponent. The last point is the novel contribution of our work.

Our algorithm is a policy-gradient like method with roots in Q-learning. Q-like values for states are calculated based on estimated values of actions and estimated opponent policies. Policies are updated along the gradient of expected increased rewards. To more accurately model the (future) changes in the environment due to an agent's own actions, the agent computes a continuously updated estimate of the *change* in the choice of opponents actions due to its own actions. StrOPM then adjusts the reward gradient by this estimate to account for the opponent's likely *re*action.

We present our results for the well studied domain of iterated-play two-player two-action matrix games. We show that our algorithm quickly arrives at the desired Nash-Equilibrium for games that are unproblematic. These games are relatively straightforward in the sense that optimal play in a single-shot play of the game is also a good solution to iterated play of the game.

Importantly, we show that the proposed StrOPM algorithm achieves cooperation ({C,C}) in repeated play of the PD. After an initial exploration period, StrOPM learns that defection leads to reciprocal defection, and an overall lower value of future play than a high level of cooperation. The agents learn to optimize their own interest as a function of the played game and the reciprocal behavior of the evolving opponent. The (possible) cooperation strategy is not a precoded option in the algorithm, but is achieved by optimizing beyond the immediate best response to the current opponent play. The StrOPM agents in self-play exhibit a learned Tit-For-Tat strategy (Axelrod, 1984). The Tit-For-Tat algorithm plays C until the opponent defects upon which a D is played followed by again C until the next defection to encourage cooperation. The StrOPM agents learn this strategy and cooperates to reach a good equilibrium, and yet also evolves a threat state where defection is returned in order to guard against exploitation by the opponent.

We also compare our approach to the recent algorithm M-Qubed of Crandall and Goodrich (2005) for the studied matrix games. M-Qubed achieves the (C, C) equilibrium in the iPD by a meta step. The M-Qubed algorithm is designed to balance play of a best-response strategy with the precoded strategy of playing one action consistently. In particular, the algorithm during play continuously considers the value of switching to the pure, precoded strategy of playing C. Using a precoded strategy may lead to being exploited, or be unsuitable for more complex settings, as we will show.

We present results for StrOPM versus M-Qubed and show that the two type of agents playing against each other can together achieve cooperation, but from entirely different principles. The StrOPM algorithm in self-play finds good strategies significantly faster than M-Qubed for the iPD game. When playing against each other, this high speed of learning leads to an initial period where StrOPM exploits M-Qubed. Furthermore, StrOPM is shown to exploit the M-Qubed algorithm in a simple matrix game where the pre-coded strategy of playing one action predictably can result in inferior strategies in repeated play. Using precoded strategies has limited application. For example, precoded strategies cannot be used when the number of possible precoded strategies must be large, the precoded strategies cannot be determined, or do not generalize to novel, or more complex settings. Fundamentally, a MARL should remain flexible in novel settings.

In summary, we have presented three important contributions. First of all, we have signaled the need for Multi-Agent learning algorithms to take into account the impact of their actions on the learned policy of the opponents. This concept is as yet lacking in the MARL community. Secondly, we presented a novel framework that allows agents to apply the above observation to achieve desired outcomes for representative iterated matrix games, and, more importantly, also for the iPD. The latter is known to be difficult for MARL approaches; the presented StrOPM framework solves the iPD game by learning and recognizing "threat states", where it will punish an opponent's uncooperative behavior with a Tit-for-Tat type strategy (and also learns the reciprocal reaction). Lastly, we demonstrate the danger of incorporating pre-coded fixed strategies in learning algorithms as these are vulnerable to exploitation or may not be suited to novel or more complex settings.

Structure. The remainder of this document is structured as follows. In Section 2, we present iterated matrix games, and the iterated Prisoners Dilemma. In Section 3, we present our general framework. We present a Multi-Agent RL framework, StrOPM, that in the limit plays best-response against stationary opponents. Importantly, StrOPM can reason strategically in games where it is relevant in how an opponent will react to the history of play. In Section 4, we present experimental results. In Section 5, we discuss and conclude.

2. MODEL AND PROBLEM SETTING

In this section, we give some introductory definitions and notation from Game Theory, Reinforcement Learning, and the iterated matrix games we study as problem domain. We discuss the iterated Prisoners Dilemma as a game of special interest.

2.1 Agents and Matrix games

We consider multi-agent Reinforcement Learning for iterated play of games, more specifically **matrix games**, and introduce some well-known concepts from Game Theory (GT).

[1]An agent playing against an identical opponent.

In general, let S denote the set of states in the game and let A_i denote the set of actions that agent/player i may select in each state $s \in S$. Let $a = (a_1, a_2, \ldots, a_n)$, where $a_i \in A_i$ be a joint action for n agents, and let $A = A_1 \times \cdots \times A_n$ be the set of possible joint actions. **Zero-sum games** are games where the rewards of the agents for each joint action sum to zero. **General sum games** allow for any sum of values for the reward of a joint action.

A **strategy (or policy)** for agent i is a probability distribution $\pi(\cdot)$ over its actions set A_i. Let $\pi(S)$ denote a strategy over all states $s \in S$ and let $\pi(s)$ (or π_i) denote a strategy in a single state s. A strategy may be a **pure strategy** (an agent selects an action deterministically) or according to a **mixed strategy** (a strategy that plays a random action, according a probability distribution). A **joint strategy** played by n agents is denoted by $\pi = (\pi_i, \ldots, \pi_n)$. Let a_{-i} and π_{-i} refer to the joint action and strategy of all agents except agent i.

We focus on the more restricted **matrix game**, defined by a set of matrices $R = \{R_1, \ldots, R_n\}$. Let $R(\pi) = (R_1(\pi), \ldots, R(\pi_n))$ be a vector of expected payoffs when the joint strategy π is played. Also, let $R_i(\pi_i, \pi_{-i})$ be the expected payoff to agent i when it plays strategy π_i and the other agents play π_{-i}. A strategy is **dominant** if, regardless of what any other players do, the strategy earns a player a larger payoff than any other strategy. Also, let $R_i(\begin{bmatrix} a_i \\ a_{-i} \end{bmatrix})$ be the payoff for agent i playing action a_i while the other agents play action a_{-i}.

A **stage game** is a single iteration of a matrix game, and a **repeated game** is the indefinite repetition of the stage game between the same agents. While matrix games do not have state, agents can encode the previous w joint actions taken by the agents as state information, as for example illustrated by Sandholm and Crites (1995).

Each individual matrix game has certain classic game theoretic values. The **minimax** value for player i is $m_i = \max_{\pi_i} \min_{a_{-i}} R_i(\pi_i, a_{-i})$, i.e. the least reward that can be achieved if the game is known and the game is only played once. A **Best-Response** (BR) to the opponents strategy π_{-i} is defined by $BR = \pi^* = \max_{\pi} R_i(\pi, \pi_{-i})$. This corresponds to the (expected) most reward that can be gained from playing, under the assumption that the game is known; the game is only played once; and the opponent strategy is known.

A **Nash Equilibrium** (NE) is a joint strategy such that no agent may unilaterally change its strategy without lowering its expected payoff in the one shot play of the game. Nash (1951) showed that every n-player matrix game has at least one such NE. A **Pareto optimal** solution of the game is a joint strategy such that no agent may unilaterally increase its expected payoff without making another agent worse off. A (joint) strategy π_1 is said to **Pareto dominate** a strategy π_2 if the expected payoff for π_1 is at least as high as for π_2 and higher for at least one of the agents. A joint strategy is **Pareto deficient** if it is not Pareto optimal.

In the studied matrix games, we assume that an agent can observe its own payoffs as well as the actions taken by all agents in each stage game, but only after the fact. All agents concurrently choose their actions. A possible adaptation of the agents' policy, i.e. learning as a result of observed opponent behavior, only takes effect in the next stage game. Repeated games are modeled as a series of stage games with the same opponent(s). Each agent then aims to maximize its reward from iterated play of the same matrix game.

R=0.35 R=0.35	T=0.5 S=0
S= 0 T=0.5	P=0.1 P=0.1

Table 1: Example payoffs for the (symmetrical) PD

We restrict our investigation to two-player, two-action games as these are well classified (Rapoport, Guyer, & Gordon, 1976). The arguments and formalism presented in the rest of the paper are however applicable to more general settings. The next section discusses the Prisoners Dilemma (PD), a special two-player two-action game.

2.2 The iterated Prisoner's Dilemma

We present the iterated **Prisoners Dilemma** (iPD) as a special matrix game. In this game, each player has a choice of two operations: either **cooperate** (C) with the other player or **defect** (D). The payoff matrix for joint actions is shown in Table 1. If both players cooperate, they both receive a given payoff R. However, if one player plays Cooperate, and the other plays Defect, the defector receives a payoff $T > R$, and the cooperator receives a much lower payoff $S < R$. If both players play Defect, they receive the low payoff P, that is however better than the payoff S ($P > S$). For each individual, the incentive is thus present to defect, hoping that the other player plays "cooperate. Myopic play by even one of the players already quickly leads both players to arrive at the suboptimal outcome of both players receiving a low reward P. Furthermore, the PD has as constraint that $R > \frac{(T+S)}{2}$; full cooperation is better than playing $\begin{bmatrix} C \\ D \end{bmatrix}$ and then $\begin{bmatrix} D \\ C \end{bmatrix}$ for the agents over two stage games.

The PD game is of particular interest as it has specific properties that make it unique among the matrix games. The PD is the only game of the taxonomy of two-player, two-action games listed by Rapoport et al. (1976) for which the natural outcome is stable (a NE), but Pareto-deficient: there are outcomes that Pareto-dominate the NE. This unique property also makes it non-trivial for competitive learning algorithms in repeated play; how can learners consider the possible future gains that are non-existent in the single shot game?

Most multi-agent learning algorithms to date have focused on an individual agent learning a (myopic) Best Response to the *current* strategies of the other agents. Play between such agents using this approach often converge, and have as goal to converge, to a one-shot NE. However, a famous result from game theory (the **folk theorem**) (Myerson, 1991) suggests that the goal of reaching a one-shot NE may be inappropriate in repeated games.

The folk theorem implies that, in many games, there exists NEs for repeated games, repeated Nash-Equilibria (rNEs), that yield higher individual payoffs to all agents than do one-shot NEs, i.e. the rNE Pareto dominates the NE. Hence, in repeated games, a successful set of agents should learn to play profitable rNEs. However, since many repeated games have an infinite number of rNEs, the folk theorem does little

to indicate which one the agents should play. Littman and Stone (2003) present an algorithm for computing rNEs that satisfies a set of desiderata, but how to learn these strategy online is unknown. Additionally, an agent may have preferences between rNEs and play one above the other, if allowed by its opponents.

It is however not given that the mainstream, or state of the art RL algorithms, will (approximately) learn these rNE equilibriums in self play or against various classes of opponents. As discussed in Section 1, and by Crandall and Goodrich (2005), the claim is that current MARL algorithms will not converge to good rNE for iPD-like games.

3. THE StrOPM FRAMEWORK

Here, we refine the StrOPM framework. We show (very successful) experimental results with an instantiation of the framework for matrix games in Section 4.

We can formulate the goal of an agent i as wanting to maximize the total reward it obtains from playing T times against an adversary:

$$R_i^{tot} = \sum_{t=0}^{t=T} val_i(\pi_{i,t}, \pi_{-i,t}). \quad (1)$$

where π_t is the policy at time t and val_i the expected reward for a agent i using policy $\pi_{i,t}$ and opponent policies $\pi_{-i,t}$. Thus, we want to specify or learn the time-varying policy $\pi_{i,t}$ that maximizes the reward obtained.

We make several observations: first, we have to start playing from some policy $\pi_{i,t=0}$, and we may not know the payoff of (joint) actions yet. We may also not know the policy of the adversary yet, nor to what extent it is adaptive.

We assume that we can define a state representation for an agent i that includes sufficient history to make the evolution of both its own policy and the opponent's policy Markovian: $\pi_{i,t+1} = f(s(t), \pi_{i,t})$, where $f(s(t), \pi_{i,t})$ is the state dependent function that adapts the agent's policy based on the agent's current policy $\pi_{i,t}$ and the current state $s(t)$. Likewise, the agent assumes that the opponent behaves similarly Markovian: $\pi_{-i,t+1} = g(s(t), \pi_{-i,t})$, where $g(s(t), \pi_{-i,t})$ is an unknown function that adapts the adversarie's policy. We assume that part of the state information in the $g(s(t))$ term depends on the actions taken by the agent i.

At its core, the StrOPM algorithm attempts to estimate this unknown adaptive function $g(s(t), \pi_{-i,t})$, and then computes forward all possible future action sequences resulting from the thus evolving policies $\pi_{i,t}$ and $\pi_{-i,t}$. It then updates the policy $\pi_{i,(t+1)}$ in the direction of those future action sequences that promise the most reward.

In the StrOPM framework, an agent applies reinforcement learning to a state-based policy as described in Algorithm 1. At each epoch of learning, the agent adapts its policy along the gradient of increasing reward. The gradient of reward is calculated including the expected changing behavior of the opponent, as a reaction of the agent's own actions. The StrOPM algorithm tracks the changes in observed opponent policy, on a state by state basis, over time. It is assumed that these changes in the opponent behavior, at least in part, reflect reaction to actions chosen by the StrOPM algorithm . The StrOPM algorithm not only tracks the changing behavior of the opponents, but also how these changes are potentially caused by the agent's own action. The policy of the StrOPM is then optimized with expected future reactions taken into account.

We describe the StrOPM algorithm from the perspectives of an agent i and its opponents, agents $-i$. We use this notation with subscripts to indicate policy, states, action, etc ... of the two types of agents. E.g., a_i and a_{-i} are actions of agents i and $-i$ respectively.

States. The set of states that an agent i can visit, S_i, fulfills the unichain assumption (Puterman, 1994): One set of "recurrent" class of states. Starting from any state in the class, the probability of visiting all the states in the class is 1. We introduce a **transition function** $T : S_i \times A_i \times A_{-i} \rightarrow S_i$ to return the next state of agent i upon playing a (joint) action from the current state.

Policies. For a state $s \in S_i$, $\pi_i(s)$ is the state-based policy of agent i, and $\pi_{-i}(s)$ is the estimate of the opponent $-i$ policies when agent i is in state s. The opponent policy is estimated online using Exponential Moving Average (EMA). The estimate of $\pi_{-i}(s)$ after observing action a_{-i} is adjusted according, for action a_i:

$$\pi_{-i,t+1}(s)(a_{-i}) = (1 - \alpha_{EMA1})\pi_{-i,t}(s)(a_{-i}) + \alpha_{EMA1}. \quad (2)$$

After this update, the policy $\pi_{-i,t+1}(s)$ is normalized to retain $\pi_{-i}(s)$ as a probability distribution by equally reducing the probabilities of the other actions to retain $\pi_{-i}(s)$ as a probability distribution.

Actions to update the policy. We introduce **policy update actions**. A policy update action $pua_i(s)$ dictates whether the likelihood of an action a_i should be increased for state s. Additionally, we introduce the *null* policy update action $pua_{null}(s)$ to indicate that the policy should not be changed.

Let $\pi_i(s)^{pua_i}$ be the policy achieved by applying the policy update action $pua_i(s)$ to $\pi(s)_i$. The policy $\pi(s)_i$ of agent i given $pua_i(s)$ not equal to the null action is updated according to:

$$\pi_{i,t+1}(s)^{pua_i} = (1 - \alpha_{LEARN})\pi_{i,t}(s)(a_i) + \alpha_{LEARN}, \quad (3)$$

where α_{LEARN} is the learning rate. The probabilities $\pi_i(s)(\cdot)$ for actions $a_j \neq a_i$ are then normalized to retain $\pi_i(s)$ as a probability distribution.

We wish to select policy update action $pua_i(s)^*$ in (3) that maximizes a measure of expected future payoff of the changed policy. Below, we explain how to compute this.

Estimation of impact actions. We introduce $\xi(\pi_{-i}(s), a_i)$ to estimate the impact of an agent i playing an action a_i on the development of the policies $\pi_{-i}(s)$ of the opponent. This function predicts the change in policy of the opponents upon playing action a_i; i.e.

$$\pi_{-i,t+1}(s) = \xi(\pi_{-i,t}(s), a_i). \quad (4)$$

As a first implementation of this function, we take the approach that the changes in the opponent policy are, at least in part, caused by an agent's own actions. Our StrOPM implementation estimates the change in policy as a continuation of the change in policy from estimation of the opponent policy N epochs in the past, i.e. how was the opponent policy at time $t - N$? This is done for each state: for state s reached after playing action joint action the change in policy for this new state is estimated as a linear extrapolation of

the change in the estimated opponent policy:

$$\xi(\pi_{-i,t}(s), a_i) = \frac{\pi_{-i,t}(s) - \pi_{-i,t-N}(s)}{N} + \pi_{-i,t}(s), \quad (5)$$

where we limit ξ to $[0, 1]$.

Thus estimating the change in opponent policies, the question becomes how to compute the value of proposed policy updates. Here, we make use of the unichain assumption that says we only need to consider loops starting from and returning to the current state to compute this value. The algorithm then updates the policy for the possible loops to increase the expected average reward.

Loops. From a state s_0, a single loop $L(s_0)$ is defined as:

$$L(s_0) = s_0, \begin{bmatrix} a_{i,0} \\ a_{-i,0} \end{bmatrix}, s_1, \begin{bmatrix} a_{i,1} \\ a_{-i,1} \end{bmatrix}, s_2, \ldots, s_{(n-1)}, \begin{bmatrix} a_{i,(n-1)} \\ a_{-i,(n-1)} \end{bmatrix}, s_0, \quad (6)$$

for a sequence of joint moves for agent i starting in s_0, going through $s1$, $s2$, ... to s_n and ending again in s_0. Each next state reached is through a joint move; $T(s_j, \begin{bmatrix} a_{i,j} \\ a_{-i,j} \end{bmatrix}) = s_{j+1}$. All intermediate states reached in the loop are unique, and not equal to s_0. For brevity, we denote the $j-th$ state s_j in the sequence by $L(s_0)^j$, the $j-th$ action pair $\begin{bmatrix} a_{i,j} \\ a_{-i,j} \end{bmatrix}$ by $L(s_0)_j$, and $L(s_0)_{j+}$ and $L(s_0)_{j-}$ the components of the $j-th$ action pair: $a_{i,j}$ and $a_{-i,j}$. The length of the sequence $L(s_0)$, $|L(s_0)|$, is equal to n; the number of joint actions played before the considered state is reached again.

Sets of loops. Let $Bag(s, n) = \{L(s) \,|\, |L(s)| \leq n\}$, i.e. $Bag(s, n)$ are all the loops starting in state s with length of at most n . The probability of a particular loop $L(s)$ occurring, denoted by $Pr(L(s))$ is:

$$Pr(L(s)) = \prod_{0 \leq j < |L|} Pr(L(s)_{j+} | \pi_{i,j}(L(s)^j)) \times Pr(L(s)_{j-} | \pi_{-i,j}(L(s)^j)), \quad (7)$$

where $L(s)_j$ and $L(s)^j$ are the respective elements in the loop as defined above, $\pi_{i,j}$ and $\pi_{-i,j}$ are the respective policies at time j, evolving according to Equation (4), and $Pr(\cdot)$ denotes the probability of a specific transition along the sequence given the respective policies $\pi(s)$.

The expected reward over the possible loops can then be expressed as the weighted expected value of individual loops:

$$E(Bag(s, n)) = \sum_{L(s) \in Bag(s,n)} Pr(L(s)) \times E(L(s)), \quad (8)$$

where $E(L(s))$ is the expected average reward for each joint action of a loop starting in state s:

$$E(L(s)) = \frac{\sum_{0 \leq j < |L(s)|} V(L(s)_j)}{|L(s)|}, \quad (9)$$

where $V(L(s)_j)$ denotes the estimated the value of the $j-th$ joint action in the loop, $L(s)_j$, to agent i.

Learning the value of joint actions. The values $V_i : A_i \times A_{-i} \rightarrow \Re$ track the estimates the value of a single joint action to agent i and through V_i, agent i learns its part of the rewards of the game, dependent upon the actions of the opponent. The function V_i is updated using EMA:

$$V_{i,t+1} = (1 - \alpha_{EMA2}) V_{i,t} + \alpha_{EMA2} \times r_t, \quad (10)$$

where r_t is the reward received by agent i in epoch t, and $0 < \alpha_{EMA2} \leq 1$ is the learning rate for the update.

Strategic Updating. Let the opponent changes in policy be estimated by $\xi(\pi_{-i}(s), a_i)$, we can then compute the policy update with highest expected payoff:

$$pua_i(s)^* = \max_i E(Bag(s, n))^{\xi}_{pua_i(s)}, \quad (11)$$

where $E(Bag(s, n))^{\xi}_{pua_i(s)}$ denotes the expected average reward for the possible loops of at most length n, starting in state s, after effecting policy update action $pua_i(s)$ and taking $\xi(\pi_{-i}(s), a_i)$ as the estimated opponent adaptation. Our StrOPM algorithm thus obtained is outlined in Algorithm 1.

Algorithm 1 StrOPM

```
Initialize π_i(s), π_{-i}(s) for all s ∈ S_i and V for all all joint actions. Set the initial state.
Do in each epoch sequentially for each agent i in state s:
loop
    calculate the highest valued pua_i(s)* using E(Bag(s,n))^ξ_{pua_i(s)} as value for the individual policy update actions.
    Update policy π_i(s) using the highest valued policy update pua_i(s)* action.
    Play action a_i ∈ A_i based on the chosen policy update action pua_i(s)*. StrOPM plays action a_i for chosen policy update action pua_i(s) = pua(s)* if pua* is not the null policy update action. Otherwise choose the action according to π_i(s).
    Receive reward R_i([a_i; a_{-i}]) for the joint action determined by agents −i.
    Update the estimate of the reward for the joint action V([a_i; a_{-i}]) and the opponent policy π_{-i}.
    set the current state to T(s, [a_i; a_{-i}]).
end loop
```

4. EXPERIMENTS

We first investigate the StrOPM algorithm for various well-known two-player, two-action games.

In Figures 1b,d,f, we show the results for StrOPM in a variety of games. Figure 1b shows the results for self-play for the game of "matching pennies", Figure 1d for a variant of a coordination game[2], and Figure 1f for the game of Chicken (see also Rapoport et al. (1976)). The corresponding payoff matrices are plotted above the respective graphs in Figures 1a,c,e. Experiments are averaged over 50 runs. A learning

[2]In the simple coordination game, the two agents must learn to either play $\begin{bmatrix} C \\ D \end{bmatrix}$ or $\begin{bmatrix} D \\ C \end{bmatrix}$ to achieve the highest reward.

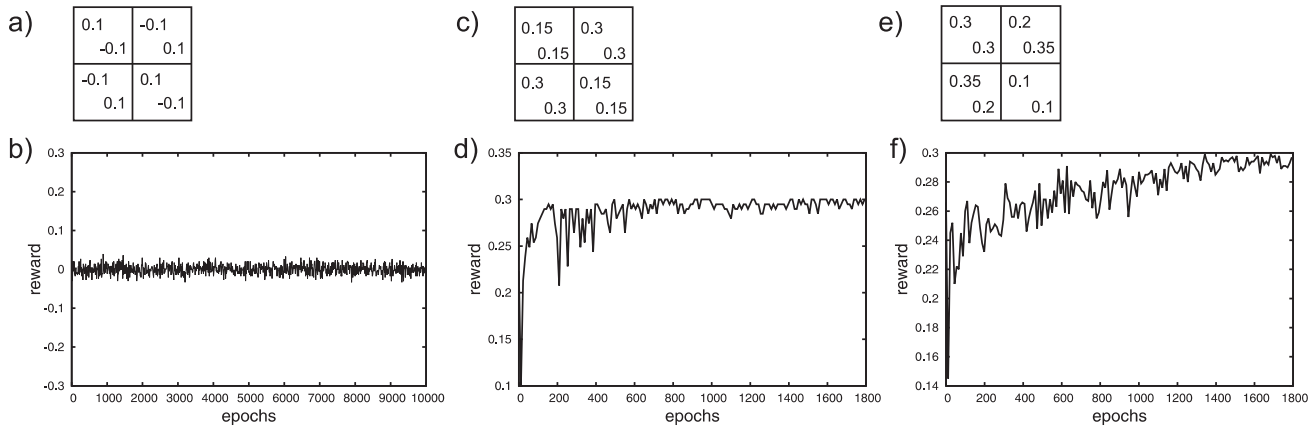


Figure 1: a) payoff matrix for "Matching Pennies", reward representation as in Table 1. b)two StrOPM players playing Matching Pennies. c,d) payoff matrix and two StrOPM players playing a Coordination Game. e,f) payoff matrix and two StrOPM players playing a game of Chicken.

rate of 0.01 was used for all the EMA equations of Section 3. The StrOPM algorithm looks back $N = 10$ epochs in Equation 5. Additionally, we added an $\epsilon = 0.01$ probability of the StrOPM agent taking an exploration move in each epoch to ensure all states are sufficiently sampled in play. Note that we reuse the payoff labels C and D for ease of exposition.

In the games shown in Figure 1, the StrOPM algorithm finds the optimal Nash-Equilibria rapidly, as was to be expected since these games are unproblematic in the sense that the Nash-Equilibria of the games is also a good solution for iterated play of the game. The results show that the StrOPM algorithm finds good solutions for these diverse games quickly, as it should since it is not designed with a bias for any particular matrix game.

StrOPM in the iPD. Of greater interest than the previous games is the iterated Prisoner's Dilemma, as it is a much harder, open problem for MARLs and is also the motivating example for the work of Crandall and Goodrich (2005).

In Figure 2a, we present results of the StrOPM algorithm for self-play in the iPD game of Figure 2a (using the payoff matrix of Table 1). To highlight the difference between a forward-thinking algorithm like StrOPM, and more reactive MOPM algorithms, we compare in Figure 1 the results for the StrOPM algorithm in self-play with that of self-play for an MOPM player in the form of a reduced variant of StrOPM that does not anticipate reactions to its actions. With the latter, we mean that the used ξ functions do not predict a change in policy of the opponent, i.e $\xi(\pi_{-i}, a_i) = \pi_{-i}$. As such, the StrOPM implementation reverts to an opponent modeler type player that simply moves its policy to a best-response to the recently observed play of the opponent.

The StrOPM learner in self-play (Figure 2a, solid line) converges to playing the (optimal) Cooperate-Cooperate ($\{C, C\}$) strategy with reward 0.35. The full cooperation equilibrium is reached as the StrOPM learner estimates that leaving the full cooperation state leads to states where more and more defection is expected. Additionally, the full cooperation state is expected to lead to more cooperation. In contrast, the MOPM variant quickly learns cooperation is risky and moves to full defection(Figure 2a, dashed line).

Furthermore, a closer study of the state-based policy of the StrOPM players reveals that the agents in self-play repeatedly exhibited a learned Tit-For-Tat strategy (Axelrod, 1984). The Tit-For-Tat algorithm plays C until the opponent defects upon which a D is played followed by again C until the next defection to encourage cooperation. For a four state agent, with each state encoding the last joint action of play, the policy is to play C in state $\begin{bmatrix} C \\ C \end{bmatrix}$, play D from state $\begin{bmatrix} C \\ D \end{bmatrix}$, and C from $\begin{bmatrix} D \\ D \end{bmatrix}$. The StrOPM agents learn this strategy and play C to cooperate and reach a good equilibrium, except for occasional exploratory moves. At the same time the StrOPM players also evolve a "threat" state where defection is retaliated in order to guard against exploitation by the opponent. The need, and the possibility, for learning a threat state as in the Tit-For-Tat play of the iPD is an interesting venue of research for iterated play of games. For example, a StrOPM player playing against a hard-coded Tit-For-Tat player quickly learned to cooperate (not shown), but did not evolve such a threat state.

As a negative result, a StrOPM agent playing against a MOPM agent converged to mutual defection (not shown) as the MOPM player learned to play defect. The MOPM agent playing agains a hard coded Tit-For-Tat player was however able to achieve full cooperation as the MOPM player was consistently taught by the Tit-For-Tat agent that defection was not worthwhile. This strategy was however not yet available to the StrOPM agent as it still had to be learned. This indicates that players in the iPD must both be reasonably savy, be that though their learning algorithm or a-priori imparted strategies, for mutual cooperation to emerge.

Ongoing preliminary results (not shown here) however indicate that generalised StrOPM-type agents are able to achieve high levels of cooperation in the generalised n-player iPD, the nIPD (Yao & Darwen, 1994). For example, for three agents, full cooperation is reached in the 3IPD game. In ('t Hoen & La Poutré, 2005), we have shown that agents participating in a sequence of auctions for individual items with as goal to win a bundle of these items can face a nIPD-like problem. Each of the agents has an individual incentive

to bid strategically. The prices paid for the items are however much higher if all the agents follow this strategy. A venue of research is to investigate whether the concepts of the StrOPM algorithm can be applied to this more complex setting to arrive at more beneficial outcomes from the perspective of the buyers.

StrOPM versus M-Qubed:. In Figure 2b we show results for an agent using the StrOPM algorithm (using the same settings as above) playing against an agent using the M-Qubed algorithm by Crandall and Goodrich (2005). We choose M-Qubed an a state-of-the-art MARL algorithm to compare StrOPM to, as Crandall & Goodrich claim it performs on par with other state-of-the-art MARLs on simple matrix games, and is additonally capable of successfully solving the iPD game.

M-Qubed as introduced in Crandall and Goodrich (2005) operates as a mix of a Q-learning type algorithm and a pure strategy player. The algorithm learns Q-like values that encode the received reward and the discounted future reward. The algorithm investigates the policy of playing according to the Q-like values various exploration strategies. At the same time, the algorithm considers whether to play any of its actions consistently to promote possibly good interactions with the opponent. A parameter $\beta \in [0, 1]$ is used in M-Qubed and adapted in play to learn the best overall strategy; M-Qubed plays a game using the Q-values with probability $(1 - \beta)$ or it plays the highest valued action as a pure strategy with probability β. We used for M-Qubed the same settings as Crandall & Goodrich note in their work[3].

When a StrOPM player plays against a player using M-Qubed, the full average reward of 0.35 is not reached as alternating Cooperate/Defect followed by Defect/Cooperate patterns often emerged, giving an average reward of 0.25 per epoch. This is not the optimal C, C outcome, but still much better than the typical D, D outcome other MARLs arrive at. This result suggests that the switching policy in M-Qubed is repeately "tripped". The non-linear switching policy threshold is hard to approximate in StrOPM's linear opponent change model. Including more history of non-linear update terms in future extensions may solve this issue.

The trouble with precoded strategies. The M-Qubed algorithm has a clear potential weakness since it has to decide between the pure (or at least precoded) strategy of playing full cooperation, and a myopic Q-Learning type strategy. We would expect it to perform badly when the optimal strategy involves a more complex strategy that is not precoded. Our StrOPM algorithm on the other hand is not biased to playing precoded strategies.

We tested this hypothesis in the game with the payoff matrix of Figure 3a, dubbed "Switching Bait", which we specifically designed so that the highest (average) joint reward is achieved when agents alternate Cooperate/Defect and Defect/Cooperate (or at least play these two joint actions equally often). Playing the Cooperate/Defect, Defect/Cooperate mixed strategy yields an average reward of 0.45 per epoch for both agents.

In experiments playing the Switching Bait game, M-Qubed

[3]These settings allowed us to successfully replicate the results in Crandall and Goodrich (2005).

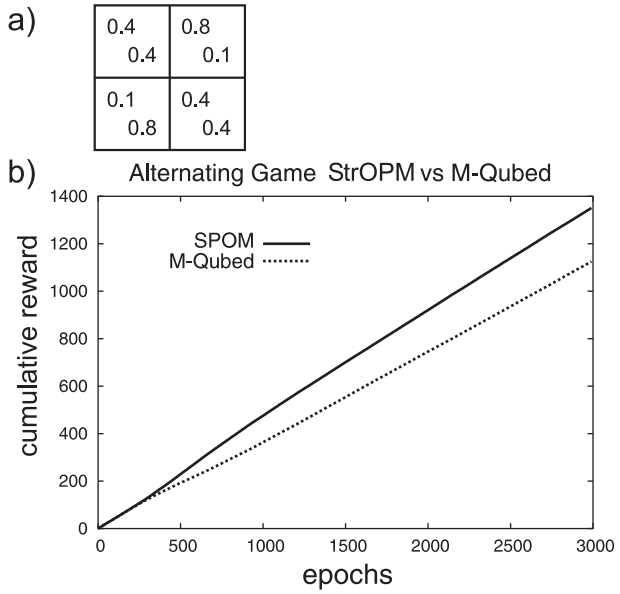


Figure 3: a) payoff matrix for "Switching Bait" game b) respective cumulative reward obtained by a StrOPM player playing against an M-Qubed player.

was frequently not able to find the optimal mixed strategy in selfplay, as it predominantly converged to playing C or D and achieves a reward of only 0.4 per epoch. Higher rewards of $\frac{0.8+0.1}{2}$ where achived by one of the M-Qubed learners by playing C and D alternatingly, while the other M-Qubed player plays C or D continouously for an average reward of $\frac{0.1+0.4}{2}$. For the Switching Bait new game, M-Qubed often decides to play the safer strategy of full cooperation or defection.

The StrOPM algorithm, unlike M-Qubed, is not biased to playing pure strategies. For the Switching Bait game, it often finds the alternating Cooperate/Defect pattern and matches it to the opponent's Defect/Cooperate pattern in self play. Importantly, a StrOPM player playing againts an M-Qubed player was found to either achieve this mutually beneficial equilibrium or was repeatedly able to exploit M-Qubed if the opponent decided to play a (precoded) pure strategy. An example of this latter behavior is shown in Figure 3b, where the cumulative payoff achieved during play is plotted. The StrOPM algorithm achieves a higher payoff as M-Quebed opts for the safer, pure strategy. This clearly demonstrates the weakness of using precoded fixed strategies (as in M-Qubed).

5. DISCUSSION AND CONCLUSIONS

We have presented a number of important contributions in this paper. First, we argued the need for Multi-Agent learning algorithms to take into account the impact of their actions on the adaptive policy of the opponents. We then presented a novel framework that allows agents to apply the above observation to achieve desired outcomes for representative iterated matrix games, and, importantly, also for the iterated Prisoner's Dilemma. The latter is notoriously difficult to solve for current state-or-the-art MARL approaches. Our MARL framework allows an agent to learn profitable strategies for long term behavior. Finally, we demonstrated

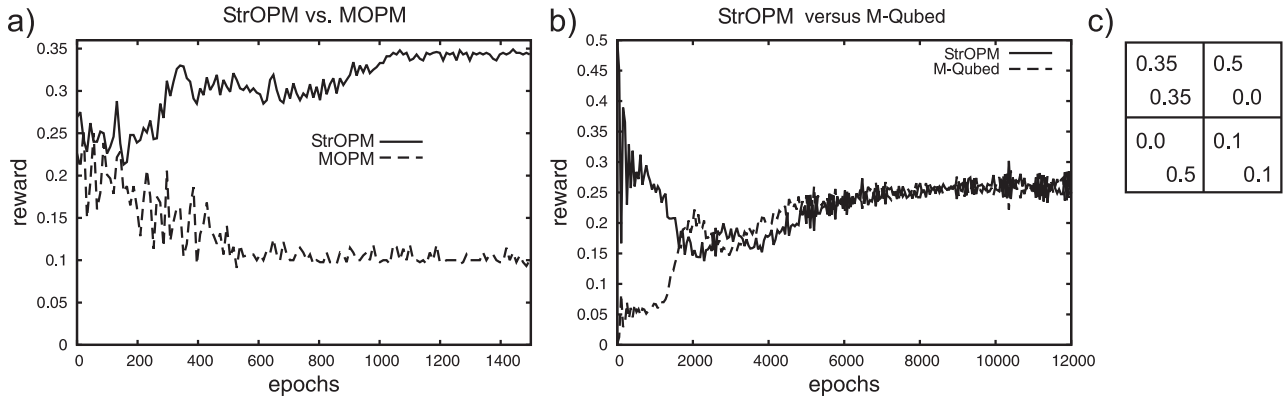


Figure 2: a) StrOPM players playing the iPD (solid line) and the MOPM-variant (dashed line). b) A StrOPM player playing vs M-Qubed for the iPD. c) iPD payoff matrix

that incorporating fixed precoded strategies in MARL algorithms makes agents vulnerable to exploitation, especially when mixed strategies can be optimal (of which there is a continuous spectrum). Additionally, fixed precoded strategies may not be suited to novel settings.

The important novel component of this work is the concept of learning the changes in opponents policy due ones own action. This is an idea that can in fact be incorporated into the quickly growing literature on MARL. One can foresee more and more complex nested opponent models (Hu & Wellman, 1998) to extract every bit of reward from complex games like iPD. Although perfectly learning about an opponent while at the same time perfectly learning to adjust oneself is problematic (Nachbar & Zame, 1996), there is still much scope to be "smarter" than your opponents.

Our future challenge is to apply our new concept to large, complex state spaces. One interesting direction may be to integrate our framework with new ideas about state space representations like Predictive State Representations (Wolfe, James, & Singh, 2005).

References

Axelrod, R. (1984). *The evolution of cooperation.* Basic Books, New York, NY.

Banerjee, B., & Peng, J. (2004). The role of reactivity in multiagent learning. In *Proc. 3rd AAMAS* (p. 538-545).

Bowling, M. (2004). Convergence and no-regret in multiagent learning. In *Proc. NIPS-17* (pp. 209–216).

Bowling, M. H., & Veloso, M. M. (2002). Multiagent learning using a variable learning rate. *Artificial Intelligence*, *136*(2), 215-250.

Conitzer, V., & Sandholm, T. (2003). AWESOME: A General Multiagent Learning Algorithm that Converges in Self-Play and Learns a Best Response Against Stationary Opponents. In *Proc. 20th ICML* (p. 83-90).

Crandall, J. W., & Goodrich, M. A. (2005). Learning to compete, compromise, and cooperate in repeated general-sum games. In *Proc. 22nd ICML.*

Greenwald, A. R., & Hall, K. (2003). Correlated Q-learning. In *Proc. 20th ICML* (p. 242-249).

Hu, J., & Wellman, M. (1998). Online learning about other agents in a dynamic multiagent system. In *Proc ACM Conf. on Autonomous Agents* (p. 239-246).

Littman, M. L., & Stone, P. (2003). A polynomial-time nash equilibrium algorithm for repeated games. In *Proc. 4th ACM Conf. on Electronic Commerce* (p. 48-54).

Myerson, R. B. (1991). *Game theory. analysis of conflict.* Harvard University Press.

Nachbar, J. H., & Zame, W. R. (1996). Non-computable strategies and discounted repeated games. *Economic Theory*, *8*, 103– 122.

Nash, J. (1951). Non-cooperative games. *Annals of Mathematics*, *54*, 286–295.

Puterman, M. L. (1994). *Markov decision process.* John Wiley and Sons, Inc., New York.

Rapoport, A., Guyer, M., & Gordon, D. (1976). *The 2x2 game.* MI: University of Michigan Press.

Sandholm, T., & Crites, R. (1995). Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, *37*, 147-166.

Shoham, Y., Powers, R., & Grenager, T. (2004). Multi-agent reinforcement learning: a critical survey. In *AAAI Fall Symposium on Artificial Multi-Agent Learning.*

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction.* Cambridge, MA: MIT Press.

't Hoen, P., & La Poutré, J. (2005). Repeated auctions with complementarities. In *AMEC VII.* (to appear, LNCS)

Tesauro, G. (2003). Extending Q-learning to general adaptive multi-agent systems. In *Nips-16* (pp. 871–878).

Weinberg, M., & Rosenschein, J. S. (2004). Best-response multiagent learning in non-stationary environments. In *Proc. 3rd AAMAS* (p. 506-513). New York.

Wolfe, B., James, M. R., & Singh, S. (2005). Learning predictive state representations in dynamical systems without reset. In *Proc. 22nd ICML.*

Yao, X., & Darwen, P. J. (1994). An experimental study of n-person iterated prisoner's dilemma games. *Informatica*, *18*, 435–450.

Learning the Required Number of Agents for Complex Tasks

Sébastien Paquet
DAMAS laboratory, department of computer science and software engineering
Laval University, Canada
spaquet@damas.ift.ulaval.ca

Brahim Chaib-draa
DAMAS laboratory, department of computer science and software engineering
Laval University, Canada
chaib@damas.ift.ulaval.ca

ABSTRACT

Coordinating agents in a complex environment is a hard problem, but it can become even harder when certain characteristics of the tasks, like the required number of agents, are unknown. In those settings, agents not only have to coordinate themselves on the different tasks, but they also have to learn how many agents are required for each task. To achieve that, we have elaborated a selective perception reinforcement learning algorithm to enable agents to learn the required number of agents. Even though there were continuous variables in the task description, the agents were able to learn their expected reward according to the task description and the number of agents. The results, obtained in the RoboCupRescue, show an improvement in the agents overall performance.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Coherence and Coordination, Intelligent Agents, Multiagent Systems*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Performance, Experimentation.

Keywords

Multiagent, Reinforcement Learning, Coordination.

1. INTRODUCTION

In a cooperative multiagent environment, agents have to divide up the different tasks among themselves in order to accomplish them efficiently. However, if agents are faced with complex tasks, it might be hard for them to determine how many agents are required to accomplish each task, which is important information for the coordination process. Without this information, agents would not be able to divide up the different tasks efficiently.

To learn the required number of agents for each task, we have developed a selective perception reinforcement learning algorithm, which is useful to manage a large set of possible task descriptions with discrete or continuous variables. It enables us to regroup common task descriptions together, thus greatly diminishing the number of different task descriptions. Starting from this, the reinforcement learning algorithm can work with a relatively small state space.

Our tests in the RoboCupRescue simulation environment show that the agents are able to learn a compact representation of the state space, facilitating their task of learning good expected rewards. Furthermore, they were also able to use those expected rewards to choose the right number of agents to assign to each task. This information helped the agents to efficiently coordinate themselves on the different tasks, thus improving the group performance.

In the remaining sections of this paper, we introduce the RoboCupRescue simulation which motivated the development of our learning algorithm. Next, we describe our selective perception reinforcement learning algorithm. Then, we present the results obtained by testing our algorithm in the RoboCupRescue simulation.

2. DOMAIN

The resource allocation problem that motivated this work requires an efficient allocation of *FireBrigade* agents to the tasks of extinguishing fires. Therefore, throughout this paper, we consider *FireBrigade* agents to be resources that are allocated to fires. Those *FireBrigade* agents evolve in the RoboCupRescue simulation [4] which takes the form of an annual competition in which participants are designing rescue agents (fire fighters, policemen and paramedics) trying to minimize damages, caused by a big earthquake, such as civilians buried, buildings on fire and blocked roads.

For this article, only the agents responsible for extinguishing fires are considered, i.e. the *FireBrigade* agents which extinguish fires and the *FireStation* agent which is their communication center. All those agents are in contact by radio, but they are limited on the number of messages they can send and on the length of those messages. Furthermore, in the simulation, each individual agent receives visual information of only the region surrounding it. Therefore, agents rely on the communication to acquire some knowledge about the environment. Since the center agent has more communication capabilities, it normally has a better global view of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

the situation than the *FireBrigade* agents.

As mentioned before, the task of the *FireBrigade* agents is to extinguish fires. Therefore, at each step in time, each *FireBrigade* agent has to choose which burning building to extinguish. However, in order to be effective, *FireBrigade* agents have to be coordinated on the same burning buildings, because more than one agent is often required to extinguish a building on fire. For this article, we do not consider the importance of a task. We suppose that the agents have a utility function allowing them to place the different fires in order of priority. Their assignment to different burning buildings depends on the number of available agents and on their distance from those fires.

The main decision for the agents is to choose how many agents to send to the most important fires. However, the problem is that they do not know how many agents are required for each building on fire. The required number of agents depends on the characteristics of the task. For example, a small building on fire may require only two agents to extinguish it, but a bigger one may require 10 agents. In order to be effective, agents have to learn the right number of agents to assign to each task which is described as:

- the intensity of the fire (3 possible values),
- the building's composition (3 possible values),
- the building's size (continuous value),
- the building's damage (4 possible values).

Therefore, there are many possible task descriptions. In fact, with the continuous attribute "building's size", there is an infinite number of task descriptions. In the results section, we show an important reduction of this huge number of possible situations. But first, in the next sections, we present our selective perception reinforcement learning algorithm.

3. PROBLEM DEFINITION

In this article, we are considering agents accomplishing tasks in an uncertain and dynamic environment where most tasks require more than one agent to be accomplished. In this case, agents are forced to be coordinated and they need to know the required number of agents to accomplish each task. This can be hard to estimate if subtle changes in a task description can change the required number of agents.

To make this estimation, we propose a new approach that uses a selective perception reinforcement learning algorithm to learn the expected reward if a certain number of agents tries to accomplish a certain type of task. An advantage of learning expected rewards instead of directly learning the number of agents is that the rewards can encapsulate the time needed to accomplish a task. A task taking more time to be accomplished has a smaller expected reward, due to the discount factor.

The method we propose is inspired by the *Utree* [5] and the *Continuous Utree* [10] algorithms. Like those methods, the agent is dynamically learning a tree representation of the state space in order to reduce the number of states considered. In our case however, we do not consider states, but task descriptions and our objective is not to find a policy for the agent, but to evaluate the capability of a given group of agents to accomplish a task. Therefore, the only implicit action is to accomplish a task, but it is never explicitly considered in the model. Our model can be described as a tuple $< D, N, R, T >$ where:

- D is the set of all possible task descriptions.
- N is the number of available agents.
- R is a reward function that gives the reward if a task is accomplished.
- T is a transition function that gives the probability to go from one task description to another. In other words, it gives the probability that the task description changes while some agents are accomplishing it.

The transition function is useful to take the dynamic aspects of the environment into consideration. It takes time to accomplish a task and, during this time, some characteristics of the current task may change and this may have an impact on the required number of agents.

Moreover, a task description is described in a factored way by a set of discrete or continuous attributes: $\{A_1, A_2, \ldots, A_n\}$. The number of different task descriptions can be huge, especially if there are continuous attributes. The primary objective of building a tree representation of the task description space is to reduce the number of task descriptions considered. The next section explains how the tree is built and how it is used to estimate the required number of agents to accomplish each task.

4. TREE CONSTRUCTION

Our algorithm uses a tree structure similar to a decision tree in which each leaf of the tree represents an abstract task description that regroups many task descriptions. This compact representation is iteratively expanded when new experiences are gathered by the learning agents.

At the beginning, all tasks are considered to be the same, so there is only the root of the tree. After each simulation, agents add new experiences to the tree and the tree is expanded. Those experiences are tasks that the agents tried to accomplished in the simulation with the rewards they obtained. All experiences are stored in the leaves of the tree. To expand the tree, the algorithm tests for each leaf l whether it would be interesting to divide the experiences stored in l by adding a new test on a task's attribute. The addition of a new test refines the agents' view of the task description space.

An advantage of this algorithm is that it distinguishes only tasks that really need to be distinguished. Therefore, the task description space is reduced, thus facilitating the reinforcement learning process.

4.1 Tree Structure

The algorithm presented here is an instance-based algorithm in which a tree is used to store all agents' experiences which are kept in the leaves of the tree. The other nodes of the tree, called center nodes, are used to divide the instances with a test on a specific attribute. Each leaf of the tree also contains a Q-value indicating the expected reward if a task that belongs to this leaf is chosen. In our approach, a leaf l of the tree is considered to be a task description (a state) for the learning algorithm.

An example of a tree is shown in Figure 1. Each rectangular node represents a test on the specified attribute.

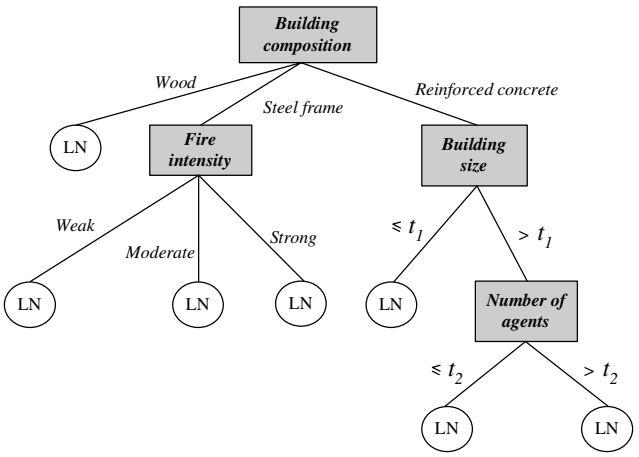


Figure 1: Structure of a tree.

The words on the links represent possible values for discrete variables. The tree also contains a center node testing on a continuous attribute, the "Building size". A test on a continuous attribute always has two possible results, it is either less or equal to the threshold or greater than the threshold. The oval nodes (LN) are the leaf nodes of the tree where the agents' experiences and the Q-values are stored.

4.2 Recording of the Agents' Experiences

In the RoboCupRescue, each simulation has 300 time steps. During a simulation, each *FireBrigade* agent records, at each time step t, its experience about which fire it is trying to extinguish. More precisely, an experience is recorded as an "instance" that contains the task it tried ($d_t \in D$), the number of agents that tried the same task (n_t) and the reward it obtained (r_t). Each instance also has a link to the preceding instance and the next one, thus making a chain of instances. Consequently, an instance at time t is defined as:

$$i_t = \langle i_{t-1}, d_t, n_t, r_t, i_{t+1} \rangle \quad (1)$$

In our case, we have one chain for each fire f that an agent chooses to extinguish. A chain contains all instances from the time an agent chooses to extinguish the fire f until it changes to another fire. Therefore, during a simulation, each *FireBrigade* agent records many instances organized in many instance chains. They keep all those instances until the end of the simulation.

The learning process takes place after a simulation, when the agents have time to learn. At this time, the *FireBrigade* agents regroup all their instances together, then the tree is updated with all those new instances and the resulting tree is returned to each agent. By regrouping their instances, agents can accelerate the learning process.

4.3 Update of the Tree

This section presents the algorithm used to update the tree using all the new recorded instances. Algorithm 1 shows an abstract version of the algorithm and the following subsections present each function used in more detail.

4.3.1 Add Instances

The first step is simply to add all the new instances, recorded by the *FireBrigade* agents, to the leaves they belong to. To find those leaves, the algorithm starts at the root of the tree and heads down the tree choosing at each center node the branch indicated by the result of the test on the instance's attribute, which could be one of the attributes of the task description d or the number of agents n.

Procedure UPDATE-TREE(*Instances*)

Inputs: *Instances*: all instances to add to the tree.
Static: *Tree*: the tree.

for all i in *Instances* **do**
  ADD-INSTANCE(*Tree*, i)
end for
UPDATE-Q-VALUES(*Tree*)
EXPAND(*Tree*)
UPDATE-Q-VALUES(*Tree*)

Algorithm 1: Algorithm used to update the tree.

4.3.2 Update Q-values

The second step updates the Q-values of each leaf node to take into consideration the new instances which were just added. The updates are done with the following equation:

$$Q'(l) \leftarrow \hat{R}(l) + \gamma \sum_{l'} \hat{T}(l, l') Q(l') \quad (2)$$

where $Q(l)$ is the expected reward if the agent tries to accomplish a task belonging to the leaf l, γ is the discount factor, $\hat{R}(l)$ is the estimated immediate reward if a task that belongs to the leaf l is chosen, $\hat{T}(l, l')$ is the estimated probability that the next instance would be stored in leaf l' given that the current instance is stored in leaf l. Those values are calculated directly from the recorded instances:

$$\hat{R}(l) = \frac{\sum_{i_t \in I_l} r_t}{|I_l|} \quad (3)$$

$$\hat{T}(l, l') = \frac{|\{i_t \mid i_t \in I_l \wedge L(i_{t+1}) = l'\}|}{|I_l|} \quad (4)$$

where $L(i)$ is a function returning the leaf l of an instance i, I_l represents the set of all instances stored in leaf l, $|I_l|$ is the number of instances in leaf l and r_t is the reward obtained at time t when n_t agents were trying to accomplish the task d_t.

To update the Q-values, the equation 2 is applied iteratively until the average squared error is less than a small specified threshold. The error is calculated using the following equation, which is the average squared difference between the new and the old Q-values:

$$E = \frac{\sum_{l} (Q'(l) - Q(l))^2}{nl} \quad (5)$$

where nl is the number of leaf nodes in the tree.

4.3.3 Expand the Tree

After the Q-values have been updated, the third step checks all leaf nodes to see if it would be useful to expand a leaf and replace it with a new center node, thus dividing the instances more finely and refining the agent's representation of the task description space, in order to help the agent to predict rewards.

Before trying to split a leaf node, the algorithm first checks if there are enough instances stored in the leaf. If the number of instances is too small, the algorithm does not try to split the leaf node. This is done to make sure that the statistical measures used to split the nodes are meaningful and that the algorithm is not overfitting the data.

If there are enough instances, the algorithm tries different tests to divide the instances more finely. To find the best test, the agent tries all possible tests, i.e. it tries to divide the instances according to each attribute describing a task or the number of agents. After all attributes have been tested, it chooses the attribute that maximizes the error reduction as shown in equation 6 [8].

The error measure considered is the standard deviation on the instances' expected rewards ($sd(I_l)$). Therefore, a test is chosen if, by splitting the instances, it ends up reducing the standard deviation on the expected rewards. If the standard deviation is reduced, it means that the rewards are closer to one another. Thus, the tree moves toward its objective of dividing the instances in groups with similar expected rewards, in order to help the agent to predict rewards. In fact, the test is chosen only if the expected error reduction is greater than a certain threshold, if not, it means that the test does not add enough distinction, so the leaf is not expanded.

The expected error reduction obtained when dividing the instances I_l of leaf l is calculated using the following equation where I_k denotes the subset of instances in I_l that have the kth outcome for the potential test:

$$\Delta error = sd(I_l) - \sum_k \frac{|I_k|}{|I_l|} \times sd(I_k) \qquad (6)$$

The standard deviation is calculated on the expected reward of each instance which is defined as:

$$Q_I(i_t) = r_t + \gamma \hat{T}(L(i_t), L(i_{t+1})) \times Q(L(i_{t+1})) \qquad (7)$$

where $\hat{T}(L(i_t), L(i_{t+1}))$ is calculated using equation 4 and $Q(L(i_{t+1}))$ using equation 2.

As mentioned earlier, one test is tried for each possible instance's attribute. For a discrete attribute, we divide the instances according to their value for this attribute. For instance, if an attribute has three possible values, it generates three subsets, thus adding three children nodes to the tree. We then use Equation 6 and record the error reduction for this test. For a continuous attribute, we have to test different thresholds to find the best one. A continuous attribute always divides the instances in two subsets, the first one is for the instances with a value less or equal to the threshold for the specified attribute and the second subset is for the instances with a value greater than the threshold.

To find the best threshold, we have used the technique described by Quinlan [7]. The instances are first sorted according to their value for the attribute being considered. Afterwards, we examine all $m-1$ possible splits, where m is the number of different values. For example, with an ordered list of values $\{v_1, v_2, ..., v_m\}$, we try all possible thresholds. So, we try the value v_1 as a threshold, thus dividing the instances in two subsets, those less or equal and those greater than v_1. We calculate and record the error reduction for this division. Then, we do the same thing for the other possible values, v_2 to v_{m-1}. At the end, we keep only the threshold with the best error reduction value.

Function NUMBER-AGENTS-REQUIRED(d)

Inputs: d: a task description.
Static: *Tree*: the tree learned.
N: the number of available agents.
Threshold: the limit to surpass.

for $n = 1$ to N **do**
  $expReward \leftarrow$ EXPECTED-REWARD($Tree$, d, n)
  if $expReward \geq Threshold$ **then**
    return n
  end if
end for
return ∞

Algorithm 2: Algorithm used to find the required number of agents for a given task description d.

When the tree has been updated, the UPDATE-Q-VALUES function is called again to take the new tree structure into consideration.

4.4 Use of the Tree

During a simulation, all learning agents use the same learned tree to estimate the number of agents that are required to accomplish a task. Since the number of agents is considered as an attribute when the tree is learned, if different numbers of agents are tested, different leaves and thus different rewards may be found, even with the same task description. Consequently, to find the required number of agents for a particular task, the algorithm can test different number of agents and look at the expected rewards returned by the tree.

Algorithm 2 presents the function used to estimate the required number of agents for a given task. In this algorithm, the function EXPECTED-REWARD returns the expected reward if n agents are trying to accomplish a task described as d. To do so, it finds the corresponding leaf in the tree, considering the task description d and the number of agents n, and records the expected reward for this abstract task.

The EXPECTED-REWARD function is called for all possible number of agents until the expected reward returned by the tree is greater than a specified *Threshold*. If the expected reward is greater then the *Threshold*, it means that the current number of agents should be enough to accomplish the task. If the expected reward is always under the *Threshold*, even with the maximum number of agents, the function returns ∞, meaning that the task is considered impossible with the available number of agents N. The *Threshold* value is set empirically and it corresponds to the minimum expected reward needed to accomplish a task. The agent stops when the *Threshold* is exceeded because the objective here is to find the minimum number of agents for each task.

4.5 Algorithm Characteristics

First of all, let's look at the algorithm complexity. The first step of the algorithm used to update the tree is to add all the new instances recorded during the last simulation. In the worst case, this step takes $O(|I|D_{\max})$, where $|I|$ is the number of instances to add and $D_{\max}$ is the maximal depth of the tree. Therefore, as the tree grows, this step takes more time. However, in our RoboCupRescue experiments, the maximum depth of the tree was always relatively small (≤ 30).

The second step is to update the Q-values using Equa-

tion 2. The reward and the transition function (Equations 3 and 4) do not take time, because they are simply updated each time a new instance is added to a leaf. The complexity of Equation 2 depends on the number of subsequent leaf nodes link to the current leaf node. To update the Q-values, the algorithm has to visit all the leaves and in the worst case, all the leaves are connected together, thus the complexity is $O(|L|^2)$, where $|L|$ is the number of leaves in the tree. This complexity is multiplied by the number of iterations needed until convergence of the Q-values. Since the Q-values are normally only slightly modified when the new instances are added, it generally does not take a lot of iterations to converge. Most of the time, it took less then 10 iterations in our tests.

The third step of the algorithm is to expand the tree. To achieve that, the algorithm has to visit all the leaves of the tree one time. For each leaf, it has to evaluate all possible splits using all the attributes describing a task. For a discrete attribute, there is only one split to try. For a continuous attribute, there are as many possible splits as there are different values for this continuous attribute in the current leaf. In the worst case, there are as many attribute values as there are instances in the leaf. Therefore, the complexity in the worst case is: $O(|L|n_d n_c|I|_{\max})$, where $|L|$ is the number of leaves in the tree, n_d and n_c are the number of discrete and continuous variables used to describe a task and $|I|_{\max}$ is the maximum number of instances in a leaf.

The last step of the algorithm does another update of the Q-values. Consequently, the total complexity of the algorithm is: $O(|I|D_{\max}) + 2O(|L|^2) + O(|L|n_d n_c|I|_{\max})$. The most expensive step is the expansion step, because it manipulates all the instances stored in all the leaves of the tree. In our tests, the time to update the tree was not a big factor since it was executed offline. It always took less time to learn the tree than to run a simulation.

Moreover, for this algorithm, we suppose that all the *FireBrigade* agents have the same vision of the situation, i.e. they see the same fires. In the RoboCupRescue it is almost always the case, because the agents are close from one another when they are extinguishing fires and the fires can be seen from a far distance.

Another hypothesis is that all the agents have the same learned tree. The tree is learned offline after each simulation, using all the instances gather by the *FireBrigade* agents during the simulation. The communications between the agents are really limited during a simulation, thus we wanted the agents to have some common ground on which to base their decisions in order to reduce the amount of communication necessary to maintain the coordination between the agents.

In the next section, we present some experiments in which we described in more detail how the learned trees are used. We also present results showing the quality of the solutions found and the speed at which the tree grows when we add new instances.

5. EXPERIMENTS

In this section, we present how our learning algorithm can be used in the RoboCupRescue simulation to help the *FireBrigade* agents to coordinate themselves on the buildings on fire to extinguish. Since there could be a lot of fires, agents do not consider all fires at once. They choose separately which fire area to extinguish and which specific building in the chosen fire area to extinguish. Fire areas are simply

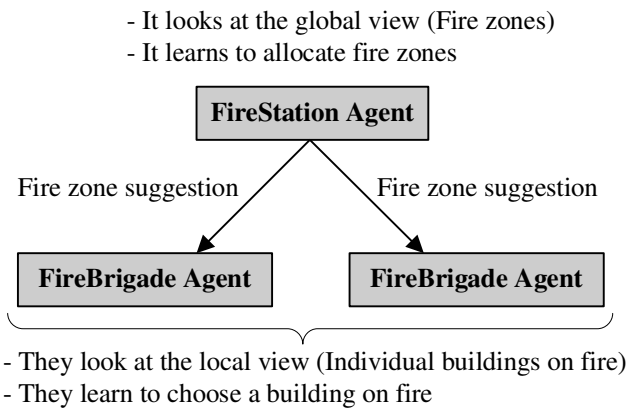


Figure 2: Illustrates how the agents collaborate to choose which fires to extinguish.

groups of close buildings on fire. To make their decision, agents use the tree created offline to estimate the required number of agents for each building on fire. All agents have the same tree and it does not change during a simulation.

As previously stated, the *FireStation* agent has a better global view of the situation and therefore it can suggest fire areas to *FireBrigade* agents. The *FireBrigade* agents have however a more accurate local view, consequently they choose which particular building on fire to extinguish in the given area. By doing so, as illustrated on Figure 2, we can take advantage of the better global view of the *FireStation* agent and the better local view of the *FireBrigade* agent at the same time.

5.1 Fire Areas Allocation

To allocate the fire areas, the *FireStation* agent has a list of all fire areas. For each fire area, it has to estimate the number of agents that are required to extinguish this area. To do so, it makes a list of all the buildings that are at the edge of the fire area. Agents only consider buildings at the edge because those are the buildings that have to be extinguished to stop the propagation of the fire. For each burning building at the edge, the agent finds the number of agents required to extinguish the fire (Algorithm 2). It then estimates the required number of agents for the fire area as the maximum number of agents returned for one building in the area. The *FireStation* agent does the same thing with all fire areas, ending up with a number of agents (n_z) for each area.

Afterwards, for each area z, the *FireStation* agent calculates the average distance of the n_z closest *FireBrigade* agents from z. Then, it chooses the fire area z with the smallest average distance. Consequently, the n_z closest *FireBrigade* agents from the chosen z are assigned to z. The *FireStation* agent then removes the assigned area and *FireBrigade* agents from its lists and continues the process with the remaining agents and the remaining fire areas. It continues until there is no agent or fire area left. At the end, the *FireStation* agent sends to each *FireBrigade* agent their assigned fire area.

5.2 Choice of Buildings on Fire

When choosing a building to extinguish, a *FireBrigade* agent builds a list of buildings on fire it knows about in

the fire area specified by the *FireStation*. This list is sorted according to a utility function that gives an idea about the usefulness of extinguishing a fire. The utility function gives a value to a fire based on the buildings and the civilians in danger if this fire propagates to buildings close by.

All *FireBrigade* agents have approximately the same list of buildings on fire. To choose their building on fire they go through the list, one building at a time. For each building, they use the tree to find the expected required number of agents to extinguish the fire, by using Algorithm 2.

The *FireBrigade* agents are assigned to the burning buildings following a prefixed order given to them at the beginning of the simulation. Knowing the sorted list of buildings on fire, the order of all *FireBrigade* agents and the number of agents required for each fire, each *FireBrigade* agent can choose the fire it should extinguish. For example, suppose that there are two fires requiring 5 and 3 agents respectively and that the *FireBrigade* agent A has to choose one of them. If the agent A's rank is 3, it would choose the first fire, because the five first agents have to go to the first fire. However, if the agent A's rank is 7, it would choose the second fire, because the agents ranked 6, 7 and 8 have to go to the second building on fire. With this coordination process, if all *FireBrigade* agents actually have the same information, they should be well coordinated on which buildings to extinguish.

6. RESULTS AND DISCUSSION

As mentioned before, experiments have been done in the RoboCupRescue simulation environment. We have made our tests on a situation with a lot of fires, but with all roads cleared. The simulations started with 8 fires, but the 15 *FireBrigade* agents were beginning to extinguish fires only after 30 simulation steps to leave some time for the fires to propagate. This gave us a hard situation to handle for the *FireBrigade* agents. Those agents started with an empty tree and they learned from one simulation to another to distinguish the tasks that had to be distinguished and the expected rewards associated with those tasks.

If we look back at our model in Section **??** ($< D, N, R, T >$), we can see that D is the set of all possible fire descriptions. In our experiments, a task was described by: the intensity of the fire (3 possible values), the building's composition (3 possible values), the building's size (730 different buildings), the building's damage (4 possible values) and the number of *FireBrigade* agents (15 possible values). Therefore, there were 394 200 possible task descriptions. N is the number of possible *FireBrigade* agents, which was 15. R is the reward function, which was simply 100 to extinguish a fire and 0 otherwise. Finally, T is the transition function which was estimated using the instances stored in the leaves of the tree as explained in Equation 4.

To compare the results obtained by our agents, we have compared them with two other strategies (see Figure 3). The first one is the team ResQFreiburg [1] which finished first at the 2004 RoboCupRescue simulation world competition. This team uses data-mining techniques to evaluate the propagation of the fires and a priority function to choose which fire to extinguish. If we look at the performance of ResQFreiburg, they only obtained an average percentage of intact buildings of 59%.

The second one is our strategy, but without the learning part, thus all agents were going to the first building on the list. This comparison shows the advantage of learning the required number of agents to accomplish a task. If all agents go to the same building, they obtained an average percentage of intact buildings of 63% and after learning, they obtained 84%. This is a huge improvement showing that the information learned is really useful. They learned how many agents are required for all possible situations so they were able to divide the tasks efficiently between the different agents. The improvement in the performances is mainly due to the fact that agents are able to split themselves on the first two or three tasks and accomplish them all at once. Thus, the more efficient they become at estimating the required number of agents, the faster they become at accomplishing all their tasks.

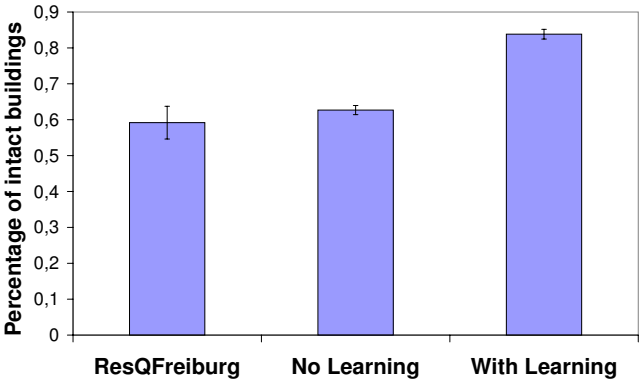


Figure 3: Comparison with other strategies.

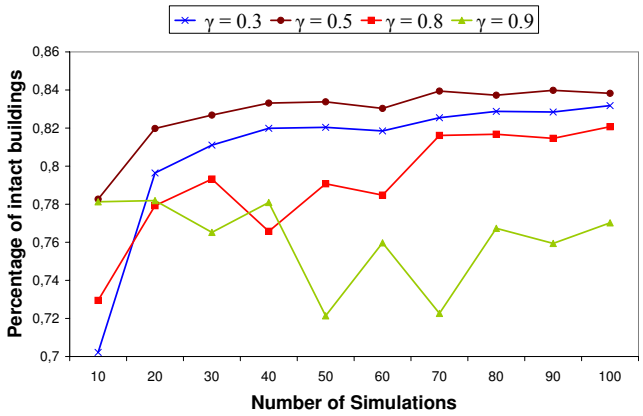


Figure 4: Percentage of intact buildings over 100 simulations for different γ values.

Another interesting result is that our agents were able to attain such good performances with trees having less than 2000 leaves. Therefore, they were just distinguishing 2000 task descriptions out of the 394 200 possible task descriptions. In other word, our agents were able to perform efficiently with an internal task description space of only 0.5% of the complete task description space. This shows a very good reduction of the task description space, enabling the learning algorithm to work on an easier problem.

Moreover, Figure 4 presents results with different γ values for the equation 2. It shows the evolution of the percentage of intact buildings at the end of a simulation. Every point represents an average over 10 simulations. We have tested

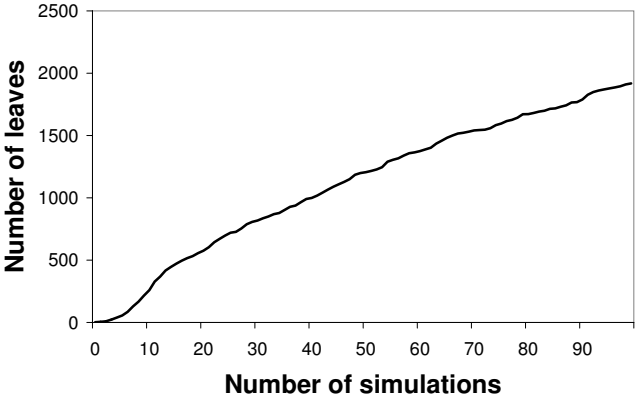


Figure 5: Number of leaves in the tree over 100 simulations.

all γ values from 0 to 1 with an increment of 0.1, but here we present only four representative learning curves. As we can see, with high γ values, the learning process is less effective. With 0.9, the agents do not get better at all, and with 1, it was even worse. With a value of 0.8, the agents have some trouble to learn at the beginning, but eventually they start to catch up after approximately 70 simulations. We have observed that with a higher gamma value, agents normally need more time to learn. With the small γ values of 0.3 and 0.5, the learning curves are quite smooth.

In each simulation, agents were able to gather approximately 2000 instances. Of course, with our algorithm, the necessary memory always grows to store all those instances, but since they were not taking too much space, this was not really a problem. Moreover, after the learning phase, the instances are not necessary anymore, therefore when the tree is used at the execution time it is really small.

Since we are expanding a tree, the growth could be exponential. However, it is not the case, because we are only expanding leaves that help to predict rewards. Therefore, the growth of the tree is controlled and in our tests it was even sub-linear (see Figure 5).

7. RELATED WORKS

In cooperative multiagent learning, most researchers have focused on coordinating agents' actions, but most of them consider that the characteristics of the tasks are known [9, 2]. In our case, an important characteristic of the tasks was unknown, i.e. the required number of agents for each task. Thus we focused our attention on developing an algorithm enabling agents to learn this important task's characteristic. Some researchers consider that the number of agents is unknown, but for very simplified tasks needing only one or two agents [3].

Other researchers have used tree structures to represent and learn the state space of the agent. One of them is McCallum with its U-tree algorithm [5]. He has used a tree structure to store Q-values for all possible basic actions that an agent can take. In our work, we use the tree to calculate the expected reward of a particular task description, considering a certain number of agents accomplishing it. It still has to find the actions to accomplish this task. The learned tree is used to estimate the number of agents needed to accomplish a task and not to find the action of the agent. Also, instead of generating all possible subtrees like McCallum, we use an error reduction estimation measure, borrowed from the decision tree theory [8], that enables us to find good tests. In our case, the generation of subtrees would really be expensive since our state space is very large.

There are different ways that can be used to expand the tree ([5, 6, 10]), but we have used an approach that has been shown to be effective in decision tree algorithms ([8]) and that enabled us to have a fast algorithm and to consider continuous attributes.

The way we manage our instance chains is also quite different. In our work, there is not only one chain, but many chains, one for each attempt to extinguish a fire or a fire zone. Our concept of instance chain is closer to the concept of *episode* described by Xuan, Lesser and Zilberstein [11].

8. CONCLUSION

In this article, we have presented a learning algorithm which is useful to learn the required number of agents for each different tasks in a complex cooperative multiagent environment. To achieve that, we have adapted a selective perception learning algorithm that learns a tree representation of the task description space. Our algorithm enables us to reduce the number of different task descriptions considered by regrouping tasks that have similar rewards. In this context, agents can learn to distinguish only between tasks that really need to be distinguished.

Afterwards, the learned expected rewards are used by our coordination algorithm to find the required number of agents to accomplish a task, both at the global and the local decision level. We added an attribute "number of agents" in the task description, which enabled us to use a greedy algorithm to find the minimum required number of agents to accomplish a given task.

With our learning approach, our agents were able to obtain good results in the RoboCupRescue simulation with an internal task description space of only 0.5% of the complete task description space. All the results presented in this article show that our approach is efficient for learning the required number of agents to accomplish a task and to coordinate agents on those tasks in a partially observable, dynamic and uncertain real-time environment.

9. REFERENCES

[1] M. Brenner, A. Kleiner, M. Exner, M. Degen, M. Metzger, T. Nussle, and I. Thon. ResQ Freiburg: Deliberative Limitation of Damage. In D. Nardi, M. Riedmiller, and C. Sammut, editors, *RoboCup-2004: Robot Soccer World Cup VIII*, Berlin, 2005. Springer Verlag.

[2] C. B. Excelente-Toledo and N. R. Jennings. The Dynamic Selection of Coordination Mechanisms. *Journal of Autonomous Agents and Multi-Agent Systems*, 9(1-2):55–85, 2004.

[3] A. Garland and R. Alterman. Autonomous Agents that Learn to Better Coordinate. *Autonomous Agents and Multi-Agent Systems*, 8(3):267–301, May 2004.

[4] H. Kitano. Robocup rescue: A grand challenge for multi-agent systems. In *Proceedings of ICMAS 2000*, Boston, MA, 2000.

[5] A. K. McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis,

University of Rochester, Rochester, New-York, 1996.

[6] L. D. Pyeatt and A. E. Howe. Decision tree function approximation in reinforcement learning. Technical Report TR CS-98-112, Colorado State University, Fort Collins, Colorado, 1995.

[7] J. R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[8] J. R. Quinlan. Combining instance-based and model-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 236–243, Amherst, Massachusetts, 1993. Morgan Kaufmann.

[9] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, 1998.

[10] W. T. B. Uther and M. M. Veloso. Tree based discretization for continuous state space reinforcement learning. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 769–774, Menlo Park, CA, 1998. AAAI-Press/MIT-Press.

[11] P. Xuan, V. Lesser, and S. Zilberstein. Modeling Cooperative Multiagent Problem Solving as Decentralized Decision Processes. *Autonomous Agents and Multi-Agent Systems*, 2004.

Extensive Games with Possibly Unaware Players*

Joseph Y. Halpern
Computer Science
Cornell University, USA
halpern@cs.cornell.edu

Leandro C. Rêgo
School of Electrical and Computer Engineering
Cornell University, USA
lcr26@cornell.edu

ABSTRACT

Standard game theory assumes that the structure of the game is common knowledge among players. We relax this assumption by considering extensive games where agents may be unaware of the complete structure of the game. In particular, they may not be aware of moves that they and other agents can make. We show how such games can be represented; the key idea is to describe the game from the point of view of every agent at every node of the game tree. We provide a generalization of Nash equilibrium and show that every game with awareness has a generalized Nash equilibrium. Finally, we extend these results to games with *awareness of unawareness*, where a player i may be aware that a player j can make moves that i is not aware of.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Economics, Theory

Keywords

Foundations of Game Theory, Awareness, Solution Concepts

1. INTRODUCTION

Standard game theory models implicitly assume that all significant aspects of the game (payoffs, moves available, etc.) are common knowledge among the players. While such common knowledge may seem unreasonable, there are well-known techniques going back to Harsanyi [7] for transforming a game where some aspects are not common knowledge to one where they are common knowledge. All these techniques assume that players are at least *aware* of all possible moves in the game. However, this is not always a reasonable assumption. For example, sleazy companies assume that consumers are not aware that they can lodge complaints if there are problems; in a war setting, having technology that an enemy is unaware of (and thus being able to make moves that the enemy is unaware of) can be critical; in financial markets, some investors may not be aware of certain investment strategies (complicated hedging strategies, for example, or tax-avoidance strategies).

In a standard game, a set of strategies is a Nash equilibrium if each agent's strategy is a best response to the other agents' strategies, so each agent i would continue playing its strategy even if i knew what strategies the other agents were using. To understand the relevance of adding the possibility of unawareness to the analysis of games, consider the game shown in Figure 1. One Nash equilibrium of this game has A playing across$_A$ and B playing down$_B$. However, suppose that A is not aware that B can play down$_B$. In that case, if A is rational, A will play down$_A$. Therefore, Nash equilibrium does not seem to be the appropriate solution concept here. Although A would play across$_A$ if A knew that B were going to play down$_B$, A cannot even contemplate this possibility, let alone know it.

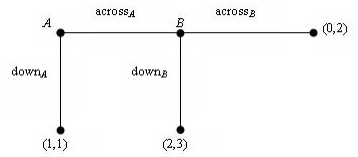


Figure 1: A simple game.

Our goal is to find appropriate solution concepts for extensive games with possibly unaware players, and more generally, to find ways of representing multiagent systems where some agents may not be aware of features of the system. To do this, we must first find an appropriate representation for such games. The first step in doing so is to explicitly represent what players are aware of at each node. We do this by using what we call an *augmented game*. An augmented game describes how awareness changes over time. For example, perhaps A playing across$_A$ will result in B becoming

*This work was supported in part by NSF under grants CTC-0208535, ITR-0325453, and IIS-0534064, by ONR under grants N00014-00-1-03-41 and N00014-01-10-511, and by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the ONR under grant N00014-01-1-0795. The second author was also supported in part by a scholarship from the Brazilian Government through the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

aware of the possibility of playing down_B. In financial settings, one effect of players using certain investment strategies is that other players become aware of the possibility of using that strategy. Strategic thinking in such games must involve taking this possibility into account.

We cannot in general represent what is going on using only one augmented game. The standard representation of a game implicitly assumes that (it is common knowledge that) the modeler and the players all understand the game the same way. This is no longer true once we allow for the possibility of unawareness, since a player's description of the game can now involve only those aspects of the game that he is aware of. Thus, the full description of the game with awareness is given by a set of augmented games, one for the modeler and one for each game that at least one of the agents thinks might be the true game in some situation.

Continuing with the game in Figure 1, the augmented game from the point of view of the type of B that is unaware of the possibility of playing down_B would just include A's moves down_A and across_A and the move across_B. In that augmented game, player A is also unaware of the move down_B. By way of contrast, the augmented game from the point of view of the type of B that is aware of down_B would include the move down_B, but may also allow for the possibility that A is not aware that B is aware of this move.

The standard notion of Nash equilibrium consists of a collection of strategies, one for each player. Our generalization consists of a collection of strategies, one for each pair (i, Γ'), where Γ' is a game that agent i considers to be the true game in some situation. Intuitively, the strategy for a player i at Γ' is the strategy i would play in situations where i believes that the true game is Γ'. To understand why we may need to consider different strategies consider, for example, the game of Figure 1. B would play differently depending on whether or not he was aware of down_B. Roughly speaking, a set of strategies, one for each pair (i, Γ'), is a *generalized Nash equilibrium* if the strategy for (i, Γ') is a best response for player i if the true game is Γ', given the strategies being used by the other players in Γ'.

We argue that this notion of equilibrium correctly captures our intuitions. We then show that every game with awareness has a generalized Nash equilibrium by associating with a game with awareness a standard game (where agents are aware of all moves) such that there is a one-to-one correspondence between generalized Nash equilibria of the game with awareness and Nash equilibria of the standard game.

For ease of exposition, for most of the paper we focus on games where agents are not aware of their lack of awareness. That is, we do not consider games where one player might be aware that there are moves that another player (or even she herself) might be able to make, although she is not aware of what they are. Such awareness of unawareness can be quite relevant in practice. For example, in the war setting described above, even if one side cannot conceive of a new technology available to the enemy, they might believe that there is some move available to the enemy without understanding what that particular move is. This, in turn, may encourage peace overtures. To take another example, an agent might delay making a decision because she considers it possible that she might learn about more possible moves, even if she is not aware of what these moves are.

If we interpret "lack of awareness" as "unable to compute" (cf. [1]), then awareness of unawareness becomes even more significant. Consider a chess game. Although all players understand in principle all the moves that can be made, they are certainly not aware of all consequences of all moves. A more accurate representation of chess would model this computational unawareness explicitly. We provide such a representation.

Roughly speaking, we capture the fact that player i is aware that, at a node h in the game tree, there is a move that j can make she (i) is not aware by having i's subjective representation of the game include a "virtual" move for j at node h. Since i does not understand what can happen after this move, i does not develop the game tree after this move; i simply describes what she believes will be the payoffs after the move. Thus, our representation is quite similar in spirit to how chess programs analyze chess games. They explore the game tree up to a certain point, and then evaluate the board position at that point. We can think of the payoffs following a virtual move by j in i's subjective representation of a chess game as describing the evaluation of the board from i's point of view. This seems like a much more reasonable representation of the game than the standard complete game tree!

Recently, Feinberg [2, 3] also studied games with awareness. Feinberg [3] gives a definition of extended Nash equilibrium in normal-form games. Although his definition stems from much the same intuitions as ours, it is expressed syntactically. Each player is characterized by a complete description of what moves and players he is aware of, what moves and players he is aware that each other player is aware of, and so on through all levels of iterated awareness. Feinberg [2] deals with extensive-form games and defines solution concepts only indirectly, via a syntactic epistemic characterization. His approach lacks a more direct semantic framework, which our model provides. He also does not deal with awareness of unawareness.

2. MODELING AWARENESS

The first step in dealing with awareness is modeling it. To this end, we consider *augmented games*. We start with a standard game, described by a game tree Γ (as in Figure 1). An augmented game Γ^+ *based on* Γ essentially augments Γ by describing each agent's *awareness level* at each node, where player i's awareness level at a node h is essentially the set of runs (complete histories) in Γ that i is aware of at node h. A player's awareness level may change over time, as the player becomes aware of more moves.

Our formal definition of augmented game is based on the definition of extensive game given by Osborne and Rubinstein [8]. We start by briefly reviewing Osborne and Rubinstein's definition.

A *(finite) extensive game* is a tuple $(N, M, H, P, f_c, \{\mathcal{I}_i : i \in N\}, \{u_i : i \in N\})$, where

- N is a finite set consisting of the players of the game.
- M is a finite set whose elements are the moves available to players (and nature) during the game.[1]
- H is a finite set of finite sequences of moves (elements of M) that is closed under prefixes, so that if $h \in H$

[1]Osborne and Rubinstein did not make M explicit in their definition of an extensive game; we find it convenient to make it explicit here.

and h' is a prefix of H, then $h' \in H$. Intuitively, each member of H is a *history*. We can identify the nodes in a game tree with the histories in H. Each node n is characterized by the sequence of moves needed to reach n. A *run* in H is a terminal history, one that is not a strict prefix of any other history in H. Let Z denote the set of runs of H. Let $M_h = \{m \in M : h \cdot \langle m \rangle \in H\}$ (where we use $\cdot$ to denote concatenation of sequences); M_h is the set of moves that can be made after history h.

- $P : (H - Z) \to N \cup \{c\}$ is a function that assigns to each nonterminal history h a member of $N \cup \{c\}$. (We can think of c as representing nature.) If $P(h) = i$, then player i moves after history h; if $P(h) = c$, then nature moves after h. Let $H_i = \{h : P(h) = i\}$ be the set of all histories after which player i moves.
- f_c is a function that associates with every history for which $P(h) = c$ a probability measure $f_c(\cdot \mid h)$ on M_h. Intuitively, $f_c(\cdot \mid h)$ describes the probability of nature's moves once history h is reached.
- $\mathcal{I}_i$ is a partition of H_i with the property that if h and h' are in the same cell of the partition then $M_h = M_{h'}$, i.e., the same set of moves is available for every history in a cell of the partition. Intuitively, if h and h' are in the same cell of $\mathcal{I}_i$, then h and h' are indistinguishable from i's point of view; i considers history h' possible if the actual history is h, and vice versa. A cell $I \in \mathcal{I}_i$ is called an (i-)*information set*.
- $u_i : Z \to \mathrm{R}$ is a payoff function for player i, assigning a real number (i's payoff) to each run of the game.

In the game of Figure 1,

- $N = \{A, B\}$, $H = \{\langle\rangle, \langle \mathrm{down}_A \rangle, \langle \mathrm{across}_A, \mathrm{down}_B \rangle, \langle \mathrm{across}_A, \mathrm{across}_B \rangle\}$,
- $P(\langle\rangle) = A$, $P(\langle \mathrm{across}_A \rangle) = B$,
- $I_A = \{\langle\rangle\}$, $I_B = \{\langle \mathrm{across}_A \rangle\}$,
- $u_A(\langle \mathrm{down}_A \rangle) = u_B(\langle \mathrm{down}_A \rangle) = 1$,
- $u_A(\langle \mathrm{across}_A, \mathrm{across}_B \rangle) = 0$, and
- $u_B(\langle \mathrm{across}_A, \mathrm{across}_B \rangle) = 2$.

In this paper, as in most work in game theory, we further assume that players have *perfect recall*: they remember all the actions that they have performed and the information sets they passed through. Formally, we require that

- if h and h' are in the same i-information set and h_1 is a prefix of h such that $P(h_1) = i$, then there is a prefix h'_1 of h' such that h_1 and h'_1 are in the same information set; moreover, if $h_1 \cdot \langle m \rangle$ is a prefix of h (so that m was the action performed when h_1 was reached in h) then $h'_1 \cdot \langle m \rangle$ is a prefix of h'.

An *augmented game* is defined much like an extensive game; the only essential difference is that at each nonterminal history we not only determine the player moving but also her awareness level. Since the awareness level is a set of runs in a game Γ, we say that $\Gamma^+ = (N^+, M^+, H^+, P^+, f_c^+, \{\mathcal{I}_i^+ : i \in N^+\}, \{u_i^+ : i \in N^+\}, \{A_i^+ : i \in N^+\})$ is an *augmented game based on the (standard) extensive game* $\Gamma = (N, M, H, P, f_c, \{\mathcal{I}_i : i \in N\}, \{u_i : i \in N\})$ if the following conditions are satisfied:

A1. $(N^+, M^+, H^+, P^+, f_c^+, \{\mathcal{I}_i^+ : i \in N^+\}, \{u_i^+ : i \in N^+\})$ is a finite extensive game with perfect recall.

A2. $A_i^+ : H_i^+ \to 2^H$ describes i's awareness level at each nonterminal history after which player i moves. For each $h \in H_i^+$, $A_i^+(h)$ consists of a set of histories in H and all their prefixes. Intuitively, $A_i^+(h)$ describes the set of histories of Γ that i is aware of at history $h \in H_i^+$. (Having $A_i^+(h)$ consist of histories rather than just runs makes it easier to deal with awareness of unawareness.)

A3. $N^+ \subseteq N$.

A4. If $P^+(h) \in N^+$, then $P^+(h) = P(\overline{h})$, where $\overline{h}$ is the subsequence of h consisting of all the moves in h that are also in M, and $M_h^+ \subseteq M_{\overline{h}}$. Intuitively, all the moves available to i at h must also be available to i in the underlying game Γ.

A5. If $P^+(h) = c$, then either $P(\overline{h}) = c$ and $M_h^+ \subseteq M_{\overline{h}}$, or $M_h^+ \cap M = \emptyset$. The moves in M_h^+ in the case where $M_h^+ \cap M = \emptyset$ intuitively capture uncertainty regarding a player's awareness level.

A6. If h and h' are in the same information set in $\mathcal{I}_i^+$, then $A_i^+(h) = A_i^+(h')$. Intuitively, i's awareness level depends only on the information that i has.

A7. If h is a prefix of h' and $P^+(h) = P^+(h')$, then $A_i^+(h) \subseteq A_i^+(h')$. This is a perfect recall requirement; players do not forget histories that they were aware of.

A8. If h and h' are in the same information set in Γ^+, then $\overline{h}$ and $\overline{h}'$ are in the same information set in Γ.

A9. If h and h' are histories in both Γ^+ and Γ (i.e., $h = \overline{h}$ and $h' = \overline{h}'$), and $\overline{h}$ and $\overline{h}'$ are in the same information set in Γ, then h and h' are in the same information set in Γ^+.

A10. For all $i \in N^+$ and $h \in H_i^+$, if $h', h'' \in A_i(h)$, h' and h'' are in the same information set in Γ, then $h' \cdot \langle m \rangle \in A_i(h)$ iff $h'' \cdot \langle m \rangle \in A_i(h)$.

A11. $\{\overline{z} : z \in Z^+\} \subseteq Z$; moreover, for all $i \in N^+$, $h \in H_i^+$, if z is a terminal history in $A_i^+(h)$ (i.e., if $z \in A_i(h)^+$ and z is not a strict prefix of another element of $A_i(h)^+$), then $z \subseteq Z$. Thus, the runs in Z^+ correspond to runs in Z, and players understand this fact.

A12. For all $i \in N^+$ and runs z in Z^+, if $\overline{z} \in Z$, then $u_i^+(z) = u_i(\overline{z})$. Thus, a player's utility just depends on the moves made in the underlying game. (By A11, we have $\overline{z} \in Z$. We have included the clause "if $\overline{z} \in Z$" so that A12 is applicable when we consider awareness of unawareness, where we drop A11.)

Conditions A1–A12 are intended to capture our intuitions regarding information sets, awareness, and common knowledge. To allow us to focus on issues directly related to awareness, we have implicitly assumed that there is common knowledge of (1) who moves at histories in the underlying game (this is captured by the fact that $P^+(h) = P(\overline{h})$ unless $P^+(h) = c$ and $M_h^+ \cap M = \emptyset$), (2) what the payoffs are in the underlying game (since $u^+(\overline{z}) = u(z)$), and (3) what the information sets are in the underlying game (see A8–A10).

Our approach is flexible enough to allow us to drop these assumptions; see the full paper for details.

To fully understand A8–A10, we must first discuss our view of information sets. As pointed out by Halpern [5], special attention must be given to the interpretation of information sets in game trees. This issue requires even more care in games with awareness. The standard intuition for information sets is that a player considers all the histories in his information set possible. But this intuition does not apply in augmented games. In an augmented game, there may be some histories in an i-information set that i is not aware of; player i cannot consider these histories possible. We interpret an i-information set to be the set of all histories where player i receives the same sequence of signals. A8-A10 reflect this interpretation under our assumption that (modulo awareness) information sets are common knowledge.

For the remainder of the paper, we use the following notation: for a (standard or augmented) game Γ^s, we denote the components of Γ^s with the same superscript s, so that we have M^s, H^s, and so on. Thus, from here on we do not explicitly describe the components of a game.

An augmented game describes either the modeler's view of the game or the subjective view of the game of one of the players, and includes both moves of the underlying game and moves of nature that change awareness. For example, consider again the game shown in Figure 1 and suppose that

- players A and B are aware of all histories of the game;
- player A is uncertain as to whether player B is aware of run $\langle \text{across}_A, \text{down}_B \rangle$ and believes that he is unaware of it with probability p; and
- the type of player B that is aware of the run $\langle \text{across}_A, \text{down}_B \rangle$ is aware that player A is aware of all histories, and he knows A is uncertain about his awareness level and knows the probability p.

Because A and B are actually aware of all histories of the underlying game, from the point of view of the modeler, the augmented game is essentially identical to the game described in Figure 1, with the awareness level of both players A and B consisting of all histories of the underlying game. However, when A moves at the node labeled A in the modeler's game, she believes that the actual augmented game is Γ^A, as described in Figure 2. In Γ^A, nature's initial move captures A's uncertainty about B's awareness level. At the information set labeled $A.1$, A is aware of all the runs of the underlying game. Moreover, at this information set, A believes that the true game is Γ^A.

At the node labeled $B.1$, B is aware of all the runs of the underlying game and believes that the true game is the modeler's game; but at the node labeled $B.2$, B is not aware that he can play down_B, and so believes that the true game is the augmented game Γ^B described in Figure 3. At the nodes labeled $A.3$ and $B.3$ in the game Γ^B, neither A nor B is aware of the move down_B. Moreover, both players think the true game is Γ^B.

As this example should make clear, to model a game with possibly unaware players, we need to consider not just one augmented game, but a collection of them. Moreover, we need to describe, at each history in an augmented game, which augmented game the player playing at that history believes is the actual augmented game being played.

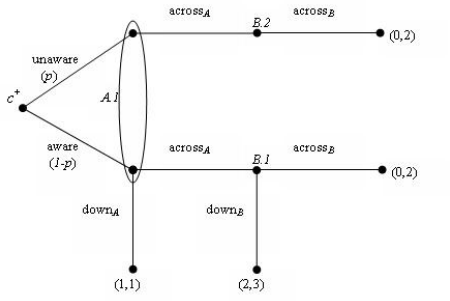


Figure 2: The augmented game Γ^A.

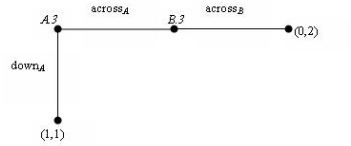


Figure 3: The augmented game Γ^B.

To capture these intuitions, we define a *game with awareness based on* $\Gamma = (N, M, H, P, f_c, \{\mathcal{I}_i : i \in N\}, \{u_i : i \in N\})$ to be a tuple $\Gamma^* = (\mathcal{G}, \Gamma^m, \mathcal{F})$, where

- $\mathcal{G}$ is a countable set of augmented games based on Γ, of which one is Γ^m;
- $\mathcal{F}$ maps an augmented game $\Gamma^+ \in \mathcal{G}$ and a history h in Γ^+ such that $P^+(h) = i$ to a pair (Γ^h, I), where $\Gamma^h \in \mathcal{G}$ and I is an i-information set in game Γ^h.

Intuitively, Γ^m is the game from the point of view of an omniscient modeler. If player i moves at h in game $\Gamma^+ \in \mathcal{G}$ and $\mathcal{F}(\Gamma^+, h) = (\Gamma^h, I)$, then Γ^h is the game that i believes to be the true game when the history is h, and I consists of the set of histories in Γ^h he currently considers possible. For example, in the examples described in Figures 2 and 3, taking Γ^m to be the augmented game in Figure 1, we have $\mathcal{F}(\Gamma^m, \langle\,\rangle) = (\Gamma^A, I)$, where I is the information set labeled $A.1$ in Figure 2, and $\mathcal{F}(\Gamma^A, \langle \text{unaware}, \text{across}_A \rangle) = (\Gamma^B, \{\langle \text{across}_A \rangle\})$.

It may seem that by making $\mathcal{F}$ a function we cannot capture a player's uncertainty about the game being played. However, we can capture such uncertainty by folding it into nature's move. For example, we capture A's uncertainty about whether B is aware of being able to move down_B in the augmented game Γ^A illustrated in Figure 2 by having nature decide this at the first step. It should be clear that this gives a general approach to capturing such uncertainty.

The augmented game Γ^m and the mapping $\mathcal{F}$ must satisfy a number of consistency conditions. The first set of conditions applies to Γ^m. Since the modeler is presumed to be omniscient, the conditions say that the modeler is aware of all the players and moves of the underlying game.

M1. $N^m = N$.

M2. $M \subseteq M^m$ and $\{\overline{z} : z \in Z^m\} = Z$.

M3. If $P^m(h) \in N$, then $M^m_h = M_{\overline{h}}$. If $P^m(h) = c$, then either $M^m_h \cap M = \emptyset$ or $M^m_h = M_{\overline{h}}$ and $f^m_c(\cdot \mid h) = f_c(\cdot \mid \overline{h})$.

M1, M2 and M3 enforce the intuition that the modeler understands the underlying game. He knows all the players and possible moves, and understands how nature's moves work in the underlying game Γ. It may seem somewhat surprising that there is no analogue of the second part of M3 (i.e., the constraint of f_c^m) for all augmented games. While it makes sense to have such an analogue if nature's moves are in some sense objective, it seems like an unreasonable requirement that all player's should agree on these probabilities in general. This is especially so in the case that a player suddenly becomes aware of some moves of nature that he was not aware of before. It does not seem reasonable to assume that this awareness should come along with an understanding of the probabilities of these moves. Of course, we could require such an analogue of M3. Since the set of games that have such a requirement is a subset of the games we consider, all our results apply without change if such a requirement is imposed.

Although the modeler understands the underlying game Γ, Γ^m is not uniquely determined by Γ. There may be many modeler's games based on Γ, where the players have different awareness levels and the awareness changes in different ways.

The game Γ^m can be thought of as a description of "reality"; it describes the effect of moves in the underlying game and how players' awareness levels change. The other games in $\mathcal{G}$ describe a player's subjective view of the situation. The constraints on the mapping $\mathcal{F}$ that we now describe capture desirable properties of awareness.

Consider the following constraints, where $\Gamma^+ \in \mathcal{G}$, $h \in H^+$, $P^+(h) = i$, $A_i^+(h) = a$, and $\mathcal{F}(\Gamma^+, h) = (\Gamma^h, I)$.

C1. $\{\overline{h} : h \in H^h\} = a$.

C2. If $h' \in H^h$ and $P^h(h') = j$, then $A_j^h(h') \subseteq a$ and $M_{\overline{h}'} \cap \{m : \overline{h}' \cdot \langle m \rangle \in a\} = M_{h'}^h$.

C3. If h' and h are in the same information set in Γ^+ and $\overline{h}' \in a$, then there exists $h'' \in I$ such that $\overline{h}'' = \overline{h}'$.

C4. If $h' \in I$, then $A_i^h(h') = a$ and $\mathcal{F}(\Gamma^h, h') = (\Gamma^h, I)$.

C5. If $h' \in H^+$, $P^+(h') = i$, $A_i^+(h') = a$, then if h and h' are in the same information set of Γ^+, then $\mathcal{F}(\Gamma^+, h') = (\Gamma^h, I)$, while if h is a prefix or a suffix of h', then $\mathcal{F}(\Gamma^+, h') = (\Gamma^h, I')$ for some i-information set I'.

C6. If $h' \in I$, then $\overline{h}$ and $\overline{h}'$ are in the same information set in Γ;

C7. If $\Gamma^h = \Gamma^+$, then $h' \in I$ iff h and h' are in the same i-information set in Γ^+.

C8. For all histories $h' \in I$, there exists a prefix h_1' of h' such that $P^h(h_1') = i$ and $\mathcal{F}(\Gamma^h, h_1') = (\Gamma', I')$ iff there exists a prefix h_1 of h such that $P^+(h_1) = i$ and $\mathcal{F}(\Gamma^+, h_1) = (\Gamma', I')$. Moreover, $h_1' \cdot \langle m \rangle$ is a prefix of h' iff $h_1 \cdot \langle m \rangle$ is a prefix of h.

C9. There exists a history $h' \in I$ such that for every prefix $h'' \cdot \langle m \rangle$ of h', if $P^h(h'') = j \in N^h$ and $\mathcal{F}(\Gamma^h, h'') = (\Gamma', I')$, then for all $h_1 \in I'$, $h_1 \cdot \langle m \rangle \in H'$.

C10. If h' and h'' are histories in both Γ^+ and Γ^h, then h' and h'' are in the same i-information set in Γ^+ iff h' and h'' are in the same i-information set in Γ^h.

Suppose that $\mathcal{F}(\Gamma^+, h) = (\Gamma^h, I)$. Player i moving at history h in Γ^+ thinks the actual game is Γ^h. Moreover, i thinks he is in the information set of I of Γ^h. C1 guarantees that the set of histories of the underlying game player i is aware of is exactly the set of histories of the underlying game that appear in Γ^h. C2 states that no player in Γ^h can be aware of histories not in a. The second part of C2 implies that the set of moves available for player j at h' is just the set of moves that player i is aware of that are available for j at $\overline{h}'$ in the underlying game. C3 guarantees that for all histories h' indistinguishable from h that player i is aware of, there exists some history $h'' \in I$ differing from h' at most in some moves of nature that change awareness levels. C4 says that at all histories in I player i indeed thinks the game is Γ^h and that the information set is I. C5 says that player i's subjective view of the game changes only if i becomes aware of more moves and is the same at histories in H^+ that i cannot distinguish. C6 captures the assumption that at all histories i considers possible, he must have gotten the same signals as he does in the actual history.

C7 says that if while moving at history h player i thinks that Γ^+ is the actual game, then he considers possible all and only histories in the information set containing h. C8 is a consequence of the perfect recall assumption. C8 says that if, at history h, i considers h' possible, then for every prefix h_1' of h' there is a corresponding prefix of h where i considers himself to be playing the same game, and similarly, for every prefix of h there is a prefix of h' where i considers himself to be playing the same game. Moreover, i makes the same move at these prefixes.

The intuition behind condition C9 is that player i knows that player j only make moves that j is aware of. Therefore, player i must consider at least one history h' where he believes that every player j made a move that j was aware of. It follows from A11, C1, C2, and C9 that there is a run going through I where every player j makes a move that player i believes that j is aware of.

Since we assume that players have (modulo awareness) common knowledge about information sets, if Γ^+ is the game from the point of view of player j (or the modeler) and there are histories h' and h'' in both Γ^+ and Γ^h, then player j (or the modeler) knows that player i gets the same signals in both h' and h'' iff he knows that player i knows that he gets the same signals in those histories. C10 captures that intuition.

Just as Γ^m is not uniquely determined by Γ, $\mathcal{F}(\Gamma^+, h)$ is not uniquely determined by Γ^+ and h. Moreover, even if $A_i(h) = A_i(h')$, we may have $\mathcal{F}(\Gamma^+, h) = (\Gamma^h, I)$ and $\mathcal{F}(\Gamma^+, h') = (\Gamma^{h'}, I')$ with $\Gamma^h \neq \Gamma^{h'}$. We do not require that the awareness level determines the game a player considers possible. This extra flexibility allows us to model a situation where, for example, players 2 and 3, who have the same awareness level and agree on the awareness level of player 1, have different beliefs about the game player 1 considers possible.

A standard extensive game Γ can be identified with the game $(\{\Gamma^m\}, \Gamma^m, \mathcal{F})$, where (abusing notation slightly) $\Gamma^m = (\Gamma, \{A_i : i \in N\})$ and, for all histories h in an i-information set I in Γ, $A_i(h) = H$ and $\mathcal{F}(\Gamma^m, h) = (\Gamma^m, I)$. Thus, all players are aware of all the runs in Γ, and agree with each other and the modeler that the game is Γ. We call this the *canonical representation of* Γ as a game with awareness.

One technical issue: We have assumed that the set $\mathcal{G}$ of

games in a game Γ^* with awareness is countable. For our purposes, this is without loss of generality. We are ultimately interested in what happens in the game Γ^m, since this is the game actually being played. However, to analyze that, we need to consider what happens in other games in $\mathcal{G}$. For example, if h is a history in Γ^m where i moves, we need to understand what happens in the game Γ^h such that $\mathcal{F}(\Gamma^m, h) = (\Gamma^h, \cdot)$, since Γ^h is the game that i thinks is being played at history h in Γ^m. It is not hard to see that the set of games we need to consider is the least set $\mathcal{G}'$ such that $\Gamma^m \in \mathcal{G}'$ and, for every $\Gamma' \in \mathcal{G}$ and history h in Γ' such that $\mathcal{F}(\Gamma', h) = (\Gamma'', \cdot)$, $\Gamma'' \in \mathcal{G}'$. $\mathcal{G}'$ is guaranteed to be countable, even if $\mathcal{G}$ is not.

Finally, we remark that our notion of a game with awareness as consisting of the modeler's game together with description of which game each agent thinks is the actual game at each history has much in common with the intuition behind Gal and Pfeffer's [4] notion *Networks of Influence Diagrams*; we discuss the connections in detail in the full paper.

3. LOCAL STRATEGIES AND GENERALIZED NASH EQUILIBRIUM

3.1 Local Strategies

In this section, we generalize the notion of Nash equilibrium to games with awareness. To do that, we must first define what a strategy is in a game with awareness. Recall that in a standard game, a strategy for player i is a function from i-information sets to a move or to a distribution over moves, depending on whether we are considering *pure* (i.e., deterministic) strategies or *behavioral* (i.e., randomized) strategies. The intuition is that player i's actions depend on what i knows; the strategy can be viewed as a *universal plan*, describing what i will do in every possible situation that can arise. This makes sense only because i is presumed to know the game tree, and thus to know in advance all the situations that can arise.

In games with awareness, this intuition no longer makes sense. For example, player i cannot plan in advance for what will happen if he becomes aware of something he is initially unaware of. We must allow i's strategy to change if he becomes aware of more moves. Let $\mathcal{G}_i = \{\Gamma' \in \mathcal{G} :$ for some $\Gamma^+ \in \mathcal{G}$ and h in Γ^+, $P^+(h) = i$ and $\mathcal{F}(\Gamma^+, h) = (\Gamma', \cdot)\}$. Intuitively, $\mathcal{G}_i$ consists of the games that i views as the real game in some history.Thus, rather than considering a single strategy in a game $\Gamma^* = (\mathcal{G}, \Gamma^m, \mathcal{F})$ with awareness, we consider a collection $\{\sigma_{i,\Gamma'} : \Gamma' \in \mathcal{G}_i\}$ of what we call *local strategies*, one for each augmented game in $\mathcal{G}_i$. Intuitively, a local strategy $\sigma_{i,\Gamma'}$ for game Γ' is the strategy that i would use if i were called upon to play and i thought that the true game was Γ'. Thus, the domain of $\sigma_{i,\Gamma'}$ consists of pairs (Γ^+, h) such that $\Gamma^+ \in \mathcal{G}$, h is a history in Γ^+, $P^+(h) = i$, and $\mathcal{F}(\Gamma^+, h) = (\Gamma', I)$.

Define an equivalence relation $\sim_i$ on pairs (Γ', h) such that $\Gamma' \in \mathcal{G}$ and h is a history in Γ' where i moves such that $(\Gamma_1, h_1) \sim_i (\Gamma_2, h_2)$ if $\mathcal{F}(\Gamma_1, h_1) = \mathcal{F}(\Gamma_2, h_2)$. We can think of $\sim_i$ as defining a *generalized information partition* in Γ^*. It is easy to check that a $\sim_i$ equivalence class consists of a union of i-information sets in individual games in $\mathcal{G}$. Moreover, if some element of a $\sim_i$ equivalence class is in the domain of $\sigma_{i,\Gamma'}$, then so is the whole equivalence class. At all pairs (Γ', h') in a $\sim_i$ equivalence class, if $\mathcal{F}(\Gamma', h') = (\Gamma^{h'}, I)$, player i thinks he is actually playing in the information set I of $\Gamma^{h'}$. Thus, we require that $\sigma_{i,\Gamma'}(\Gamma_1, h_1) = \sigma_{i,\Gamma'}(\Gamma_2, h_2)$ if (Γ_1, h_1) and (Γ_2, h_2) are both in the domain of $\sigma_{i,\Gamma'}$ and $(\Gamma_1, h_1) \sim_i (\Gamma_2, h_2)$.

The following definition summarizes this discussion.

DEFINITION 3.1. *Given a game with awareness $\Gamma^* = (\mathcal{G}, \Gamma^m, \mathcal{F})$, a* local strategy $\sigma_{i,\Gamma'}$ *for agent i is a function mapping pairs (Γ^+, h) such that h is a history where i moves in Γ^+ and $\mathcal{F}(\Gamma^+, h) = (\Gamma', I)$ to a probability distribution over $M'_{h'}$, the moves available at a history $h' \in I$, such that $\sigma_{i,\Gamma'}(\Gamma_1, h_1) = \sigma_{i,\Gamma'}(\Gamma_2, h_2)$ if $(\Gamma_1, h_1) \sim_i (\Gamma_2, h_2)$.*

Note that there may be no relationship between the strategies $\sigma_{i,\Gamma'}$ for different games Γ'. Intuitively, this is because discovering about the possibility of a different move may cause agent i to totally alter his strategy. We could impose some consistency requirements, but we have not found any that we believe should hold in all games. We believe that all our results would continue to hold in the presence of reasonable additional requirements, although we have not explored the space of such requirements.

3.2 Generalized Nash Equilibrium

We want to define a notion of generalized Nash equilibrium so as to capture the intuition that for every player i, if i believes he is playing game Γ', then his local strategy $\sigma_{i,\Gamma'}$ is a best response to the local strategies of other players in Γ'.

Define a *generalized strategy profile* of $\Gamma^* = (\mathcal{G}, \Gamma^m, \mathcal{F})$ to be a set of local strategies $\vec{\sigma} = \{\sigma_{i,\Gamma'} : i \in N, \Gamma' \in \mathcal{G}_i\}$. Let $EU_{i,\Gamma'}(\vec{\sigma})$ be the expected payoff for i in the game Γ' given that strategy profile $\vec{\sigma}$ is used. Note that the only strategies in $\vec{\sigma}$ that are needed to compute $EU_{i,\Gamma'}(\vec{\sigma})$ are the strategies actually used in Γ'; indeed, all that is needed is the restriction of these strategies to information sets that arise in Γ'.

A *generalized Nash equilibrium* of $\Gamma^* = (\mathcal{G}, \Gamma^m, \mathcal{F})$ is a generalized strategy profile $\vec{\sigma}$ such that for all $\Gamma' \in \mathcal{G}_i$, the local strategy $\sigma_{i,\Gamma'}$ is a best response to $\vec{\sigma}_{-(i,\Gamma')}$, where $\vec{\sigma}_{-(i,\Gamma')}$ is the set of all local strategies in $\vec{\sigma}$ except $\sigma_{i,\Gamma'}$.

DEFINITION 3.2. *A generalized strategy profile $\vec{\sigma}^*$ is a* generalized Nash equilibrium *of a game $\Gamma^* = (\mathcal{G}, \Gamma^m, \mathcal{F})$ with awareness if, for every player i, game $\Gamma' \in \mathcal{G}_i$, and local strategy σ for i in Γ',*

$$EU_{i,\Gamma'}(\vec{\sigma}^*) \geq EU_{i,\Gamma'}((\vec{\sigma}^*_{-(i,\Gamma')}, \sigma)).$$

The standard definition of Nash equilibrium would say that $\vec{\sigma}$ is a Nash equilibrium if σ_i is a best response to $\vec{\sigma}_{-i}$. This definition implicitly assumes that player i can choose a whole strategy. This is inappropriate in our setting. An agent cannot anticipate that he will become aware of more moves. Essentially, if $\Gamma_1 \neq \Gamma_2$, we are treating player i who considers the true game to be Γ_1 to be a different agent from the version of player i who considers Γ_2 to be the true game. To understand why this is appropriate, suppose that player i considers Γ_1 to be the true game, and then learns about more moves, and so considers Γ_2 to be the true game. At that point, it is too late for player i to change the strategy he was playing when he thought the game was Γ_1. He should just try to play optimally for what he now considers the true game. Moreover, while player i thinks that the game Γ_1 is the true game, he never considers it possible that

he will ever be playing a different game, so that he cannot "prepare himself" for a change in his subjective view of the game.[2]These considerations suggest that our notion of Nash equilibrium is appropriate.

It is easy to see that $\vec{\sigma}$ is a Nash equilibrium of a standard game iff $\vec{\sigma}$ is a (generalized) Nash equilibrium of the canonical representation of Γ as a game with awareness. Thus, our definition of generalized Nash equilibrium generalizes the standard definition.

Consider the game with awareness shown in Figures 1 (taking this to be Γ^m), 2, and 3. We have $\mathcal{G}_A = \{\Gamma^A, \Gamma^B\}$ and $\mathcal{G}_B = \{\Gamma^m, \Gamma^B\}$. Taking $dom(\sigma_{i,\Gamma'})$ to denote the domain of the strategy $\sigma_{i,\Gamma'}$, we have

$dom(\sigma_{A,\Gamma^A}) = \{(\Gamma^m, \langle\,\rangle), (\Gamma^A, \langle\text{unaware}\rangle), (\Gamma^A, \langle\text{aware}\rangle)\}$,
$dom(\sigma_{B,\Gamma^m}) = \{(\Gamma^m, \langle\text{across}_A\rangle), (\Gamma^A, \langle\text{aware, across}_A\rangle)\}$,
$dom(\sigma_{A,\Gamma^B}) = \{(\Gamma^B, \langle\,\rangle)\}$, and
$dom(\sigma_{B,\Gamma^B}) = \{(\Gamma^A, \langle\text{unaware,across}_A\rangle), (\Gamma^B, \langle\text{across}_A\rangle)\}$.

Each of these domains consists of a single generalized information set. If $p < 1/2$, then the unique generalized Nash equilibrium is $\sigma_{A,\Gamma^A} = \text{across}_A$, $\sigma_{A,\Gamma^B} = \text{down}_A$, $\sigma_{B,\Gamma^m} = \text{down}_B$, $\sigma_{B,\Gamma^B} = \text{across}_B$. Thus, in the modeler's game, A plays across_A, B plays down_B, and the resulting payoff vector is $(2, 3)$. On the other hand, if $p > 1/2$, then the unique generalized Nash equilibrium is $\sigma_{A,\Gamma^A} = \text{down}_A$, $\sigma_{A,\Gamma^B} = \text{down}_A$, $\sigma_{B,\Gamma^m} = \text{down}_B$, $\sigma_{B,\Gamma^B} = \text{across}_B$. Thus, in the modeler's game, A plays down_A, and the payoff vector is $(1, 1)$. Intuitively, even though both A and B are aware of all the moves in the modeler's game, A considers it sufficiently likely that B is not aware of down_B, so A plays down_A.

We now show that every game with awareness has at least one generalized Nash equilibrium. We proceed as follows. Given a game $\Gamma^* = (\mathcal{G}, \Gamma^m, \mathcal{F})$ with awareness, let ν be a probability on $\mathcal{G}$ that assigns each game in $\mathcal{G}$ positive probability. (Here is where we use the fact that $\mathcal{G}$ is countable.) We construct a standard extensive game Γ^ν by essentially "gluing together" all the games $\Gamma' \in \mathcal{G}$, except that we restrict to the histories in Γ' that can actually be played according to the players' awareness level. The components of Γ^ν are independent of ν except for nature's initial move (as encoded by f_c^ν). In Γ^ν, the set of players is $\{(i, \Gamma') : \Gamma' \in \mathcal{G}_i\}$. The game tree of Γ^ν can be viewed as the union of the pruned game trees of $\Gamma' \in \mathcal{G}$. The histories of Γ^ν have the form $\langle\Gamma^h\rangle \cdot h$, where $\Gamma^h \in \mathcal{G}$ and h is a "pruned" history in H^h. The move that a player or nature makes at a history $\langle\Gamma^h\rangle \cdot h$ of Γ^ν is the same as the move made at h when viewed as a history of Γ^h. The only move in Γ^ν not determined by Γ^* is nature's initial move (at the history $\langle\,\rangle$), where nature chooses the game $\Gamma' \in \mathcal{G}$ with probability $\nu(\Gamma')$. We leave the details of the construction of Γ^ν to the full paper.

THEOREM 3.1. *For all probability measures ν on $\mathcal{G}$*

(a) Γ^ν is a standard extensive game with perfect recall; and

(b) if ν gives positive probability to all games in $\mathcal{G}$, then $\vec{\sigma}$ is a Nash equilibrium of Γ^ν iff $\vec{\sigma}'$ is a generalized Nash equilibrium of Γ^*, where $\sigma_{i,\Gamma'}(\langle\Gamma^h\rangle \cdot h') = \sigma'_{i,\Gamma'}(\Gamma^h, h')$.

[2]In games with awareness of unawareness, an agent may consider it possible that he will become aware of more information. But this too is incorporated in his view of the game, so he can still do no better than playing optimally in his current view of the game.

Although a Nash equilibrium does not necessarily exist in games with infinitely many players, Γ^ν has three special properties: (a) each player has only finitely many information sets, and (b) for each player (i, Γ'), there exists a finite subset $N(i, \Gamma')$ of N^ν such that (i, Γ)'s payoff in Γ^ν depends only on the strategies of the players in $N(i, \Gamma')$, and (c) Γ^ν is a game with perfect recall. This turns out to be enough to show that Γ^ν has at least one Nash equilibrium. Thus, we get the following corollary to Theorem 3.1.

COROLLARY 3.1. *Every game with awareness has a generalized Nash equilibrium.*

4. MODELING AWARENESS OF UNAWARENESS

In this section, we describe how to extend our representation of games with awareness to deal with awareness of unawareness. In an augmented game that represents player i's subjective view of the game, we want to model the fact that i may be aware of the fact that j can make moves at a history h that i is not aware of. We do this by allowing j to make a "virtual move" at history h. A virtual move is always a terminal move; associated with it there is a payoff vector. Intuitively, these payoffs are agent i's subjective feeling about what would happen if some move she is unaware of is made by agent j. Player j may have several virtual moves available at history h, and may make virtual moves at a number of histories in the augmented game. These virtual moves do not necessarily correspond to a move in the underlying game Γ (i.e., i may falsely believe that j can make a move at h that he is unaware of), and even if j's virtual move does correspond to a move in Γ, the payoffs after the virtual move may bear no relationship to the actual payoffs after that move in Γ. Agent i's subjective game may include virtual moves for i himself; i may believe that he will become aware of more moves (and may take active steps to try and learn about these moves).

To handle awareness of unawareness, we consider a generalization of the notion of augmented game. We continue to refer to the generalized notion as an augmented game, using "augmented game without awareness of unawareness" to refer to the special case we have focused on up to now. Formally, $\Gamma^+ = (N^+, M^+, H^+, P^+, f_c^+, \{\mathcal{I}_i^+ : i \in N^+\}, \{u_i^+ : i \in N^+\}, \{A_i^+ : i \in N^+\})$ is an *augmented game based on the (standard) finite extensive game* $\Gamma = (N, M, H, P, f_c, \{\mathcal{I}_i : i \in N\}, \{u_i : i \in N\})$ if it satisfies conditions A1–A3, A6–A10 and A12 of augmented games, and the following variants of A4 and A5:

A4′. If $P^+(h) \in N^+$, then $P^+(h) = P(\overline{h})$, and $M_h^+ \subseteq M_{\overline{h}} \cup (M^+ - M)$. Moreover, every history $h \cdot \langle m\rangle \in H^+$ with $m \in M_h^+ - M_{\overline{h}}$ is terminal. Intuitively, the moves in $M_h^+ - M_{\overline{h}}$ are virtual moves.

A5′. If $P^+(h) = c$ and $M_h^+ \cap M \neq \emptyset$, then $M_h^+ \subseteq M_{\overline{h}} \cup (M^+ - M)$. Moreover, if $m \in M_h^+ - M$ and $M_h^+ \cap M \neq \emptyset$, then $h \cdot \langle m\rangle$ is terminal. Intuitively, if $M_h^+ \cap M \neq \emptyset$, then all the moves in $M_h^+ - M$ are virtual moves.

A *game with awareness of unawareness based on* Γ is defined as a tuple $\Gamma^* = (\mathcal{G}, \Gamma^m, \mathcal{F})$ just as before. The modeler's extended game Γ^m satisfies the same conditions M1-M3 as before, and the mapping $\mathcal{F}$ satisfies C1 and C3–C10 and the following variant of C2:

C2′. If $h' \in H^h$ and $P^h(h') = j$, then (a) $A_j^h(h') \subseteq a$, (b) $(M_{\overline{h}'} \cap \{m : \overline{h}' \cdot \langle m \rangle \in a\}) \cup (M_{h'}^h - M_{\overline{h}'}) = M_{h'}^h$, and (c) if $\mathcal{F}(\Gamma^h, h') = (\Gamma', I')$, then for all $h'' \in I'$, we have $M_{h''}' \subseteq M_{h'}^h$.

Part (a) of C2′ is the same as the first part of C2. Part (b) implies that the set of moves available for player j at h' is the set of moves that player i is aware of that are available for j at $\overline{h}'$ in the underlying game together with some virtual moves. It is not hard to check that in games without awareness of unawareness, part (c) follows from A4, C1, and C2, so it does not need to be explicitly stated in C2. However, now that A4 has been weakened to A4′, we must mention it explicitly.

Note that Γ^m is an augmented game with no awareness of unawareness; there are no virtual moves, since the modeler is indeed aware of all possible moves (and knows it). We can now define local strategies, generalized strategy profiles, and generalized Nash equilibrium just as we did for games with awareness. The same technique as that used to show Corollary 3.1 can be used to prove the following.

THEOREM 4.1. *Every game with awareness of unawareness has a generalized Nash equilibrium.*

5. CONCLUSION

We have generalized the representation of games to take into account agents who may not be aware of all the moves or agents, but may be aware of their lack of awareness. As we show in the full paper, our representation is also flexible enough to deal with subjective games when there is lack of common knowledge about the game, even if awareness is not an issue. We have also shown how to define strategies and Nash equilibrium in such settings. These generalizations greatly increase the applicability of game-theoretic notions in multiagent systems. In large games involving many agents, agents will almost certainly not be aware of all agents and may well not be aware of all the moves that agents can make. Moreover, by giving awareness a more computational interpretation, we can provide a more realistic model of a game like chess.

There is clearly much more to be done to understand the role of awareness (and lack of awareness) in multiagent systems. We list some of the many issues here:

- In a Nash equilibrium of an extensive-form game, it may be the case that the move made at an information set is not necessarily a best response if that information set is not reached. There are refinements of Nash equilibrium that are arguably more appropriate for an extensive-form game [8]. Our representation of games with awareness (of unawareness) allows for relatively straightforward generalizations of such refinements of Nash equilibrium. However, there are subtleties involved in showing that generalized versions of these refinements always exist.

- We have analyzed situations where agents may be unaware of some moves in the underlying game and may be aware of their unawareness. Of course, there are other cases of interest where additional properties may hold. For example, consider a large geographically-dispersed game where agents interact only with nearby neighbors. In such a game, an agent may be unaware of exactly who is playing the game (although she may realize that there are other agents besides her neighbors, and even realize that the moves made by distant agents may have an indirect effect on her). To model such a situation, we may want to have virtual moves after which the game does not end, and to allow agents to be aware of subsequences of histories in the underlying game. We suspect that extensions of the ideas in this paper can deal with such situations, but we have not worked out the details.

- There has been a great deal of work on computing Nash equilibria. As we have shown, a generalized Nash equilibrium of a game with awareness is a Nash equilibrium of a standard game. However, this standard game can be rather large. Are there efficient computational techniques for computing generalized Nash equilibrium in interesting special cases?

- If there is little shared knowledge regarding the underlying game, the set $\mathcal{G}$ of augmented games can be quite large, or even infinite. Is it important to consider all the iterated levels of unawareness encoded in $\mathcal{G}$? Halpern and Moses [6] showed that, in analyzing coordinated attack, no finite level of knowledge suffices; common knowledge is needed for coordination. Stopping at any finite level has major implications. On the other hand, Weinstein and Yildiz [9] provide a condition under which the effect of players' kth order beliefs is exponentially decreasing in k. While we strongly suspect that there are games in which higher-order unawareness will be quite relevant, there may be conditions under which higher-order awareness becomes less important, and a simpler representation may suffice.

6. REFERENCES

[1] R. Fagin and J. Y. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34:39–76, 1988.

[2] Y. Feinberg. Subjective reasoning—games with unawareness. Technical Report Resarch Paper Series #1875, Stanford Graduate School of Business, 2004.

[3] Y. Feinberg. Games with incomplete awareness. Technical Report Resarch Paper Series #1894, Stanford Graduate School of Business, 2005.

[4] Y. Gal and A. Pfeffer. Reasoning about agents' beliefs and decision-making processes in games. Unpublished manuscript, 2005.

[5] J. Y. Halpern. On ambiguities in the interpretation of game trees. *Games and Economic Behavior*, 20:66–96, 1997.

[6] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.

[7] J. Harsanyi. Games with incomplete information played by 'Bayesian' players, parts I–III. *Management Science*, 14:159–182, 320–334, 486–502, 1968.

[8] M. J. Osborne and A. Rubinstein. *A Course in Game Theory.* MIT Press, Cambridge, Mass., 1994.

[9] J. Weinstein and M. Yildiz. Impact of higher-order uncertainty. Unpublished manuscript., 2003.

Learning Against Multiple Opponents

Thuc Vu, Rob Powers, Yoav Shoham
Stanford University
{thucvu, powers, shoham}@cs.stanford.edu

ABSTRACT

We address the problem of learning in repeated n-player (as opposed to 2-player) general-sum games, paying particular attention to the rarely addressed situation in which there are a mixture of agents of different types. We propose new criteria requiring that the agents employing a particular learning algorithm work together to achieve a joint best-response against a target class of opponents, while guaranteeing they each achieve at least their individual security-level payoff against any possible set of opponents. We then provide algorithms that provably meet these criteria for two target classes: stationary strategies and adaptive strategies with a bounded memory. We also demonstrate that the algorithm for stationary strategies outperforms existing algorithms in tests spanning a wide variety of repeated games with more than two players.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithm, Design, Theory

Keywords

Learning, Game Theory, Multi-agent Systems

1. INTRODUCTION

Recently there have been several proposals for criteria with which to evaluate learning algorithms in multi-agent environments and corresponding algorithms achieving these criteria [2, 6, 21, 20]. However, these approaches focus primarily on settings with two agents and do not adequately address the different situations arising from multiple opponents. While other research has focused on situations with more than two agents, it has for the most part either concentrated on games with common payoffs or assumed no cooperation is possible and focused on no-regret payoff guarantees for each agent. In this paper we will try to address how an agent should behave in general-sum games in which the opponents can be either cooperative or adversarial. Following the approach of [21], we propose new criteria focusing on payoff guarantees for the agent against various classes of opponents:

Targeted Group Optimality: *The payoffs achieved by all the agents using this algorithm are at least within ϵ of being Pareto-efficient over the set of possible outcomes given the actual strategies of any agents that are members of the target set.*

Safety: *Against any combination of opponents, the algorithm always receives at least within ϵ of its security value for the game.*

In the rest of this paper we will first start with a brief review of related work on learning in multi-agent systems, focusing on the limitations of current approaches. Building on this we detail our proposal for a set of desirable criteria for agents in n-player repeated games and confront some of the thorny issues that arise when playing against a mix of different opponent types during a single game. Next, we outline a novel algorithm achieving these criteria and show encouraging empirical results against a number of existing algorithms in a wide range of multi-agent environments. In the final section, we'll point out some of the successes and limitations of our approach and pose some open questions for future work in this domain.

Throughout this paper, we will focus our attention on the class of repeated games with average reward. In this setting the players repeatedly play a simultaneous move normal form game (the stage game), represented as a tuple, $G = (n, A, R_{1...n})$, where n is the number of players, $A = A_1 \times ... \times A_n$, where A_i is the set of actions for player i, and $R_i : A \rightarrow \Re$ is the reward function for agent i. We also use m to denote the maximum number of actions for an agent in G, and $|A|$ the number of stage-game outcomes in G. After each round, the agents accumulate their reward from the joint outcome and get to observe the previous actions of the other agents. Each agent is assumed to be trying to maximize its average reward for games with finite repetitions or the limit average for infinitely repeated games. For our purposes we assume that the full game structure and payoffs are known to all agents from the start of the game and that the payoffs in the game are bounded.

2. RELATED WORK

One of our main objectives in this paper is to develop a set of criteria for learning algorithms in repeated games with an

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

arbitrary number of players. In reviewing existing work, we start with work focusing explicitly on games with more than two players. Following this we consider proposals of criteria for learning algorithms and see how well they generalize to games with more than two players.

Many researchers have focused on the problem of coordinating multiple agents to achieve mutually beneficial outcomes, but have for the most part restricted their attention to team games [15, 14, 23, 4]. In a team game, all the agents get identical payoffs for each action, so the challenge is focused entirely on how the agents can independently coordinate on an optimal equilibrium. Additional related work has been carried out in the areas of multi-robot planning and multi-agent pursuit games, but most of these approaches assume explicit communication or sharing of information between the cooperative agents.

Bowling and Veloso[2] were among the first to propose specific requirements for effective learning in multi-agent systems with their two criteria of rationality and convergence:

Rationality: *If the other players' policies converge to stationary policies then the learning algorithm will converge to a stationary policy that is a best-response (in the stage game) to the other players' policies.*

Convergence: *The learner will necessarily converge to a stationary policy.*

While these criteria are suitably general to apply to games with any number of players, the WoLF algorithm they proposed is only guaranteed to achieve these criteria in two player games. Conitzer and Sandholm[6] then proposed a new algorithm that achieved the above criteria for arbitrary repeated games, but only when all the opponents were of the same type (either all stationary or all self-play).

Moreover, as we pointed out in [21], there are limitations to these criteria. The first limitation is that the property of convergence cannot be applied unconditionally, since one cannot ensure that a learning procedure converges against all possible opponents in finite time without sacrificing rationality. So implicit in that requirement is some restriction on the class of opponents. And indeed both [2] and [6] acknowledge this and choose to concentrate on the case of self-play, in which the opponents are identical to the agent in question. However, neither approach makes any guarantees about the behavior of the learning algorithm when there is a mixture of self-agents and stationary opponents.

The second limitation is that the requirement of convergence to a stationary strategy is particularly hard to justify. When combined with the requirement to play a best response to any stationary opponent, this requires the agents to converge to playing a Nash equilibrium of the stage game. While at first glance this may seem desirable, consider the game of Prisoner's Dilemma. Any algorithm satisfying the above criteria will be forced to Defect at each period in order to arrive at the unique Nash equilibrium. In the repeated game, however, two agents could instead use a strategy such as Tit-for-Tat, to achieve a much higher reward for each agent without providing the opponent with an incentive to deviate. (Tit-for-Tat starts by cooperating and thereafter repeats whatever action the opponent played last.)

Brafman and Tennenholtz addressed this problem directly in [3] and made a counter-proposal for how to consider equilibria in repeated games. They require that the learning algorithms form an Efficient Learning Equilibrium (ELE) in which any agent deviating from its algorithm will suffer a net loss of payoff within a polynomial number of stage games. They also propose an ELE algorithm based on the folk theorem that satisfies this requirement for 2-player repeated games in a perfect monitoring setting. However, once generalized to games with more than two players, the ELE algorithm requires a communication mechanism outside the game or is only applicable for a restricted number of games.

Game theory also addressed the issue of reasonable criteria for learning in multi-agent systems at numerous times with the proposals of universal consistency, no-regret learning, and the Bayes envelope dating back to at least [10] (see [8] for an overview of this history). There is a fundamental similarity in approach throughout, and we will take the approach of Fudenberg and Levine in [9] as being representative. They first proposed two criteria:

Safety: *The learning rule must guarantee at least the minimax payoff of the game.*

Consistency: *The learning rule must guarantee that it does at least as well as the best response (in the stage game) to the empirical distribution of play when playing against an opponent whose play is governed by independent draws from any fixed distribution.*

They then defined *universal consistency* as the requirement that a learning rule do at least as well as the best response to the empirical distribution of play regardless of the actual strategy the opponent is employing (this implies both safety and consistency) and propose an algorithm that achieves this requirement. Recently, these ideas have also been adopted by researchers in the artificial intelligence community (see [11], [12] and [25] for examples in both game theory and AI). In recent work [1], Bowling attempted to combine these criteria by proposing that an agent should both guarantee a no-regret payoff and achieve convergence in self-play. He then put forth GIGA-WoLF, a no-regret algorithm that provably achieves convergence in self-play for games with two players and two actions per player.

A limitation common to all these approaches is that the game theoretic basis they're derived from was initially focused on large-population games and therefore ignores the effect of the agent's play on the future play of the opponent. This can pose problems in smaller games. Let us again consider the game of Prisoner's Dilemma with a Tit-for-Tat opponent. The only universally consistent strategy would be to defect at every time step, ruling out the higher payoff achievable by cooperating. Clearly, a universally consistent (or no-regret) policy is not the best response in this richer strategy space.

In response to the existing work, we proposed a new set of criteria in [21]:

Targeted Optimality: *Against any member of the target set of opponents, the algorithm achieves within ϵ of the expected value of the best response to the actual opponent.*

Auto-Compatibility: *During self-play, the algorithm achieves at least within ϵ of the payoff of a Nash equilibrium that is not Pareto dominated by another Nash equilibrium.*

Safety: *Against any opponent, the algorithm always receives at least within ϵ of the security value for the game.*

In addition these requirements were required to hold with probability at least $1 - \delta$ after an initial polynomial period of time. Our first paper provided an algorithm that consid-

ered only stationary opponents, but a new algorithm meeting these criteria against memory-bounded opponents was presented in [20]. Unfortunately, neither of the algorithms can easily be generalized to situations with more than two agents. A critical component of the design of the two algorithms is a teaching component based on the Bully and Godfather algorithms proposed by Littman and Stone in [16]. This component relies on having a single opponent in order to calculate the best action or sequence of actions to play given that the opponent will play a best response.

3. NEW CRITERIA

In our work we seek to pull together the advances made in existing work and encourage the development of algorithms that can both cooperate with one another and achieve strong guarantees on payoff against a variety of opponents.

In the criteria of [21] we had three categories of opponents we cared about: members of the target class, identical (self-play) agents, and other (unconstrained) agents. When only considering games for two agents, these three categories are sufficient for all the scenarios. However, once we extend to games with more than two agents, we need to consider mixes of agent types. Since each agent can be any of the three types, there are seven different possible sets of opponent types. We divide those seven sets into two scenarios:

- Each of the opponents is either a self-agent or in the target class.
- At least one opponent is of the Unconstrained type.

We also need to consider the issues of coordination between the agents when selecting actions. Although other researchers may find different assumptions appropriate for particular settings, we have chosen to focus on the most pessimistic/conservative assumptions in which the self-agents have no communication mechanism outside the game. Thus they must choose independently if they are selecting actions according to a probability distribution.

For the first scenario, ideally the agents should achieve a "joint best response" against the given target class. This is clearly defined when there is only one self-agent. The agent just needs to adopt the best response to the joint play of the agents in the target class. However, it remains an interesting issue when there are at least two self-agents. The self-agents will need to distribute the payoffs between themselves since selfishly trying to maximize one's payoff will not help the agent in many cases. Each of the self-agents should also be able and willing to cooperate with other agents who are using the same or similar algorithms as long as this will help to increase its current payoff. Thus the self-agents should achieve a Pareto-optimal (PO) outcome among themselves, i.e. there is no other joint outcome that could provide a higher payoff for one self-agent without decreasing the payoff of some other self-agent given the strategies of the opponents in the target class.

The PO condition alone, however, is not sufficient. Each agent has a minimum payoff that it can guarantee by itself without the cooperation of any other agents. In the worst case when all other agents are trying to minimize its payoff, this minimum value is $V_{security}$, defined for agent i as $\max_{\pi_i \in \Pi_i} \min_{\pi_{-i} \in \Pi_{-i}} EV_i(\pi_i, \pi_{-i})$. In this formula, Π_i is the set of strategies for agent i, and Π_{-i} is the set of joint strategies for the other agents. Any strategy that achieve this value on expectation is called a security strategy. Note that if an agent has determined the strategies of the opponents in the target class, it can guarantee a new security value V_g which is at least $V_{security}$. Thus it is only rational for an agent to cooperate in a PO joint outcome if its payoff is at least V_g. Notice that the best response condition for one self-agent is a special case of this criterion, since the PO condition will guarantee that the agent is using an optimal strategy against the target class.

When there are opponents that are neither members of the target class nor self-agents, we instead require that each agent achieve at least V_g. Ideally, multiple self-agents could each exceed this value in some settings by cooperating with each other against the unconstrained agents.

Combining these two scenarios, we put forth the following new criteria:

Let n be the number of players in the game and m the maximum number of actions for a player. We require that for any choice of $\epsilon > 0$ and $\delta > 0$ there exist a τ, polynomial in $\frac{1}{\epsilon}, \frac{1}{\delta}, n$, and m, such that for any number of rounds $t > \tau$ the algorithm achieves the following payoff guarantees with probability at least $1 - \delta$:

Targeted Group Optimality: *When each of the agents in the game is either a self-agent or in the target class, the payoffs of all the self-agents are at least $V_g - \epsilon$ and within ϵ of a PO outcome, given the actual strategies of agents in the target class.*

Safety: *Against any set of opponents, the agent must achieve at least $V_g - \epsilon$.*

Note that the Target Group Optimality condition combines and generalizes the Targeted Optimality and Auto-Compatibility conditions from [21].

4. CORRSTRATEGY(S): AN ALGORITHM FOR STATIONARY OPPONENTS

Besides proposing the novel criteria, we also want to provide algorithms that provably achieve the criteria for different target sets. We first consider such an algorithm for stationary opponents which we call CorrStrategy(S).

4.1 Algorithm Description

CorrStrategy(S) is composed of four modules:

- **Learn Best Response**: Using observations about the opponents' play estimate and play a best-response strategy for the agent to the actual strategy of the opponents.
- **Coordinate**: Select a single, common deterministic joint strategy for all the self-play agents from among a set of Pareto-optimal possibilities.
- **Secure Value**: Play a strategy that ensures that the player receives at least the security value against any possible set of opponents.
- **Signal/Explore**: Observe the opponents' play and use explicit signaling to distinguish self-play agents from agents in the target class.

We show how these modules can be put together in Figure 1. The four bolded rectangles represent the modules. Note that in order to preserve clarity, we only show a detailed view of the "Coordinate" module, the most complex one. The agents following the framework will start with the "Signal" module and make transition between the modules based on

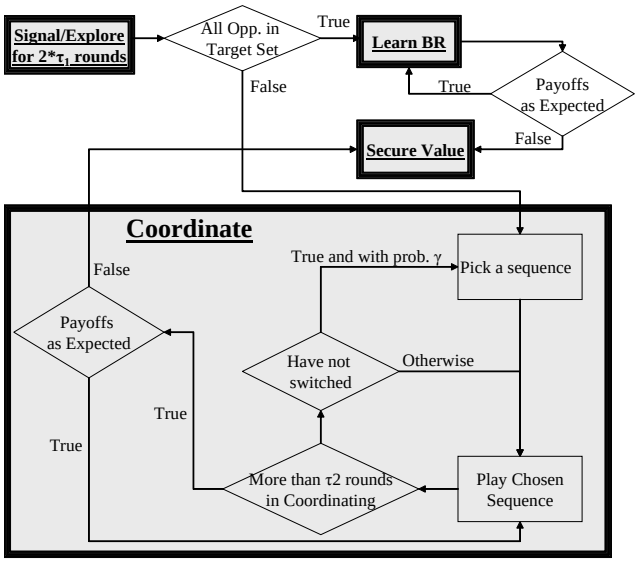


Figure 1: Flow of control for CorrStrategy(S).

the payoffs they receive and the observed behavior of the other agents in the environment.

It is necessary for the agents to distinguish between self-agents and members of the target class. Otherwise it would be possible for the agents to adopt best-responses assuming the other self-agents are stationary and play a Nash equilibrium with possibly non-PO payoffs. In general, it is not always trivial to distinguish between different types of opponents. One self-agent might appear stationary if the most rational action it needs to take always happens to be the same. This issue can be resolved easily if we allow the self-agents to explicitly signal each other in the "Signal" module.

Since the target class is stationary, the self-agents will signal each other by playing a pure strategy for τ_1 rounds and then switching to a different pure strategy for another τ_1 rounds. By the end of this block, the self-agents will be able to correctly partition all self-agents and stationary agents into two different sets with high probability. Agents whose distribution of actions during the last τ_1 rounds is within ϵ of the distribution from the full history are assumed to be stationary.

Each self-agent can now essentially reduce the current game to a smaller game by removing all stationary opponents and using the expected payoffs for each of the remaining outcomes instead. If there is only one remaining player in the sub-game, it transitions to the "Learn Best Response" module to find the best response to the stationary opponents. Using the sub-game, finding a BR strategy against stationary opponents is straightforward, since the agent can simply choose the action that gives the highest expected payoff. However, the agent still needs to keep monitoring its payoff in order to protect itself against an adversarial opponent that pretends to be stationary during the "Signal" phase and changes strategy later on. The agent can calculate its security value in the sub-game which is V_g. At any time, if its payoff drops below $V_g - \epsilon$, the agent will switch to the "Secure Value" module.

If there are multiple non-stationary agents left in the sub-game, the self-agents will make the transition to the "Coordinate" module instead. In this module, they will try to synchronize with each other in order to achieve the Targeted Group Optimality criterion. Each self-agent will first solve a linear program with size polynomial in the number of stage game outcomes, $|A|$, to find a PO outcome in the repeated game:

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{|A|} \sum_{j=1}^{n} \alpha_i * PO_j(O_i) \\ \text{subject to} & \forall j (V_g(j) - \sum_{i=1}^{|A|} \alpha_i * PO_j(O_i) \leq \epsilon) \\ & \sum_{i=1}^{|A|} \alpha_i = 1 \end{array}$$

In the equations above, $PO_j(O_i)$ is the payoff of agent j in the stage game outcome O_i and $V_g(j)$ is the guaranteed value of agent j. We denote this linear program LP^*.

The solution to LP^* is a distribution over joint outcomes in the stage game that maximizes the sum of the payoffs for the self-agents in the repeated game. In this distribution, α_i is the frequency with which the stage-game outcome O_i should be played by the self-agents to achieve a PO repeated game outcome. The solution always exists because if every self-agent, j, plays its security strategy they are all guaranteed to receive at least $V_g(j)$. Since the self-agents are not allowed to communicate outside the game, they can't mix over joint outcomes so will instead have to approximate this distribution by repeating a deterministic sequence S of joint actions. S will specify which action each agent needs to choose at each step. By using Hoeffding's inequality, it can be proved that self-agents can have an arbitrarily close approximation to the PO outcome in the repeated game with S of length polynomial in $m, n, \frac{1}{\epsilon}$, and $\frac{1}{\delta}$.

As there are possibly many such sequences S, the self-agents also need to coordinate with each other to converge to the same one. Each self-agent will pick one such sequence at the beginning of the "Coordinate" module and then with probability γ in each round will switch to a different sequence. Let C be the group of self-agents who have already switched. The algorithm ensures that all the agents in C are always jointly maximizing the sum of payoffs for the self-agents. Whenever an agent attempts to switch to a different sequence, i.e. joining C, it will pick one that preserves this property. Note that since each agent joins the group C when it switches, it will only actually switch strategies once if all the agents are self-agents.

Since the agents in C are approximating some distribution over joint outcomes by using deterministic sequences, the switching agent just needs to observe this distribution and then find a correspondent deterministic sequence. To achieve this, the switching agent can re-solve LP^* with additional constraints to guarantee that in the new solution of LP^*, the distributions over joint outcomes of the agents in C are still the same.

$$\forall o \in \prod_{j \in C} A_j : \sum_{i=1}^{|A|} f(O_i, o) * \alpha_i = \pi(o)$$

In the above equation, o is the joint action of agents in C, $\pi(o)$ is the observed distribution for o , and $f(O_i, o) = 1$ if agents in C play o in outcome O_i and $f(O_i, o) = 0$ otherwise. If no agents have switched sequences, the agent will simply pick a different sequence that also approximates the solution for the current LP^*.

At the end of the process, there are two possibilities that occur with high probability: either all the self-agents are playing the same deterministic sequence of joint actions or there exist unconstrained agents. In the first case, the payoff profile for the self-agents is at most ϵ away from the payoff profile in the PO outcome they are trying to approximate.

However, they still need to monitor their payoffs to avoid the case in which there are unconstrained agents that only pretend to cooperate with the self-agents. Once the payoff for any self-agent drops more than ϵ below their target payoff in LP^*, the agents will switch to the "Secure Value" module to prevent being further exploited. This will also handle the second case where the unconstrained agents prevented them from converging to a common sequence. Note that in the case the unconstrained agents fully cooperate with the self-agents in a PO outcome, the self-agents will implicitly achieve the Safety requirement, and therefore can safely treat the unconstrained ones as self-agents.

A self-agent uses the "Secure Value" module to guarantee its payoff is at least the guaranteed value of $V_g - \epsilon$ in the presence of unconstrained agents. The agent can easily achieve this by calculating and adopting the security strategy for the sub-game obtained by removing all stationary opponents. While using this module, the self-agent still needs to monitor the opponents that are assumed to be stationary. Otherwise, unconstrained agents could impersonate members of the target set at the beginning of the game and then lower the payoff of a self-agent by changing strategies, or more subtly, by using a correlated joint mixed-strategy that still makes them appear stationary. In both cases, any harmful variations can be easily detected with high probability by replacing each stationary agent's actual play with random draws from their observed distribution and detecting if the payoffs to any of the self-agents change by more than ϵ. If such an unconstrained agent is detected, a new V_g can be calculated and a new security strategy played.

THEOREM 1. *CorrStrategy(S) is guaranteed to satisfy the Targeted Group Optimality and Safety criteria for the target class of stationary opponents after a number of rounds polynomial in $n, m, \frac{1}{\epsilon}$ and $\frac{1}{\delta}$.*

PROOF. Since we only consider games with bounded payoffs, we can assume, without loss of generality, that all the payoffs are normalized to be between 0 and 1. The proof can be constructed naturally from the following lemmas. The proofs for the lemmas can be constructed using Hoeffding's inequality and are omitted due to space constraints:

LEMMA 1. *For any given $\delta_1 > 0$, $0.5 > \epsilon_1 > 0$, there exists τ_1 polynomial in $\frac{1}{\epsilon_1}, \log(\frac{1}{\delta_1}), \log n$ and $\log m$ such that if an agent uses a full action history of length at least $2\tau_1$, and a recent action history of length τ_1, all self-agents will correctly partition stationary and self-agents into two different sets and the observed action distribution for all stationary opponents will be within ϵ_1 of the true distribution with probability at least $1 - \delta_1$.*

LEMMA 2. *For any given $\delta_2 > 0$, $\epsilon_2 > 0$, there exists a deterministic sequence of joint actions, S, with length polynomial in $\frac{1}{\epsilon_2}, \log(\frac{1}{\delta_2}), \log n$, and m that can approximate within ϵ_2 the distribution over PO outcomes of any solution to LP^*. When there is no unconstrained agent, the difference between the payoffs achieved by the self-agents using S and a PO outcome will be at most $n * m * \epsilon_2$ with probability at least $1 - \delta_2$.*

LEMMA 3. *Within the "Coordinate" block, for any given $\delta_3 > 0$, $T > 0$, and $\gamma \leq 1 - (1 - \delta_3)^{\frac{1}{n^2 T}}$, if each cooperating player attempts to change its distribution of actions on each round with probability γ, there exists a τ_2 polynomial in $\frac{1}{\delta_3}, T$ and n such that with probability at least $1 - \delta_3$, after τ_2 rounds all the self-agents will change their sequence exactly once and no two self-agents will change within T rounds of each other.*

From Lemma 1, by the end of the "Signal" module (after $2\tau_1$ rounds), the self-agents have correctly partitioned stationary players and coop players into two different sets with probability at least $1 - \delta_1$.

Consider first the case in which there are no unconstrained agents. Within the "Coordinate" module, once an agent decides to pick a different sequence, it has to recalculate the optimal solution to LP^*. From Lemma 1, we know that the observed distribution of each stationary opponent is within ϵ_1 of its true distribution. Since the payoffs are bounded between 0 and 1 and there are at most m actions, the payoff for the self-agent can be reduced by at most $m * \epsilon_1$ for each of the up to n stationary opponents. Thus if we choose ϵ_1 to be $\frac{\epsilon}{2mn}$ and ϵ_2 in Lemma 2 to be $\frac{\epsilon}{2mn}$, once all the self-agents converge to the same sequence, the payoff for each agent can only be at most ϵ away from the optimal payoff. From Lemma 3, if we set $T \geq L$, we can choose a γ such that no two agents will switch sequence within a period of L. Thus, each self-agent will always correctly determine the distribution of the agents that have already switched sequences. All self-agents will switch sequence once within a polynomial number of rounds and therefore converge to one sequence. Combining all of these conditions we have that the self-agents will converge to a sequence with payoffs within ϵ_3 of a PO outcome of the repeated game with probability at least $1 - \delta_1 - \delta_2 - \delta_3$. If we then set $\delta_1, \delta_2, \delta_3$ to be $\frac{\delta}{3}$ we satisfy Target Group Optimality. With these values, τ_1 and τ_2 are clearly polynomial in $m, n, \frac{1}{\epsilon}$, and $\frac{1}{\delta}$. Moreover, since the losses of the payoffs over the initial period of length $\tau = 2 * \tau_1 + \tau_2$ are bounded by τ, by allowing another $\frac{2\tau}{\epsilon}$ additional rounds to pass, the average payoffs of the agents for the entire game will converge within ϵ of a PO outcome.

Let us now consider the case when there is at least one unconstrained agent. Notice that in all the modules the self-agents switch to the Security Value module whenever their payoffs fall below $V_g - \epsilon$. If an agent has correctly divided the stationary and unconstrained agents it can now be guaranteed a payoff at least $V_g - \epsilon$ with probability at least $1 - \delta$ after a period polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, m, and n by the Hoeffding inequality. If on the other hand an unconstrained agent is simulating a stationary opponent, it will not be able to drive the agent's value below $V_g - \epsilon$ without causing its play to vary from that of its assumed stationary distribution and thus revealing itself. The agents can then update their estimates of V_g and adopt a new security strategy where the unconstrained agent is categorized correctly. Thus they will always achieve the Safety criteria.

Therefore, there exists a τ polynomial in $m, n, \frac{1}{\epsilon}$, and $\frac{1}{\delta}$ such that after an initial experimentation period of length τ, CorrStrategy(S) will satisfy the two criteria in section 3. □

THEOREM 2. *The computational complexity of CorrStrategy(S) for playing one round of the repeated game is polynomial in $|A|$, the size of the game.*

PROOF. To find the worst case complexity for one iteration, we can consider separately the complexity for each module of the algorithm as presented in Figure 1:

- Within the "Signal" module, once all the stationary

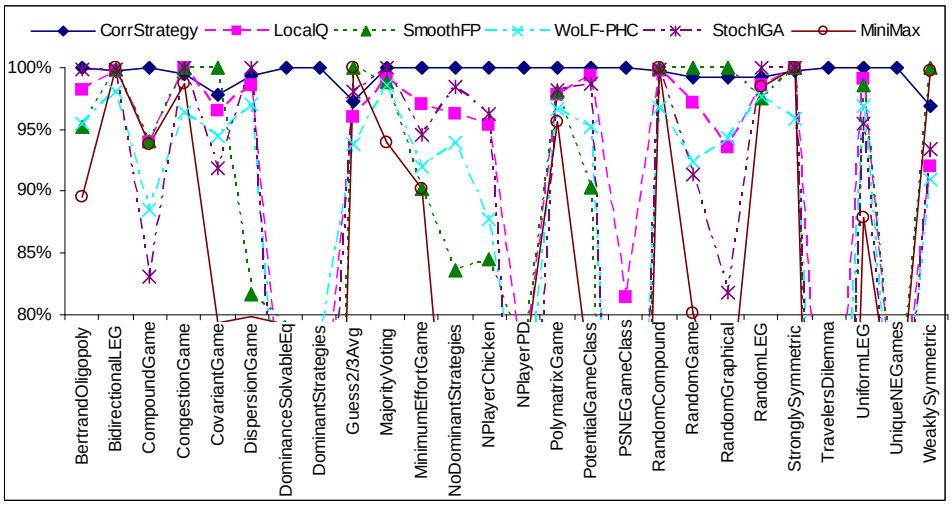


Figure 2: Percent of maximum value for 200K rounds averaged for each agent over all different pairs of opponents for selected games in GAMUT.

opponents are recognized, the self-agent can find the sub-game in $O(|A|)$ time.

- Within the "Learn Best Response" module, finding the best response in the sub-game is $O(m)$.
- Within the "Secure Value" module, solving the linear program for the security strategy can be done in time polynomial in $|A|$.
- Within the "Coordinate" module, in the worst step the agent has to switch to a different sequence. This process involves solving a linear program with the number of variables and constraints polynomial in the number of outcomes. Thus the complexity for the worst step within this module is also polynomial in $|A|$.

□

4.2 Experimental Results

Even though our algorithm has been proven to meet the proposed criterion, we want to demonstrate empirically that the algorithm performs well against a variety of opponents including those outside the target class. We will use the testing environment first described in [21] by testing against a number of existing approaches from the multi-agent learning literature over a wide variety of repeated games from GAMUT [18]. GAMUT is the result of a project to develop a comprehensive collection of game theoretic matrix games that have been described by researchers in either game theory or artificial intelligence. It contains generators for creating random instances of 34 individual base game classes as well as numerous additional variants and specialized parameter settings (more information and downloads are available at `gamut.stanford.edu`). The existing algorithms we tested against include Local Q-learning[24], a stochastic version of IGA[22], WoLF-PHC[2], JointQ-Max[5], and smooth fictitious play[9]. We also tested all the algorithms against random stationary strategies (Random), the security value strategy (MiniMax), and random strategies that condition their actions on the past outcome (Mem1).

Focusing our attention on settings with more than two players, our first test measured the average performance of a pair of players using the same algorithm playing against another pair of players using identical algorithms. In figure 2, we show the average payoffs of the five most successful algorithms as well as the performance of the MiniMax algorithm for a representative set of games in GAMUT. The payoffs have been normalized by dividing each algorithm's payoff by the best payoff achieved by any strategy for that game in order to make visual comparisons across games easier. We can see that CorrStrategy(S) performs well across all the games, achieving the highest or close to the highest payoff in nearly every game and unlike other algorithms, CorrStrategy(S) has no pitfalls in which its payoffs are significantly worse than the highest payoffs achieved. This is at least partly due to the fact that agents using CorrStrategy(S) can cooperate with agents using other algorithms as long as they each still achieve their security value.

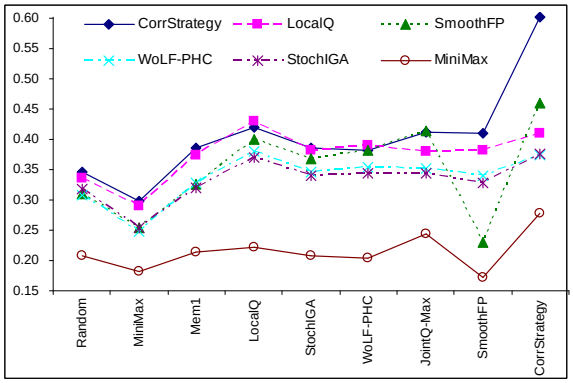


Figure 3: 2v2 By Opp: Value achieved against each pair of opponents averaged over all games.

In Figure 3, we show the results for the same algorithms averaged across games for each possible pair of opponents (note that all payoffs for each game are in the range $[-1, 1]$). We can see that CorrStrategy performs well against each of the different classes of opponents, but truly excels when playing against other CorrStrategy agents. Smooth fictitious play fares similarly well against several of the strategies, but fails to cooperate in self-play resulting in a payoff near that of the MiniMax algorithm. One of CorrStrategy's largest advantages lies in its ability to cooperate when pos-

sible while maintaining a security value guarantee against opponents outside its target set if cooperation fails.

To further investigate the contribution of this advantage to the strong results achieved by CorrStrategy(S), we ran two alternate experimental settings. In one setting we had each algorithm play by itself against a pair of algorithms using identical algorithms. In the second setting we instead let two versions of the algorithm being tested play against a single instance of an opponent algorithm. In Table 4.2 we can see the average results for these settings compared with the results in the four player setting shown earlier. As we would predict, CorrStrategy(S)'s advantage over the other algorithms increases in settings where it outnumbers its opponents, but its results remain competitive even when it is outnumbered in turn.

	1 Agent vs 2 Opps	2 vs 1	2 vs 2
CorrStrategy	0.45	0.49	0.40
LocalQ	0.42	0.44	0.37
SmoothFP	0.43	0.36	0.35
WoLF-PHC	0.38	0.38	0.34
StochIGA	0.39	0.37	0.33
MiniMax	0.29	0.20	0.21

Table 1: Summary of average payoff by setting.

5. CORRSTRATEGY(A): AN ALGORITHM FOR ADAPTIVE OPPONENTS

As encouraging as the theoretical and experimental results for CorrStrategy(S) may be, we still have yet to address one of our critiques of existing research, which was the tendency to only focus on stationary opponents. Although we addressed this concern explicitly in [20], our algorithm depended critically on the assumption that there was only a single opponent. We are aware of very little additional work to date that deals with adaptive opponents explicitly, although de Farias and Megiddo [7] address it in the design of their experts algorithm and the rational learning approach of Kalai and Lehrer [13] can in principle handle adaptive algorithms of arbitrary complexity as long as they are assigned positive probability in the prior.

In the remainder of this section, we'll outline a way to extend our CorrStrategy algorithm to deal with opponents whose play may be a function of the prior history of the game. We do this by expanding the target set against which we can guarantee a best-response. Note however that we still need to limit the capabilities of the opponents in some way. If we were to consider opponents whose future behavior could depend arbitrarily on the entire history of play, we would lose the ability to learn anything about them in a single repeated game, since we would only ever see a given history once and an opponent's past strategy may bear no relation to their future play.

We therefore adopt the model for bounded memory we used in [20] and assume a limit on each opponent's ability to condition on the history. This model requires that the opponents play a conditional strategy where their distribution over actions can only depend on the most recent k periods of past history, $F_i : o^{-1} \times ... \times o^{-k} \rightarrow \Delta A_i$, where o^{-t} is the outcome of the game t periods ago. Additionally, the opponents have a default past history they assume at the start of the game. Note that even this simple model allows us to capture many methods, such as Tit-for-Tat, that current approaches are unable to properly handle.

Taking the set of conditional strategies with history k as our new target set, we propose the extension CorrStrategy(A). CorrStrategy(A) shares the same basic algorithmic framework as CorrStrategy(S) with the following changes:

- For the "Signal/Explore" module, the self-agents play a uniform mixed strategy for a period long enough to gather sufficient observations to calculate each opponent's distribution of play for each possible history of length k. The self-agents then switch to a pure strategy that is distinct from the prior observed distribution of their play in order to signal each other. Opponents using a consistent distribution for each history are assumed to belong to the target class.

- For the "Learn Best Response" module, we calculate a best response against conditional strategies. This approach maintains counts of each opponent's actions after each history of length k, which it uses to solve an optimization problem over the deterministic conditional strategies the self-agents should themselves play in order to maximize the sum of their payoffs.[1]. Given sufficient observations, we can guarantee an ϵ-best response against any members of our target opponent set.

- Note that in order to use this new best-response function in CorrStrategy(A), we need to insure that the algorithm observes each length k history a sufficient number of times. This will be satisfied as long as the initial exploration phase continues for a length of time exponential in k. This exponential exploration period is unavoidable since we need to consider the possibility of opponents that only play a desirable action distribution for a single one of the exponential possible histories.

- For the "Coordinate" block, the optimization problem needs to take into account all possible sequences of actions for the adaptive opponents in the target set.

THEOREM 3. *CorrStrategy(A) is guaranteed to satisfy the Targeted Group Optimality and Safety criteria for the target class of conditional strategies with bounded memory k.*

The proof of Theorem 3 follows the same framework as the proof of Theorem 1 but we need to take into account a much longer initial period for signaling and observing opponent's distributions and also a longer sequence S to approximate the PO outcome. The initial experimentation period τ required will unfortunately now depend on m^k and $(\frac{1}{\lambda})^{(m^{nk})}$, where λ is the minimum probability the opponent assigns to any action ($\lambda = 1$ for opponents that condition only on the coop player's actions). Note that our worst case time complexity also grows similarly as we may now need to solve an optimization problem with up to m^k variables, although both of these bounds (computational complexity and amount of training) are based on extremely pessimistic

[1]For conditional strategies that only depend on the self-agents' actions, we can simply calculate the cycle of joint agent actions with the highest expected reward out of all possible unique action sequences (those that don't contain a length k repeated subsequence)

assumptions and are likely to prove tractable in practice for larger games with small values of k.

6. CONCLUSIONS AND FUTURE WORK

Even though there have been several recent proposals for different criteria to evaluate learning algorithms in multi-agent systems, little attention has been given to two important scenarios: learning in games with more than two players, and learning against adaptive opponents. We have addressed both scenarios by proposing a new set of criteria for learning in games with any number of players that takes a target class as parameter, allowing the designer to choose a class of opponents of interest. This set of criteria encourages learning algorithms to allow cooperation between agents while still guaranteeing a basic security guarantee for each agent. We then designed two algorithms that can achieve the criteria against two target classes: stationary strategies and adaptive strategies with bounded memories. Moreover, our implementation of the algorithm for stationary strategies outperforms a wide range of opponents across the games of a comprehensive test-bed of repeated games.

One interesting problem we will continue to investigate is how to improve the guaranteed payoff for self-agents against adversaries outside their target class. In many settings the agents should be able to work together in order to reduce the adversarial effects of their opponents and guarantee payoffs above their individual security values. In addition, we are continuing to extend our approach to other models of adaptive opponents. A common approach used in the literature on bounded rationality [17, 19] is to assume the players can be modelled by finite automata with k states. It should prove relatively straightforward, although computationally demanding, to extend our CorrStrategy(A) algorithm to handle deterministic automata by again replacing the best response function and optimization functions. Automata where the state transitions and action selection are non-deterministic pose a more challenging problem.

We are also looking at several ways to expand the set of environments these algorithms can be employed within. Of particular concern is looking for ways to weaken the requirement of full prior knowledge about the payoffs of the game. The major challenge seems to lie in creating the capability to cooperate without knowing or being able to observe the space of payoffs available to the other players. An additional area for further consideration would focus on games in which the agents have only partial observability of the previous actions of the other players.

7. ACKNOWLEDGMENTS

This work was supported by DARPA Grant HR0011-05-1 and by NSF Grant IIS-0205633.

8. REFERENCES

[1] M. Bowling. Convergence and no-regret in multiagent learning. In *Advances in Neural Information Processing Systems 17*. MIT Press, 2005.

[2] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002

[3] R. Brafman and M. Tennenholtz. Efficient learning equilibrium. In *Advances in Neural Information Processing Systems 15*. MIT Press, 2002.

[4] G. Chalkiadakis and C. Boutilier. Coordination in multiagent reinforcement learning: A bayesian approach. In *The Second International Joint Conference on Autonomous Agents & Multiagent Systems*, 2003.

[5] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 746–752, 1998.

[6] V. Conitzer and T. Sandholm. Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In *Proceedings of the 20th International Conference on Machine Learning*, pages 83–90, 2003.

[7] D. P. de Farias and N. Megiddo. How to combine expert (or novice) advice when actions impact the environment. In *Advances in Neural Information Processing Systems 16*, 2004.

[8] D. Foster and R. Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29:7–36, 1999.

[9] D. Fudenberg and D. Levine. Universal consistency and cautious fictitious play. *Journal of Economic Dynamics and Control*, 19:1065–1089, 1995.

[10] J. Hannan. Approximation to bayes risk in repeated plays. *Contributions to the Theory of Games*, 3:97–139, 1957.

[11] S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68:1127–1150, 2000.

[12] A. Jafari, A. Greenwald, D. Gondek, and G. Ercal. On no-regret learning, fictitious play, and nash equilibrium. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 226–223, 2001.

[13] E. Kalai and E. Lehrer. Rational learning leads to nash equilibrium. *Econometrica*, 61(5):1019–1045, 1993.

[14] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 326–331, 2002.

[15] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the 17th International Conference on Machine Learning*, pages 535–542. Morgan Kaufman, 2000.

[16] M. Littman and P. Stone. Implicit negotiation in repeated games. In *Proceedings of The Eighth International Workshop on Agent Theories, Architectures, and Languages*, pages 393–404, 2001.

[17] A. Neyman. Bounded complexity justifies cooperation in finitely repeated prisoner's dilemma. *Economic Letters*, pages 227–229, 1985.

[18] E. Nudelman, J. Wortman, K. Leyton-Brown, and Y. Shoham. Run the gamut: A comprehensive approach to evaluating game-theorectic algorithms. *The Third International Joint Conference on Autonomous Agents & Multiagent Systems*, 2004.

[19] C. H. Papadimitriou and M. Yannakakis. On complexity as bounded rationality. In *STOC-94*, pages 726–733, 1994.

[20] R. Powers and Y. Shoham. Learning against opponents with bounded memory. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 2005.

[21] R. Powers and Y. Shoham. New criteria and a new algorithm for learning in multi-agent systems. In *Advances in Neural Information Processing Systems 17*, 2005.

[22] S. Singh, M. Kearns, and Y. Mansour. Nash convergence of gradient dynamics in general-sum games. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 41–48, Morgan Kaufman, 2000.

[23] X. Wang and T. Sandholm. Reinforcement learning to play an optimal nash equilibrium in team markov games. In *Advances in Neural Information Processing Systems 15*,'02.

[24] C. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3/4):279–292, May 1992.

[25] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, 2003.

Selecting Informative Actions Improves Cooperative Multiagent Learning

Liviu Panait
George Mason University
4400 University Drive MSN 4A5
Fairfax, VA 22030, USA
lpanait@cs.gmu.edu

Sean Luke
George Mason University
4400 University Drive MSN 4A5
Fairfax, VA 22030, USA
sean@cs.gmu.edu

ABSTRACT

In concurrent cooperative multiagent learning, each agent simultaneously learns to improve the overall performance of the team, with no direct control over the actions chosen by its teammates. An agent's action selection directly influences the rewards received by all the agents, resulting in a co-adaptation among the concurrent learning processes. Co-adaptation can drive the team towards suboptimal solutions because agents tend to select those actions that are rewarded better, without any consideration for how such actions may affect the search of their teammates. We argue that to counter this tendency, agents should also prefer actions that inform their teammates about the structure of the joint search space in order to help them choose from among various action options. We analyze this approach in a cooperative coevolutionary framework, and we propose a new algorithm, iCCEA, that highlights the advantages of selecting informative actions. We show that iCCEA generally outperforms other cooperative coevolution algorithms on our test problems.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Experimentation

Keywords

Multiagent Learning, Cooperation, Coordination, Cooperative Coevolution

1. INTRODUCTION

Multiagent learning is challenging because the problem dynamics are often complex and fraught with local optima. Of particular interest to us is cooperative multiagent learning, where multiple agents learn to work together as a team to accomplish common goals [16]. More specifically, we are interested in *concurrent learning*, where each agent performs its own learning and has little or no control over the other agents' selection of actions.

Multiagent learning is problematic for existing machine learning techniques because the concurrent learning processes are not independent. Consider an agent that observes an environment containing other agents and that tries to improve its performance. This leads to a modification in its behavior, which is then sensed by the other agents, who then change their behaviors in order to improve their performance as well. This "moves the goalpost" on the original agent: its newly-learned behavior may no longer be appropriate. Thus as the agents co-adapt to one another, the environment is essentially changing beneath the agents' feet. Moreover, the agent itself contributes directly to how the landscape changes. Learning in the face of this dynamic is not easy: such co-adaptation can result in cyclical or chaotic adaptive behavior, and may gravitate towards suboptimal solutions.

Most cooperative multiagent learning algorithms assume the agents are rational: each agent searches for actions that fare well when used in combination with the actions currently favored by its teammates. This "best-response" approach usually results in the learners converging to Nash equilibria. Such "rational" convergence to equilibria may well be movement away from globally *team-optimal* solutions [13, 23]. To counter this, we argue that agents must *also* explore actions that inform their teammates about the structure of the space of rewards for joint actions. For example, if an agent identifies an action that helps other agents rank their available actions better, the agent should explore that action to help guide the teammates' learning processes.

Though we believe it to be general, we will demonstrate the application of this approach to a particular multiagent learning method of interest to us, namely *cooperative coevolution* [10, 22]. Coevolution is the use of evolutionary computation learning techniques in a multiagent setting. Ordinarily, evolutionary computation employs only a single learner to discover a global solution to an optimization problem: the learner first creates an initial pool of randomly-generated candidate solutions (a "population" of "individuals"), then assesses their quality ("fitness") independently of one another, then forms a new population of individuals through iteratively selecting, copying, and modifying ("breeding") individuals from the previous population with an emphasis on the fitter members of that previous population. The new population replaces the old one, and this cycle of fitness assessment, breeding, and population replacement continues until a sufficiently fit individual is discovered or until resources have expired. Each iteration of this cycle is known as a "generation".

Cooperative coevolutionary algorithms (CCEAs), in the form we will discuss here, use not one but multiple populations, each involved in its own separate learning cycle of fitness assessment, population formation, and population replacement. However, indi-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

viduals in a given population are no longer assessed independently, but rather in the context of individuals chosen from the other populations. Each population represents a sub-component of a full solution to the problem, and as part of its fitness assessment, an individual in a given population may be evaluated only by combining it with one individual from each of the other populations to form a complete solution. It is in this fashion that coevolution involves multiple learners (each of the populations' evolutionary search procedures) whose learning trajectories are intertwined (via joint fitness evaluation), and so coevolution must deal with the same coadaptation challenges as other multiagent learning methods.

A natural approach to applying CCEAs to cooperative multiagent learning is to assign one population to each of the learning agents in the team. Each individual in the population represents a potential behavior for the agent, and so from now on, for consistency, we will refer to actions rather than individuals. An action may be as simple as a single action in trivial environments, or as complex as policies involving internal states and memory for real-world problems. Thus each population represents a finite sample from an infinite space of possible actions. As the team reward permits only the evaluation of joint actions, an action in one agent's population may be evaluated when combined with actions from the other agents' current populations. Multiple such combinations are generally used.

Section 2 highlights related cooperative multiagent learning algorithms. Following, Section 3 introduces a novel learning algorithm where agents pay special attention to informative actions. We compare it against other cooperative multiagent learning techniques in Section 4. The paper concludes with a brief discussion of our findings, accompanied by directions for future work.

2. RELATED WORK

Two learning algorithms that are guaranteed to find the globally optimal joint action in a stateless environment are proposed in [3]. Both algorithms have two phases: agents first explore the entire space of joint actions (either deterministically or randomly); this is followed by a greedy selection of only the action (one per agent) that returned the highest reward. Both algorithms find the global optimum in polynomial time in the number of actions for each agent. However, scaling these algorithms to environments with states or with possibly infinite numbers of actions per agent may be problematic.

Instead of choosing actions deterministically or randomly, Claus and Boutilier [7] argue that agents should be more optimistic about their teammates: an agent should not prefer actions that do well in the context of the actions currently preferred by its teammates, but rather the agent should prefer actions that do well in the context of better actions that its teammates might learn. The application of this heuristic assumption results in additional multiagent reinforcement learning algorithms for stateless environments, such as the ones in [12, 11]. Research on scaling these algorithms from simple coordination games to more complex domains is reported in [2, 9, 15, 24].

The cooperative coevolution literature has followed a similar path. The properties of cooperative coevolutionary algorithms are analyzed in [5, 6, 26]; results of such experiments indicate that assessing the fitness of an action based on the maximum of multiple joint rewards works better than if it were based on the minimum or on the average. Recent work has analyzed the conditions under which coevolutionary systems gravitate towards suboptimal solutions [25], has provided a visual illustration of the basins of attraction for simple multiagent domains [20], and has proposed a biased CCEA that is more likely to find the global optimum [18].

Bucci and Pollack [4] apply recent advances from *competitive* coevolution research to improve CCEAs, resulting in the pCCEA algorithm. The authors argue that the aggregation of multiple joint rewards to compute the fitness of an action may result in loss of useful information. Instead, pCCEA uses all joint rewards to compute the set of actions in each population that are Pareto non-dominated: given two actions a_1 and a_2 for one agent, a_1 dominates a_2 if and only if (1) for any action b for the other agent, a_1 receives higher or equal reward when joined with b than a_2 does, and (2) there exists an action c for the other agent such that a_1 receives a strictly higher reward when joined with c than a_2 does. This set of non-dominated actions, termed an *archive*, is automatically copied to the next generation to help evaluate the new population of actions. After the evaluation is completed for that generation, a new archive is computed for each population. Our experiments in Section 4 indicate that pCCEA's archive tends to converge to the Pareto frontier, which unfortunately may be infinite in even simple cooperative multiagent domains.

3. THE iCCEA ALGORITHM

When multiple agents learn concurrently, each of them is afforded only a partial glimpse at the overall search space. Specifically, each agent may weight its actions using only a projection of the entire space, a projection that is influenced by the actions currently chosen by the other agents according to their own learning processes. Differences among such projections are illustrated in [17]: projections at early stages of learning may provide more information about the search space, because the agents' actions are more randomly distributed and so sample the joint space better. As the agents start to converge, the projections may become skewed, sometimes losing all information about the globally optimal solutions. As each agent's choice of actions influences the projections searched by the other agents, we may view each of the multiagent learning algorithms in Section 2 as recipes for agents to influence each other's learning processes.

We argue that the team of learning agents may benefit if each agent is concerned about the projection of the search space that it provides to its teammates via the actions it explores. In other words, agents should not necessarily explore only their most promising actions, but also those actions that provide the other agents with accurate projections of the joint search space. We propose a coevolutionary algorithm, iCCEA ("i" standard for "informative"), to illustrate the advantages of such an approach.

For simplicity, we present the pseudocode for the algorithm using only two agents (and hence two populations), although it may be extended to arbitrary numbers of agents. Given an agent, the other agent is referred to as its *teammate*. We will assume that all populations have equal size *PopSize*, though the algorithm can easily be extended to allow for different sizes. We define $Reward_p(i,a)$ as the reward received when an agent (whose actions are represented in population p) selects action i and its teammate selects action a.

iCCEA follows the standard architecture of a generational cooperative coevolutionary algorithm [22]. As is the case for other CCEAs, iCCEA assumes that an agent can perceive the actions chosen by its teammates. In iCCEA, as in pCCEA, each agent maintains a population of actions, a subset of which is defined as an *archive* responsible for ensuring that some actions exist primarily to keep the teammate's projection well informed. iCCEA agents learn concurrently, meaning, the populations advance through their generation cycles together, rather than one population advancing, then the other.

The evaluation process tests actions by combining them with actions from the teammate's archive, plus possibly some additional actions in the teammate's population. In addition to computing an action's fitness, the evaluation process stores the joint reward of any pair of actions that are evaluated together; this information is later used to update the archives of each agent.

Each generation, every action in a population is first evaluated by testing it in combination with every action from the teammate's archive. As the very first generation's populations have not built an archive yet, their entire population is used as an archive (an expensive process but one which ensures a thorough exploration of the joint space and good bootstrapping for the archive in future generations). If the maximum size of the archives is less than $MaxEvals$, the actions are also tested with enough randomly-chosen actions from the teammate's population (in the archive or not) to provide at least $MaxEvals$ tests per action. $MaxEvals$ establishes a threshold for the number of evaluations desired per each action: if the archive is too small, additional actions are sampled at random from the other population to provide additional assessments.

The fitness of action i is then set to maximum of F_i^j over all actions j in the teammate's population (F_i^j represents the fitness of action i when partnered with action j from the other population, and it equals $-\infty$ if i was not tested in combination with j). If the archive size is 1, this reduces to a common evaluation approach for CCEAs [22, 26]: the evaluation is equivalent to using the best action (from the previous generation) plus some actions chosen at random from the other population. The pseudocode for the evaluation process is:

iCCEA-Evaluation
Parameters
  $MaxEvals$: maximum evaluations per action
Initial Settings
  For each population p
    $ArchiveSize_p = PopSize$
Evaluation phase (at each generation)
  For each population p
    $p' =$ other population than p
    For each action i in p
      For each action j in p'
        $F_i^j = -\infty$
    For each action i in p
      For each action a in $Archive_{p'}$
        $F_i^a = Reward_p(i,a)$
        $F_a^i = Reward_{p'}(a,i)$
  $MaxArchive = \max_p ArchiveSize_p$
  Repeat for $\max(0, MaxEvals - MaxArchive)$ times
    For each population p
      Shuffle p
    For each index i in $1..PopSize$
      a_1 = action with index i in one population
      b_1 = action with index i in the other population
      $F_{a_1}^{b_1} = Reward_1(a_1, b_1)$
      $F_{b_1}^{a_1} = Reward_2(b_1, a_1)$
  For each population p
    $p' =$ other population than p
    For each action i in p
      $Fitness(i) = \max_{j \in p'} F_i^j$

We next assemble the archive through a process called archive selection, which is intended to select those actions which revealed features of the projected joint space useful to the other teammate. Specifically, we would like to select as an archive a minimal set of actions from each agent's population such that when assessing the fitness of actions in a given population, testing them against the full teammate's population would not change the rank ordering of their fitnesses beyond just testing against the teammate's archive. The hope is that this archive would provide an accurate evaluation and ranking of the teammate's actions in the next generation as well. Rank order is all that matters because the evolutionary selection procedure we use later compares individuals not on raw fitness, but solely on their relative rank order in the population.

We'd like the archive set to be as small as possible, because as each action in an agent's population is tested in combination with *every* action from the other agent's archive, large archives imply a prohibitive $O(n^2)$ evaluation cost. Therefore we add actions to the archive only if they cause actions in the other population to improve significantly enough so as to effect the ranking — causing actions to *worsen* is not considered helpful information. Of the various actions which change this ranking, we will select the ones which do so by raising fitnesses to the highest levels.

The archive selection process starts from the empty set and proceeds iteratively. For each action i not in an agent's archive, and for each action x in its teammate's population, we compute the fitness of x if evaluated in combination with all actions in the current archive, that is, $M1Fit_x = \max_{j \in Archive_p} F_x^j$. We also compute the fitness of x if i were added to the current archive, $M2Fit_x^i = \max_{j \in Archive_p \cup \{i\}} F_x^j$. Note that $M2Fit_x^i \geq M1Fit_x$. Our first criterion for adding i to the archive (it improves upon the current ordering of the teammate's population) translates into finding two actions x and y such that adding i to the archive would change their ranking relative to one another ($M1Fit_x \leq M1Fit_y$ and $M2Fit_x^i > M2Fit_y^i$, or equivalently $M3Fit_{x,y}^i \neq -\infty$ for a simpler shorter notation). Of all actions i that meet this first criterion, we greedily prefer the one that changed the ranking by raising the fitness of a teammate's action to the highest level. Note that the first action to be selected for the archive is always the action with the highest fitness. The pseudocode is:

iCCEA-Archive-Selection
Parameters
  $MaxArchiveSize$: maximum archive size
Archive Selection (at each generation)
  For each population p
    $p' =$ other population than p
    $Archive_p = \emptyset$
    While $Size(Archive_p) \leq MaxArchiveSize$
      For each action x in p'
        $M1Fit_x = \max_{j \in Archive_p} F_x^j$
        For each action i in $p - Archive_p$
          $M2Fit_x^i = \max_{j \in Archive_p \cup \{i\}} F_x^j$
          For each action y in p'

$$M3Fit_{x,y}^i = \begin{cases} M2Fit_x^i & \text{if } M1Fit_x \leq M1Fit_y \text{ and } M2Fit_x^i > M2Fit_y^i \\ -\infty & \text{otherwise} \end{cases}$$

      For each action i in $p - Archive_p$
        $MaxFit_i = \max_{x \in p'} \max_{y \in p'} M3Fit_{x,y}^i$
      Select $a =$ argmax$_i MaxFit_i$
      If $MaxFit_a = -\infty$
        Break from while loop
      Add a to $Archive_p$
    End while

The breeding and population reassembly phase of iCCEA proceeds similarly to the one in pCCEA: the archive members are selected from the old population and are copied directly into the new population, and remainder of the new population is filled with children bred using standard evolutionary computation methods[1] applied to the old population (including the old archive). The whole previous population (including the archive) competes for breeding. The pseudocode is straightforward:

iCCEA-Breeding
Breeding phase (at each generation)
  For each population
    Select its new archive with iCCEA-Archive-Selection
    Copy the archive into the new population
    Fill the remainder of the new population using standard EC breeding with a rank-based selection procedure (such as tournament selection)

4. EXPERIMENTS

In this section, we investigate the behavior of four different coevolutionary algorithms, and we compare them in terms of performance and in terms of the number of evaluations they require to achieve that performance. The first algorithm is the pCCEA algorithm introduced in [4]. Second is the cCCEA algorithm, which is a traditional CCEA algorithm that evaluates the fitness of an action as the maximum reward it receives when in combination with any of the actions in the teammate's population. cCCEA is guaranteed to converge to the global optimum if the population size is sufficiently large [18]. Third is the iCCEA algorithm proposed in Section 3. Fourth, rCCEA evaluates the fitness of an action as the maximum when partnered with six actions from the teammate's population: five chosen at random, plus the fittest action from the teammate's *previous-generation* population. We included this algorithm in the comparison because it uses a small fixed number of evaluations in each generation, such as is preferable for applications of multiagent learning to real problems.

We will test these algorithms using a class of problem domains called the *maximum of two quadratics* (or MTQ). These problems include a global optimum and a local suboptimum, where the suboptimum covers a much wider range of the search space and is thus difficult to escape. The problems have been used before by [4, 19].

We will assume that each action is a real-valued number from 0 to 1 inclusive. This defines a metric space for actions: in some sense action 0.5 is more similar to action 0.6 than action 0.9 is. While other techniques search for optima among sets of actions that have no "distance" relation among them, EC methods assume a distance relation: when breeding an action to form a new one, they will generally make more small (distance) changes than large changes.

The joint reward function for the MTQ class is defined as:

$$\text{MTQ}(x,y) \leftarrow \max \begin{cases} H_1 * (1 - \frac{16*(x-X_1)^2}{S_1} - \frac{16*(y-Y_1)^2}{S_1}) \\ H_2 * (1 - \frac{16*(x-X_2)^2}{S_2} - \frac{16*(y-Y_2)^2}{S_2}) \end{cases}$$

where x and y may take values (actions) ranging between 0 and 1. Different settings for H_1, H_2, X_1, Y_1, X_2, Y_2, S_1, and S_2 affect the difficulty of the problem domain in one of the following aspects. H_1 and H_2 affect the heights of the two peaks: higher peaks may increase the chance that the algorithm converges there. S_1 and S_2 affect the area that the two peaks cover: a higher value for one of them results in a wider coverage of the specific peak. This makes it more probable that the coevolutionary search algorithm will converge to this peak, even though it may be suboptimal. Different values for X_1, Y_1, X_2, and Y_2 result in changes in the locations of the centers of the two quadratics, which also affect the relatedness of the two peaks: similar values of the x or y coordinates for the two centers imply higher overlaps of the projections along one or both axes (the projections of the joint action space for one or both agents may retain more information about the globally optimal solution even if the other agent's population starts to converge to the suboptimal solution). In these experiments, we set $S_1 = \frac{16}{10}$, $X_1 = \frac{3}{4}$, $Y_1 = \frac{3}{4}$, $H_2 = 150$, $S_2 = \frac{1}{32}$, $X_2 = \frac{1}{4}$, $Y_2 = \frac{1}{4}$; H_1 was varied across experiments, but it was always less than 125.

MTQ using such settings is fairly difficult to optimize: the probability that a random sample has a function value within 0.01 of the optimal solution (which has a value of 150) can be computed as $\pi * \left(1 - \frac{149.99}{150}\right) * \frac{S_2}{16} = 0.0000004090615$. Given 51200 random samples (approximately the number of action evaluations performed during a typical evolutionary run), the probability that one of them exceeds a function value of 149.99 is $1 - (1 - 0.0000004090615)^{51200} = 0.02072615$.

As we will see in the next sections, the median of the results for the proposed iCCEA method is significantly higher than 149.99, which implies that iCCEA finds better approximations of the global optimum in more than 50% of the runs. This shows that iCCEA significantly outperforms random search in this domain, and implicitly the algorithms proposed in [3].

To further increase the difficulty of the problem domains with respect to the algorithms we analyze, we created a second class of problem domains, SMTQ, which is defined as:

$$\text{SMTQ}(x,y) \leftarrow \max \begin{cases} H_1 * (1 - \frac{16*(x_1^r-X_1)^2}{S_1} - \frac{12*(y_1^r-Y_1)^2}{S_1}) \\ H_2 * (1 - \frac{16*(x_2^r-X_2)^2}{S_2} - \frac{12*(y_2^r-Y_2)^2}{S_2}) \end{cases}$$

where x_1^r, y_1^r, x_2^r, and y_2^r are the original x and y values (which ranged between 0 and 1) rotated around the centers of the two peaks by $\frac{\pi}{4}$:

$$x_1^r = (x - X_1) * \cos\frac{\pi}{4} + (y - Y_1) * \sin\frac{\pi}{4} + X_1$$

$$y_1^r = (x - X_1) * \cos\frac{\pi}{4} - (y - Y_1) * \sin\frac{\pi}{4} + Y_1$$

$$x_2^r = (x - X_2) * \cos\frac{\pi}{4} + (y - Y_2) * \sin\frac{\pi}{4} + X_2$$

$$y_2^r = (x - X_2) * \cos\frac{\pi}{4} - (y - Y_2) * \sin\frac{\pi}{4} + Y_2$$

Observe that the two peaks have ellipsoid shapes aligned diagonally with the axes, as opposed to circular shapes in the MTQ problem domains. The two Nash equilibria from the MTQ class have now become an infinity of Nash equilibria in the SMTQ class. This creates an additional difficulty for the coevolutionary search. We used the same values for H_2, X_1, Y_1, X_2, Y_2, S_1, and S_2 as for the MTQ class.

The experiments used the ECJ package [14]. Each population contained 32 actions. cCCEA and rCCEA used elitism of size 1, meaning that the fittest individual in each population in the previous generation is automatically copied into the next generation's population. The entire archive survived automatically from one generation to the next for iCCEA and pCCEA. Unless stated otherwise, iCCEA used $MaxEvals = 5$ and $MaxArchiveSize = \infty$. The

[1] [1, 8] are good references for standard evolutionary algorithms.

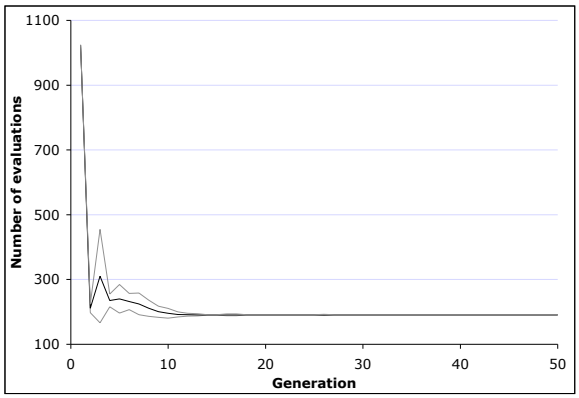


Figure 1: Average number of evaluations for the iCCEA algorithm in the MTQ domain instance with $H_1 = 50$

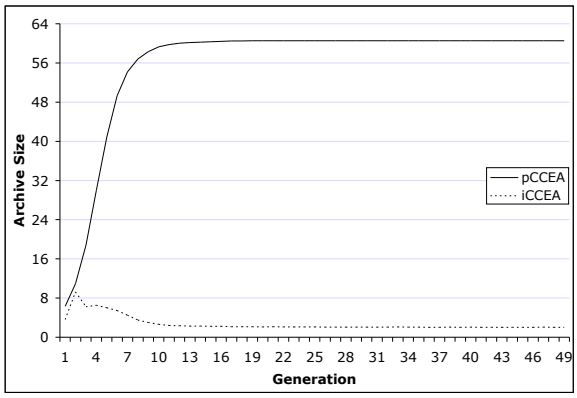


Figure 2: Average archive size for the iCCEA and the pCCEA algorithms in the MTQ domain instance with $H_1 = 50$

Table 1: Results of the four methods in the MTQ domain instance with $H_1 = 50$

Method	First Quartile	Median	Third Quartile	Average # Evals
pCCEA	148.62776	149.74876	149.94931	51200
cCCEA	149.99971	149.99995	149.99998	51200
iCCEA	149.99990	149.99997	149.99998	10676.5
rCCEA	50	50	149.99998	19200

Table 2: Results of the four methods in the SMTQ domain instance with $H_1 = 50$

Method	First Quartile	Median	Third Quartile	Average # Evals
pCCEA	133.21958	146.85126	149.49486	51200
cCCEA	149.99975	149.99995	149.99998	51200
iCCEA	149.99991	149.99997	149.99998	10982.34
rCCEA	50	149.99986	149.99998	19200

EC breeding method created children by selecting a parent, copying it, then "mutating" the copy by adding a gaussian random variable from a distribution with mean 0 and standard deviation 0.01, bounding the value to between 0 and 1. Parents were selected using "tournament selection", whereby two random parents are picked with replacement from the population, and the fitter of the two is selected. Runs lasted 50 generations.

The quality of a technique was defined as the average, over 250 independent runs, of the fitness of the best action (one per population) in the last generation of that run. The results usually have a peculiar bimodal distribution, with many values close to one of the two peaks. For this reason, we report information on the quartiles, as opposed to mean and standard deviation. For the same reason, we verify statistical significance via non-parametric t-tests combined using the Bonferroni correction. Sometimes these non-parametric tests will return the opposite result than a regular parametric test would, and although some conclusions are different from the ones reported in [4], we believe our comparison methodology is well-founded.

4.1 Experiment 1: MTQ and SMTQ, $H_1 = 50$

In the first experiment, we set H_1 to 50 to have a wide difference between the height of the two peaks. In this case, coevolution may have difficulties finding the global optimum primarily because its coverage is significantly smaller than that of the suboptimal peak.

Table 1 presents the performance of the four methods in the MTQ instance ($H_1 = 50$), as well as the number of evaluations required to achieve that performance. Although the differences appear small, the large number of observations (250) leads to statistically significant differences among the methods with confidence 95%. The results indicate that rCCEA performs worst, followed by pCCEA, cCCEA and iCCEA. There are statistically significant differences between all pairs of methods. And importantly, iCCEA achieves a significant reduction in the number of evaluations as compared to the other methods.

Figure 1 plots the number of evaluations required by the iCCEA algorithm at each generation. The algorithm starts with a complete round-robin evaluation (requiring 32×32 evaluations), followed by a drastic decrease in the number of evaluations. This is due to the fact that the number of actions needed to accurately rank the other population (hence the archive size) decreases significantly. In contrast, in the pCCEA algorithm the Pareto front in the MTQ domain is not discrete, and thus the archive grows rapidly in size until it occupies the whole population (Figure 2). At this point, learning stagnates.

Table 2 presents the results of the four methods in the SMTQ domain, as well as the number of evaluations required to achieve that performance. The rCCEA and pCCEA methods perform worst (with no statistically significant difference among them). cCCEA is better than both rCCEA and pCCEA. iCCEA significantly outperforms all other methods. The dynamics of the archive size for pCCEA and iCCEA are very similar to those in the MTQ domain: pCCEA's archive again rises to consume most of the population, while iCCEA's archive size rises to 8 early but converges to approximately 5. This is summed over both populations. The average archive size for each population is half the value: an average size of 4 early on, and 2.5 later.

4.2 Experiment 2: MTQ and SMTQ, $H_1 = 125$

Similar to the experiments in [4], we set H_1 to 125 to create a more deceiving domain instance: the actions on the suboptimal peak have higher fitness and they are thus more likely to be selected.

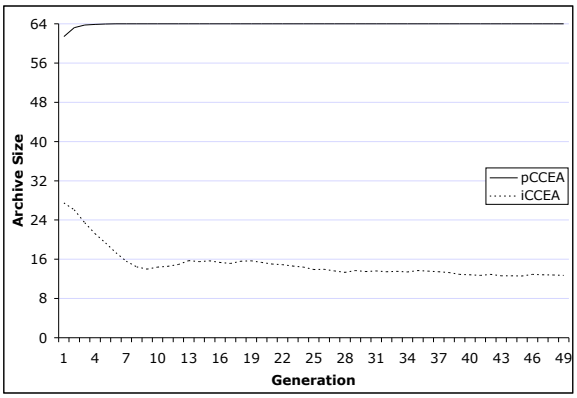


Figure 3: Average archive size for the iCCEA and the pCCEA algorithms in the OneRidge domain

Tables 3–4 present the results of the four methods in the MTQ and SMTQ domain instances with $H_1 = 125$. The results are consistent with the ones in Section 4.1: iCCEA is always better than cCCEA, which is in turn always better than pCCEA (with confidence 95%). The rCCEA method is worst: it finds the global optimum in only 16% of the runs in both the MTQ and the SMTQ domain ($H_1 = 125$). As before, iCCEA requires significantly fewer evaluations than the other methods. The average archive size is slightly higher (statistically significantly higher for iCCEA) than in the case of $H_1 = 50$, but it follows the same trend as that shown in Figure 2.

4.3 Experiment 3: The OneRidge Domain

Our last experiment examined the performance of the search methods in the OneRidge domain proposed in [21]. The OneRidge domain is defined as:

$$OneRidge(x,y) \leftarrow 1 + 2 * \min(x,y) - \max(x,y)$$

where x and y range between 0 and 1. OneRidge is particularly difficult for concurrent learners because it contains a very large number of Nash equilibria: for any value v between 0 and 1, (v,v) is a Nash equilibrium. This implies that for almost any Nash equilibrium (except for the global optimum $(1,1)$) there are an infinite number of better Nash equilibria that are infinitesimally close; unfortunately, both agents need to concurrently change their actions for the team to advance to these better solutions. To better study the algorithms' capacity to follow this ridge to the global optimum, we randomly initialized the populations of actions for each agent to only values smaller than 0.5.

If SMTQ adds more non-linear interactions among the agents, OneRidge goes even further. As a consequence, the methods have a very different ranking based on their performance in this domain (as shown in Table 5). cCCEA performs best, followed in order by rCCEA, iCCEA, and finally pCCEA; there are statistically significant differences among all the methods.

This poor performance of pCCEA and iCCEA seems unexpected at first. To shed more light onto the behavior of the two algorithms, we plotted the average archive size for pCCEA and iCCEA in Figure 3. As expected, the Pareto frontier in the OneRidge domain makes pCCEA think that *every possible action* is interesting and needs to be added to the archive; consequently, the size of pCCEA's archive is close to 64 even in the very first generations, and so learning stagnates. Unlike MTQ and SMTQ, it is relatively easy in the OneRidge domain to improve upon a joint action by small variations in the actions chosen by each agents. As a consequence, both cCCEA and rCCEA are able to improve until outperforming pCCEA.

Table 3: Results of the four methods in the MTQ domain instance with $H_1 = 125$

Method	First Quartile	Median	Third Quartile	Average # Evals
pCCEA	142.31862	149.03314	149.83407	51200
cCCEA	125	149.99974	149.99998	51200
iCCEA	125	149.99994	149.99998	11277.47
rCCEA	125	125	125	19200

Table 4: Results of the four methods in the SMTQ domain instance with $H_1 = 125$

Method	First Quartile	Median	Third Quartile	Average # Evals
pCCEA	125	145.66753	149.38516	51200
cCCEA	125	149.99977	149.99998	51200
iCCEA	125	149.99995	149.99998	11406.65
rCCEA	125	125	125	19200

Table 5: Results of the four methods in the OneRidge problem domain

Method	First Quartile	Median	Third Quartile	Average # Evals
pCCEA	1.45762	1.47224	1.48298	51200
cCCEA	1.89700	1.91584	1.93125	51200
iCCEA	1.50217	1.51778	1.53398	21545.3
rCCEA	1.82785	1.84294	1.86300	19200

The poor performance of iCCEA in this domain has a slightly different cause: the archive mechanism was designed to inform the concurrent learning processes of multiple Nash equilibria that are surrounded by large basins of attraction which cannot be avoided by small variations in actions. Given that OneRidge has no such equilibria, the archives serve little purpose, and they instead act to slow the optimization process by reducing the random exploration of the space. As shown in Figure 3, the average archive size of iCCEA is higher than *MaxEvals* (which was set to 5), and thus actions are not evaluated when in combination with random actions from the other population.

Revised Experiment. To test this hypothesis, we restricted the maximum archive size of iCCEA to only one action (*MaxArchiveSize* = 1 in iCCEA-Archive-Selection), and we performed another 250 runs in the OneRidge domain. The median performance of the algorithm was 1.84439, with a first quartile of 1.82448 and a third quartile of 1.86095. This is indistinguishable from the performance of the rCCEA method in Table 5. We further doubled the maximum number of generations for the iCCEA algorithm with a maximum archive size of 1 (this setting still involved only around 40% of the budget used by cCCEA and pCCEA), and we ran it another 250 times. The global optimum was consistently and precisely found in all of them. This is significantly better than all other algorithms we tested in this domain.

5. DISCUSSION AND CONCLUSIONS

It is rational for learning agents to explore those actions that are rewarded better. However, each agent's experimentation with a particular action directly affects the way the other agents perceive the search space. Therefore, we argue that agents may benefit from also exploring those actions that inform their teammates about the structure of the search space. We suggest that cooperating agents should be altruistic: the entire team may benefit if each agent helps its teammates to rank their actions better. We demonstrated this idea in a new cooperative coevolutionary algorithm, iCCEA, which requires significantly fewer evaluations to outperform other cooperative multiagent learning methods on our test problems. We also noted that too much altruism may hurt performance: an agent may waste resources when attempting to inform its teammates about the many Nash equilibria in the space. Restricting the number of informative actions an agent may choose solved the problem in our simple experiments, but we are exploring alternatives that can automatically balance the information provided to other learning agents with the desirability of searching for optimal joint actions. Future work will also examine formal models and guarantees for concurrent learners employing informative actions, as well as demonstrate such methods in complex multiagent domains.

6. REFERENCES

[1] T. Bäck, D. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. Oxford University Press, 1997.

[2] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Solving Transition Independent Decentralized Markov Decision Processes. *Journal of Artificial Intelligence Research*, 22:423–455, December 2004.

[3] R. Brafman and M. Tennenholtz. Efficient learning equilibrium. In *Advances in Neural Information Processing Systems (NIPS-2002)*, 2002.

[4] A. Bucci and J. Pollack. On identifying global optima in cooperative coevolution. In Hans-Georg Beyer et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2005*, pages 539–544. ACM, 2005.

[5] L. Bull. Evolutionary computing in multi-agent environments: Partners. In T. Back, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 370–377. Morgan Kaufmann, 1997.

[6] L. Bull. Evolutionary computing in multi-agent environments: Operators. In D. W. V W Porto, N Saravanan and A. E. Eiben, editors, *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pages 43–52. Springer Verlag, 1998.

[7] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of National Conference on Artificial Intelligence (AAAI-98)*, pages 746–752, 1998.

[8] K. De Jong. *Evolutionary Computation*. The MIT Press, 2006.

[9] E. Hansen, D. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 2004.

[10] P. Husbands and F. Mill. Simulated coevolution as the mechanism for emergent planning and scheduling. In R. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 264–270. Morgan Kaufmann, 1991.

[11] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-02)*, 2002.

[12] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 535–542, 2000.

[13] M. I. Lichbach. *The cooperator's dilemma*. University of Michigan Press, 1996.

[14] S. Luke. ECJ 13: A Java EC research system. Available at http://cs.gmu.edu/∼eclab/projects/ecj/, 2005.

[15] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[16] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3), 2005.

[17] L. Panait and S. Luke. Time-dependent collaboration schemes for cooperative coevolutionary algorithms. In *Proceedings of the 2005 AAAI Fall Symposium on Coevolutionary and Coadaptive Systems*, 2005.

[18] L. Panait, R. P. Wiegand, and S. Luke. Improving coevolutionary search for optimal multiagent behaviors. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 653–658, Acapulco, Mexico, 2003. Morgan Kaufmann.

[19] L. Panait, R. P. Wiegand, and S. Luke. A sensitivity analysis of a cooperative coevolutionary algorithm biased for optimization. In Kalyanmoy Deb et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2004*, page (to appear). Springer, 2004.

[20] L. Panait, R. P. Wiegand, and S. Luke. A visual demonstration of convergence properties of cooperative coevolution. In *Parallel Problem Solving from Nature — PPSN-2004*. Springer, 2004.

[21] E. Popovici and K. D. Jong. Understanding cooperative co-evolutionary dynamics via simple fitness landscapes. In Hans-Georg Beyer et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2005*, pages 507–514. ACM, 2005.

[22] M. Potter. *The Design and Analysis of a Computational Model of Cooperative CoEvolution*. PhD thesis, George Mason University, Fairfax, Virginia, 1997.

[23] Y. Shoham, R. Powers, and T. Grenager. On the agenda(s) of research on multi-agent learning. In *Proceedings of Artificial Multiagent Learning. Papers from the 2004 AAAI Fall Symposium. Technical Report FS-04-02*, 2004.

[24] D. Szer, F. Charpillet, and S. Zilberstein. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of the Twenty First Conference on Uncertainty in Artificial Intelligence*, 2005.

[25] R. P. Wiegand. *Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, Department of Computer Science, George Mason University, 2003.

[26] R. P. Wiegand, W. Liles, and K. De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1235–1242, 2001.

The Use of Agents in Human Learning Systems

Elizabeth Sklar
Dept of Computer and Information Science
Brooklyn College, City University of New York
2900 Bedford Ave, Brooklyn, NY, USA 11210
sklar@sci.brooklyn.cuny.edu

Debbie Richards
Dept of Computing, Div of Information and Communication Sciences
Macquarie University
Sydney NSW 2109 AUSTRALIA
richards@ics.mq.edu.au

ABSTRACT

This paper examines agent-based systems designed for a variety of human learning tasks. These are typically split into two areas: "training", which generally refers to adult learning of job-related skills, frequently but not exclusively in military settings; and "education", which generally refers to child and adult learning in academic settings, including primary and secondary schools, colleges and universities. While the terms may indicate diverse areas within the field of human learning, from the standpoint of agent-based systems development, many of the more prominent issues are held in common; as well, these issues can be generalized to most interactive agent-based environments. Here, we categorize three major trends in development of agents to assist human learners: pedagogical agents, peer learning agents and demonstrating agents. We highlight recent work within each of these categories, bringing to light common themes and issues.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence; K.3.1 [**Computers and Education**]: Computer Uses in Education

General Terms

Design, Human Factors, Experimentation

Keywords

learning, training, education, pedagogical agents

1. INTRODUCTION

The primary goal in a human learning environment is for the learner to advance. Software applications built for this environment, unlike applications for other environments, are not designed to simplify or perform a task *for* the user, but rather to help the user *learn* how to accomplish the task herself. This goal is strikingly different from typical interactive systems, where agents are constructed specifically to assist users by handling tedious or complicated jobs. Such systems address tasks such as browsing [33], sorting email and filtering news group messages [13, 17, 29, 30], or finding other users who share similar interests [4, 15, 28]. By embedding personalized agents and applying a variety of adaptive techniques, these systems can be thought of as tools designed to relieve the user's burden by taking over repetitive or complex duties from an overwhelmed human.

Malone [37, 38] makes an important distinction between *toys* and *tools* when discussing computer games. He defines toys to be systems that exist for their own sake, with no external goals; in contrast, tools are systems that exist because of their external goals. Good tools should be easy to use, in order to expedite the user's external goal. Good games are difficult to play, in order to increase the challenge provided to the player. Whereas typical interactive systems are like good tools, systems built for human learning should of course be easy to use, but primarily should provide challenges for the human learner. The external goal is for the user to learn how to perform a given task, so the system should make the process of learning how to accomplish that task easy — the *process*, not the task [64].

Systems designed to aid human learners are typically split into two categories:

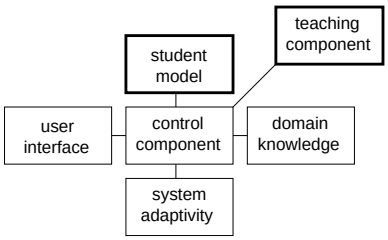
- **training**, which generally refers to adult learning of job-related skills, frequently but not exclusively in military settings; and
- **education**, which generally refers to child and adult learning in academic settings, including primary and secondary schools, colleges and universities.

While the terms "education" and "training" may indicate diverse areas within the field of human learning, from the standpoint of computer systems development, many of the more prominent issues are held in common. We use the general term "interactive learning system" (ILS) to include not only the more specific (and perhaps more familiar) term "intelligent tutoring system" (ITS), but also to provide a broader definition encompassing environments that are designed for more exploration on the part of the student than ITS's (which are typically more structured and scripted according to highly engineered, domain-dependent models). A typical ILS consists of the following components [39, 63]: (1) *domain knowledge* — a representation of the topic that the student is learning; (2) *teaching component* — an instructional model that is used to guide the student through the knowledge domain; (3) *user interface* — the interaction mechanism that lies between the human student and

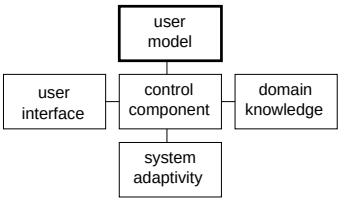
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

the computerized system; (4) *student knowledge* — a "user model" of the student in relation to the domain knowledge, indicating how much of and how well the student knows the domain; (5) *system adaptivity* — the means by which the system adapts automatically to the student's behavior, backtracking when the student makes mistakes and moving ahead when the student demonstrates proficiency with portions of the domain; and a (6) *control component* — the central part of the system architecture which holds all the pieces together.

Figure 1a illustrates these components and their relationships to each other. This architecture is quite similar to that of a typical interactive system for non-learning applications, as shown in Figure 1b. The *student model* is replaced by a *user model*, and there is no *teaching component*. In both cases, different types of agent technology are included at different levels. For example, where *interface agents* are used in non-learning applications to help a user navigate a complex domain, *pedagogical agents* are implemented in a learning application to guide users through a problem space. Understanding and embracing the special characteristics that learning tasks require of an application is necessary in order to adapt techniques from non-learning applications to those designed for education and training. While the focus in this article is on agent-based environments for human learning, we also highlight the relevance of issues for any agent-based interactive application.



(a) interactive learning system



(b) interactive system

Figure 1: Typical system architectures

There are three main types of agents embedded in human learning environments. The first are *pedagogical agents* [23], personalized assistants that interact directly with a learner and explicitly guide her through the domain. Referring back to Figure 1a, pedagogical agents are involved in the teaching component and user interface. Typically they consult the user model in order to understand the student and provide feedback that encourages the learner within her appropriate "zone of proximal development" [71]. The second type of agents are *peer learning agents*, the explicit use of agents as interactive partners in the learning process itself [27, 58]. These agents are built into the user interface and, as with pedagogical agents, have knowledge of the user. While these agents may have teaching capabilities, they are typically less engineered for guiding learning overtly than pedagogical agents. The third type of agents are *demonstrating agents*, where the agents themselves are interactive mediums for learning; for example, interactive agent-based simulations [54, 73] or educational robotics [19, 65]. These agents embody the domain knowledge and are removed from the other components of the learning system. An open area for future work is the development of systems which combine this third type of agent with one or both of the others.

Many systems employ an underlying agent architecture that is not necessarily apparent to the user. Agents can be implemented as part of any system component, but are particularly useful for the control component and providing a means for system adaptivity (e.g., [63, 67]). This is a natural approach as the use of independent modules that have specific tasks to perform while being able to interact with other independent modules fits nicely with the agent paradigm and with the principles of high cohesion and low coupling in software engineering. An example is the Dynamic Adaptive Learning study [67] that uses an agent-based architecture involving a student agent, evaluation agent, record agent, learning object agent and modeling agent. In this architecture, the agents cooperate to determine the appropriate learning objects to be presented to the student.

This paper focuses on discussing the three primary types of agents in use by learning systems today. Thus the work is organized as follows: section 2 describes the use of pedagogical agents; section 3 discusses ways in which agents act as peers in learning envionments; and section 4 highlights examples of environments in which students learn about various phenomena by interacting with agents that demonstrate aspects of the phenomena. Section 5 delves into the topic of testing and evaluation, briefly outlining key components of assessment in human learning and evaluation of computer-aided learning systems. Finally, we close by discussing current open issues.

2. PEDAGOGICAL AGENTS

Much like a narrator in a movie who provides voice-over to explain scenes but never actually appears in the film, a *pedagogical agent* will "pop up" when the learner indicates that she needs help. There are three methods used to determine when this help is needed: (1) directly, upon request by the user, for example by clicking a "help" button; (2) indirectly, by the system monitoring the learner's performance and automatically detecting when she seems to need assistance; and (3) mixed initiative, which relies on a combination of (1) and (2).

The work of Lucas *et al.* [35] involves the use of pedagogical agents that assist learners when the agent perceives that the user needs help, based on a model of the learner and the learning goals of the system. Animated Pedagogical Agents (APA)s are used to assist students in distance learning environments. The initial project involves popping up and providing guidance in the use of a calculator. The APAs work in conjunction with the initial loader (instantiated by the FIPA-OS Agent-Loader) and a graphical interface agent called the CalcAgent for handling the display, input and output to the calculator. Mbalad and Anyouzoa [46] have also used pedagogical agents to support online distance learning.

Johnson *et al.* [25] present comprehensive research on

animated pedagogical agents, highlighting issues that are faced when developing computer-based learning companions. Questions are addressed such as when to initiate dialog, as above, as well as how to include non-verbal communication and how to design for a wide range of client technologies. Animated agents can mimic human gestures and emotion-driven expressions, which can help engage students more than static avatars. Several agents were developed and tested, including: Steve (Soar Training Expert for Virtual Environments), Adele (Agent for Distance Learning, Light Edition) and Herman the Bug. Steve is embedded in an immersive three-dimensional environment that includes sound and was designed for naval training. Adele was built for use by university students over the internet, using standard workstation platforms and web technology. Herman the Bug is a "talkative" agent that gives advice to young students about plants, while teaching about botany. Animated agents can provide a number of advantages, such as navigational guidance, interactive demonstrations, gesturing for attention and use of other non-verbal communication devices. Empirical results using animated pedagical agents are positive, with some cautions. For example, there is evidence that young students may be distracted from the learning task by the agent's interface; however, a well-designed agent can keep students on-task and avoid this pitfall.

More recent work in this realm has focused on *interactive pedagogical drama* [41], where animated pedagogical agents become actors in a pseudo theatrical environment and learners either become immersed as participants in the drama or act as observers, like members of an audience. There are advantages to each approach; the former, active approach, requires learners to act in the drama which can be challenging and motivating, while the latter gives learners an opportunity to participate passively, with time for reflection and perhaps impartial analysis. Johnson [24] and Marsella *et al.* [41] use an interactive pedagogical drama for a system called "Carmen's Bright IDEAS" in which an adult human is guided through scenarios designed to improve problem-solving skills. Carmen, a character in the drama, is the mother of a pediatric cancer patient. She has a job and has another (healthy) child to take care of. When using the system, the learner observes Carmen's thoughts and can choose actions for her, in sessions with a counselor, discussions with her child's doctor, interactions with her boss, and so on. The pedagogical goal is for the human user of the system to improve her problem-solving skills and gain insight into similar situations in her own life.

The underlying architecture of Carmen's Bright IDEAS has been developed into a generalized framework called *Thespian* [62, 59] and applied to other domains. The Tactical Language Training System (TactLang) is a military application in which the learner engages in role-playing activities to acquire knowledge of the language, idiom and customs of particular geographic regions. One example is a drama which unfolds in a village cafe in the Middle East and the learner interacts with characters who are speaking Arabic. The system architecture integrates *PsychSim*, a multiagent system for mental modeling [52], with a story-line and basic script as well as pedagogical goals and social norms to guide agent-agent and agent-human interaction. The premise is that such an architecture will aid in rapid deployment of interactive pedagogical dramas.

3. PEER LEARNING AGENTS

There are many learning environments where agents interact with human learners as peers. These agents appear less intrusive than pedagogical agents. Many agent-based learning systems leverage game technology to provide both motivation and a situated, simulated training environment [31]. Human learners engage with agents as opponents or partners. These agents act more like peers than pedagagical agents who are more like tutors or instructors. The primary issues in developing peer learning agents involve (1) developing believable situated environments in which the human learner and agent peer interact, and (2) providing believable agents and a natural mode of interaction.

To address the issue of developing situated environments, which can become very expensive, many projects take advantage of a number of free or off-the-shelf game engines. For example, the TactLang mission environment (mentioned above) employs a modified version of the game engine Unreal Tournament, known as Gamebots [1]. Many games include *simulation elements* [3] that provide practical experience and *game elements* that offer an environment of engagement, discovery and competition [6]. Simulations can be particularly useful to provide training in several categories of skill sets: internalising processes, understanding systems, decision-making, perspective-shifting, team-building and cooperation [16].

Further, Wilkinson [74] states: "optimal learning occurs in the simulation of real world, problem-based activities. This happens in a safe environment where errors are expected, and failure deepens learning experience". The notion of a safe environment is particularly critical for the Mission Rehearsal Exercise (MRE) project [69]. Safety is also a key consideration in the training simulation being developed by Richards *et al.* [55] where agents are created in a game environment to allow the user to explore various risk scenarios. This project is focused on addressing issues concerning the agents' acquisition and reuse of knowledge and the language, cognitive and behavioural abilities of the agents to provide a more believable, engaging and immersive learning environment.

A number of game engines are available, with various strengths. Well-known engines include: Torque, Doom, Unreal Tournament, Aurora, Hammer, Embryo and Quake. Only some, however, are complete systems of high calibre, such as Hammer 4 and Doom 3. Simulation games have also been used in business environments, for example in teaching administrative skills. Off-the-shelf game simulations such as Doom II have been used in conjunction with free tools downloaded from the internet to provide cost-effective military training, for example, where real-world environments or locations may be unavailable to troops. The Gamebots project, mentioned earlier, started at the University of Southern California's Information Sciences Institute and is a prime example of the use of a game engine for agent-based training that was created for wider applications in artificial intelligence research. A good overview and lists of game engines is provided by Isakovic [21]. A shorter discussion is provided in Barles *et al.* [5].

To address the issue of providing believable agents, one strategy currently being explored by many comes from examining human characteristics such as emotion and empathy and extending agents to handle and even emulate such traits. Selvarajah and Richards [60] have developed an agent-based

emotion architecture to be used for psychological testing of subjects in a three-dimensional virtual reality Cocktail Party World. The study compared an avatar and an agent world to see whether agents reacting to the emotions of the subject and other agents in their vicinity was more realistic than the avatar world in which emotions were scripted. It was found that the agent world more deeply affected the subjects than the avatar world.

Similarly, van den Broek [70] developed empathic agents that mimic human empathy in a study concerning stress levels in individuals, where stress was detected through analysis of human voice recordings. In both of these studies, the goal was not to use agents to teach the subjects, like pedagogical agents would, but rather to add human abilities to the agent software so that the results would be more sociologically valid, a key concern and common shortcoming of laboratory style testing with humans. Also related to better understanding humans is the work of Sehaba and Estrailier [58], who use an agent approach to help rehabilitate children with autism. A multiagent system is used to model the knowledge of the therapists, the child's profile and the dynamics of the interaction.

A key difference between agent systems for human learning and other agent systems is the need for human communication languages as well as agent communication languages (ACL). In general, it is unrealistic to expect a human to communicate with a software agent using a FIPA-compliant ACL. The burden is on the system developer to create a communication channel that bridges the human-agent gap. It is common to use a personal agent to bridge this gap. Lazzari *et al.* [32] use personal agents in their Remote Assistant for Programmers (RAP) system to allow human users to interact with other humans and other parts of the system, which are represented by other types of agents. Personal agents are created for each online and offline user. Personal agents handle a range of tasks such as selecting answer types, submitting queries, finding answers, finding experts, receiving expert ratings, selecting experts, receiving answers and rating answers. Some of these tasks are performed in conjunction with other agents in the system. As in the case of RAP, personal agents are tailored to the particular human user based on a user profile, stereotype or model and often require a high degree of sophistication. This is clearly true in the case of the User Observation Agent (UOA) used in the studies of the behaviour of autistic children by Sehaba and Estrailier [58]. This agent takes input from a software/hardware system known as FaceLab to capture features of the face and the orientation of the gaze. In addition, the UOA takes into account actions with the mouse, touch screen and keyboard.

In a similar vein, Kim [27] uses agents as learning companions. The idea is motivated by the pedagogical strategy of providing a learner with peer support. This work reveals that the competency of the learner has a large impact on the nature of the interaction with agents as learning companions, referred to as PALs. Strong students, identified by their GPA[1], preferred for the PALs to take a leading role and expected to be given correct advice. Weak students preferred to control the PAL and asked for assistance only when they wanted it and were happy with some wrong answers, as they found that a PAL that was always correct was intimidating. These interesting findings emphasize the varied influence of agents for human-based learning. Not only will differences in settings and interfaces affect learners, but also the personalities of the learners themselves will be a factor. Refer to section 5 for discussion of issues relating to assessment of agent-based learning systems.

[1]Grade point average

4. DEMONSTRATING AGENTS

The notion of agency is useful for teaching and demonstrating a wide range of phenomena in the world. From introductory programming concepts [7] to dynamic systems [73], innovative researchers and teachers have developed virtual and embodied agents that interact with students of all ages. These environments take advantage of the popular and proven *constructionist* [49] pedagogical paradigm, motivating students and helping them to learn by doing. Here we highlight two complementary directions within this paradigm that concentrate on the use of agency to provide meaningful learning experiences: multiagent simulation and educational robotics.

Multiagent simulation allows students to program agents using simple commands and view graphically, and instantly, the effect of their code. This not only teaches students about programming concepts, but also provides powerful lessons in modeling. Students observe the world, invent rules about it, program them and analyse how well their rules represent the phenomena in the world they are attempting to model. Sengupta and Wilensky [61] use the NetLogo [48] agent simulation package to assist physics students to better understand the field of electromagnetics at the micro level. By modeling concepts such as electrons and atoms as agents, students are able to discover for themselves the emergent phenomena and learn to predict the behaviours in a way that produces deeper learning and understanding. The NetLogo Investigations in Electromagnetics (NIELS) studies demonstrate how learning in the domain can be broken down: "thinking in levels using multi-agent based models allows the students to establish concrete relationships between submicroscopic objects (e.g., electrons) which are shrouded in mathematical equations in traditional Physics instruction."

Blikstein and Wilensky [8] also employed the NetLogo environment, but this time to understand the difficult concepts involved in Materials Science. The MaterialSim system is a modeling-for-understanding framework that uses multiagent modeling languages where each agent is a basic computational construct with simple rules which control their behaviour and from which more complex higher level behaviours emerge. The study included classroom observations, pre/post interviews and data analysis of the usage session.

The simulation system Swarm [14, 68], based on the simple notion of ants in a nest gathering sugar, has been used to model, test and refine complex economic and language theories. Examples of other multiagent simulation environments include AgentSheets [2, 54] and RePast [53]. While the nature and architecture of environmental agents differs significantly from that of pedagogical agents, the use of modeling and simulation as a learning medium is shared by other applications. Examples are the training for risk situations project [55], MRE [69] and related projects.

Educational robotics refers to the use of robots in classrooms to teach a wide variety of topics, not necessarily robotics in particular. With the advent of the LEGO Mind-

storms Robotics Invention Kit [47] in the late 1990's, today robots are being used all over the world to engage students from early primary through undergraduate classrooms [7, 65]. While some courses focus more on the hardware and engineering design aspects of robotics [42, 43], others concentrate on control mechanisms, agency, behavior-based paradigms and multiagent systems [44, 50, 76]. Many include some type of contest in the course to help motivate students as well as take advantage of fully tested and readily available environments, such as Botball [9] and RoboCup [56, 65]. A broad range of experience reports have been published detaling lessons learned using robotics with younger students, in primary and secondary school classrooms and after-school programs [19].

As well as hands-on, hardware-based approaches, a number of simulators have been developed recently to give students who do not have access to robot hardware an opportunity to explore the concepts behind controlling robots or to speed up development by providing a rapid-prototyping environment where debugging can occur more quickly than on real robots. Chu *et al.* [12] have developed an agent-based simulation environment for children, designed to be used in conjunction with the popular RoboLab [57] graphical programming interface and LEGO Mindstorms robot. The motivation is to give students an opportunity to learn about agent-based programming by using RoboLab in a "safe and friendly" place – they can "try out" programs in the simulator before loading them onto the robot platform and before being faced with real-world, physical constraints and issues such as noise.

When dealing with varying robotic platforms, even with a relatively simple one like the LEGO Mindstorms, it is effective for students to be able to classify and specify behavior patterns for their robots. Behavior patterns can range from basic behaviors, such as "move forward for 4 seconds", to complex behaviors such as "open a gate". Due to the complexity of defining and implementing robot behaviors, many control architectures are designed to build complex behaviors out of simple, low-level commands [45]. Another approach by Goldman [18] provides a custom behavior-based interface to RoboLab and an XML-based translator for downloading on to the Sony AIBO robot.

5. TESTING AND EVALUATION

One of the more prominent issues that separates human learning systems from other agent-based and interactive system development is the aspect of testing and evaluation. It is expected that not only the software learning environment be fully debugged and tested, but also, particularly amongst education researchers, the system must be evaluated with respect to its effectiveness as a learning environment. This section provides a brief description of evaluation in this context.

While there is no fixed standard for evaluating the effectiveness of interactive learning systems, there are two generally accepted categories of assessment [34, 40]:

- *formative assessment* tests the design and behavior of a system in-progress, generally performed by computer scientists, system designers and builders; and
- *summative assessment* evaluates the effectiveness of a completed system, generally performed by educators and/or psychologists.

Researchers begin by identifying what is being evaluated. Design and performance aspects need to be examined differently. The nature of the testing will vary depending on whether the goal is to assess the theoretical basis underlying the system or the software components themselves.

Each of a learning system's components (see Figure 1a) can be evaluated individually. Domain knowledge should be checked for accuracy and coverage. The teaching component can be evaluated for the range of instructional method(s) offered, its level of adaptability and the degree to which its instruction is based on proven educational and psychological methods. The user interface can be examined by comparing multiple user interfaces for the same underlying engine and looking, in particular, for improvement in student learning. System adaptivity can be compared using interactions at different skill levels. The control component can be evaluated using various system performance measures, such as speed. Finally, and probably the most important, improvement in student knowledge (i.e., learning) can be measured using the same criteria in a computer-based environment that are employed within standard educational and/or psychological testing. These include: (1) *validity* — does the test show evidence that it measures what it says it measures? (2) *reliability* — are multiple results for the same subject consistent? (3) *objectivity* — is the test administered and scored the same way for every participant? (4) *standardisation* — can results be translated into a meaningful representation of student performance?

The techniques for performing assessments vary depending on which component is being evaluated, where in the system development cycle the evaluation is being performed and who is performing the evaluation. Similar to typical HCI lifecycle models, such as the star life cycle [51] that involves evaluation after each step, evaluation of learning systems typically follows an iterative cycle. Beginning with *system development* and extending through to *experimental research*, steps may be revisited at any time during the formative phases of system development. Once summative assessment begins, in the experimental research phase, the system cannot change; otherwise, the summative results will be invalid. *Pilot testing* often occurs late during formative assessment, bridging the gap to summative assessment. There are three methods of pilot testing: (1) *one-to-one*, which is performed early in the development cycle, with one student, instructor, trainer or researcher providing feedback; (2) *small-group*, which is performed later in the development cycle, with a small group of students, instructors or trainers providing feedback; and (3) *field*, which is performed near the end of development, emulating experimental conditions with teachers or trainers and students in a "live" (i.e., classroom) setting.

Other techniques are more pertinent during summative assessment. In *criterion-based evaluation*, a general list of guidelines is developed and systems are evaluated based on their adherence to these guidelines, for example, program construction, behavior and characteristics. While developing relevant criteria is not an easy task, this method may prove useful in formative assessment and in comparing different systems. With *expert knowledge and behavior* assessment, system performance is compared to that of a human expert performing the same task. Software systems may be subjected to a standard *certification* process, through careful examination by qualified human experts. In *sensitivity*

analysis, the responsiveness of a system is tested on a variety of different user behaviors. This may be particularly useful for evaluating system adaptivity. After system development and pilot testing are complete, *experimental research* can begin. The conditions should be the same as those during the field testing phase.

Two mechanisms for collecting evaluation data are common:

- *quantitative*, in which numerical data is analysed, frequently by comparing scores on pre- and post-tests and surveys, to measure changes in student performance and attitudes; and
- *qualitative*, in which interviews and surveys are conducted and observations are made.

Mixed methods research [22] combines the two, but typically, at least in the education arena, researchers tend to adhere to the methods of one category or the other. Quantitative methods rely on standards type testing, with multiple choice style questioning and Likert scale surveys. System logs are also examined. Qualitative, or "open", methods encompass data taken in both written and oral forms, as part of interviews, questionaires and open surveys containing short-answer questions (rather than multiple choice). Transcripts are "coded" and analysed based on measures such as frequency of broad term usage, often borrowing techniques from natural language processing in order to compute semantic similarity between answers.

Reviewing the literature describing agent-based systems issues, written from an agents research perspective (as opposed to an education research perspective), the primary type of evaluation is formative assessment, particularly testing the accuracy and functionality of system design and early pilot testing. Within the sampling of literature referenced earlier in this article, a wide range of test environments have been used, including formal primary, secondary and undergraduate classrooms, research laboratories, industrial workplaces, military bases and distance learning (or "e-learning") settings. As well, there are broad differences in the maturity of systems presented. Some have completed only the design phase while others have been fully implemented. Many have undergone some aspects of formative testing, including architecture and system design reviews, and user interface studies. Few have reached the summative testing phases, but some have conducted focus groups and pilot studies.

Many systems which are designed do not get past the prototype phase for various reasons. Firstly, gaining access to human subjects, particularly minors, may be difficult. Hurdles can include: availability of representative population; costs for setup and recruitment and human ethics requirements. One of the more serious issues is that learning systems take so long to develop, by the time they are operational, the customer does not need or want them any more. This is frequently due to the large costs (in time and money) associated with building a teaching component into a system, which adds to the typically unwieldy costs of developing software on time and within budget [10]. Nonetheless, just as one would expect user evaluation to be found in research publications concerning user interfaces, there is a higher expectation in the work reported on agent-based human learning systems that an evaluation with real users will be conducted, will be well designed and analysed carefully.

6. DISCUSSION

The types of research, application development and studies being performed using agent technology for human learning are varied. In some cases, the central theme of the project is to explore and extend current agent technologies and capabilities. In other cases, agent theory is not being developed but rather is being applied and the research focuses on the application.

Differences in learners' ages and genders must be considered when designing agents that will interact as pedagogical tutors or peer learners. While most learning systems using games are focused on making the learning more palatable for children, the motivation of Richards *et al.* [55] is quite different and in keeping with Kearsley [26] who has found that "instruction for adults needs to focus more on the process and less on the content being taught. Strategies such as case studies, role-playing, simulations and self-evaluation are most useful. Instructors adopt a role of facilitator or resource rather than lecturer or grader." Using these virtual environment technologies in conjunction with agent components allows production of less expensive and more accessible systems that offer increased control of the environment together with increased relevance to the real world.

Many studies have highlighted differences in the way females and males approach, interact with and think about technology. Inkpen [20] found gender differences in the way children approach game environments. Girls tended to perform better when another person was also playing on the same machine, but boys performed better when the other player was on a different machine. Girls were found to have less physical contact with their human partner or the mouse compared to boys. Brunner *et al.* [11] showed that females tend to view technology as a tool used to facilitate human interaction, whereas males tend to view technology as an object that can be used to extend their abilities and/or power. These gender-based attitudinal differences will effect a student's experience with a learning environment; as well, these issues generalize to any interactive agent-based system.

A more philosophical question addresses the intersection of agent-based technologies and interactive learning systems. What does each do for the other? Agent-based systems, since the infant stages of the field [36, 75], have been highly focused on finding effective, modular means to intelligently interact with human users in a personalized, but not annoying way (go away, Mr Clippy). The notion of an automated assistant is quite natural in a learning or tutoring environment. The difficulty comes in building generalizable systems because effective tutoring, by both humans and agents, involves deep understanding of the knowledge domain being taught. Knowledge engineering, identifying and fixing common bugs in students' learning paths and constructing tutoring agents that reflect the experience of a master teacher is a daunting endeavor, shadowed by the desire to avoid repeating the mistakes of expert systems. Agents have been shown to learn to adapt in dynamic environments, such as robotic soccer [66] and electronic markets [72]. If we view the human learner as an agent's changing environment, perhaps tomorrow's solutions will include agents that can learn to teach.

Acknowledgments

This work was partially supported by NSF #ITR-02-19347.

7. REFERENCES

[1] R. Adobbati, A. N. Marshall, A. Scholer, S. Tejada, G. A. Kaminka, S. Schaffer, and C. Sollitto. Gamebots: A 3D Virtual World Test-Bed for Multi-Agent Research. In *Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, 2001.

[2] `http://agentsheets.com/`.

[3] C. Aldrich. *Simulations and the Future of Learning : An Innovative (and Perhaps Revolutionary) Approach to E-Learning*. Pfeiffer, San Francisco, 2003.

[4] M. Balabanović. *Learning to Surf: Multiagent Systems for Adaptive Web Page Recomendation*. PhD thesis, Stanford University, 1998.

[5] J. Barles, M. Dras, M. Kavakli, D. Richards, and A. Tychsen. An overview of training simulation research and systems. In *Agent-based Systems for Human Learning, AAMAS Workshop*, 2005.

[6] R. Bartles. *Designing Virtual Worlds*. New Riders, Indianapolis, IN, 2003.

[7] D. S. Blank, D. Kumar, L. Meeden, and H. Yanco. Pyro: A Python-based Versatile Programming Environment for Teaching Robotics. *Journal on Educational Resources in Computing (JERIC), Special Issue on robotics in undergraduate education, part I*, 4(2), 2005.

[8] P. Blikstein and U. Wilensky. Less is more: Agent-based simulation as a powerful learning tool in materials science. In *Agent-based Systems for Human Learning, AAMAS Workshop*, 2005.

[9] `http://www.botball.org`.

[10] F. P. Brooks. *The mythical man-month: essays on software engineering*. Addison-Wesley, 1975.

[11] C. Brunner, D. Bennett, and M. Honey. Girls' games and technological desire. *From Barbie to Mortal Kombat: Gender and Computer Games*, 1998.

[12] K.-H. Chu, R. Goldman, and E. Sklar. Roboxap: an agent-based educational robotics simulator. In *Agent-based Systems for Human Learning, AAMAS Workshop*, 2005.

[13] A. Cypher. Eager: Programming repetitive tasks by example. In *Computer-Human Interaction*, 1991.

[14] J. M. Epstein and R. Axtell. *Growing artificial societies: social science from the bottom up*. The Brookings Institution, Washington, DC, USA, 1996.

[15] L. Foner. Yenta: A multi-agent referral based matchmaking system. In *First International Conference on Autonomous Agents*, 1997.

[16] L. Galarneau. The elearning edge: Leveraging interactive technologies in the design of engaging, effective learning experiences. In *e-Fest 2004*, 2004.

[17] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave and information tapestry. *Communications of the ACM*, 35(12), 1992.

[18] R. Goldman. From RoboLab to AIBO: Capturing Agent Behavior. Master's thesis, Columbia University, 2005.

[19] R. Goldman, A. Eguchi, and E. Sklar. Using Educational Robotics to Engage Inner-City Students with Technology. In *Sixth International Conference of the Learning Sciences*, 2004.

[20] K. Inkpen. Gender differences in an electronic games environment. In T. Ottman and I. Tomek, editors, *ED-MEDIA 94 World Conference on Educational Multimedia and Hypermedia*, 1994.

[21] `http://cg.cs.tu-berlin.de/~ki/engines.html`.

[22] R. B. Johnson and A. J. Onwuegbuzie. Mixed methods research: a research paradigm whose time has come. *Educational Researcher*, 33(7), October 2004.

[23] W. Johnson. Pedagogical agents for virtual learning environments. In *International Conference on Computers in Education*, 1995.

[24] W. L. Johnson. Pedagogical Agent Research at CARTE. *AI Magazine*, Winter, 2001.

[25] W. L. Johnson, J. W. Rickel, and J. C. Lester. Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments. *International Journal of Artificial Intelligence in Education*, 11, 2000.

[26] G. Kearsley. Explorations in learning and instruction: The theory into practice database. `http://tip.psychology.org/index.html`, 2004.

[27] Y. Kim. Agent learning companions: Learners expectations of the desirable characteristics. In *Agent-based Systems for Human Learning, AAMAS Workshop*, 2005.

[28] D. Kuokka and L. Harada. Matchmaking for information agents. In M. Huhns and M. Singh, editors, *Readings in Agents*. Morgan Kaufman, 1997.

[29] K. Lang. Newsweeder: Learning to filter news. In *Twelfth International Conference on Machine Learning*, 1995.

[30] Y. Lashkari, M. Metral, and P. Maes. Collaborative interface agents. In *Twelfth National Conference on Artificial Intelligence*. AAAI Press, 1994.

[31] J. Lave and E. Wenger. *Situated learning: Legitimate peripheral participation*. Cambridge University Press, Cambridge, England, 1991.

[32] L. Lazzari, M. Mari, A. Negri, and A. Poggi. Support remote software development in an open distributed community. In *Agent-based Systems for Human Learning, AAMAS Workshop*, 2005.

[33] H. Lieberman. Letizia: An agent that assists web browsing. In *International Joint Conference on Artificial Intelligence*, 1995.

[34] D. Littman and E. Soloway. Evaluating itss: the cognitive science perspective. In M. C. Polson and J. J. Richardson, editors, *Foundations of Intelligent Tutoring Systems*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1988.

[35] J. P. Lucas, B. Wilges, and R. A. Silveira. Inserting animated pedagogical agents inside distributed learning environments by means of fipa specifications. In *Agent-based Systems for Human Learning, AAMAS Workshop*, 2005.

[36] P. Maes. Modeling Adaptive Autonomous Agents. *Artificial Life: An Overview*, 1995.

[37] T. Malone. Toward a theory of intrinsically motivating instruction. *Cognitive Science*, 4, 1981.

[38] T. Malone. What makes computer games fun? *Byte*, December, 1981.

[39] M. A. Mark and J. E. Greer. Evaluation

methodologies for intelligent tutoring systems. *Journal of Artificial Intelligence and Education*, 4, 1993.

[40] M. A. Mark and J. E. Greer. Evaluation methodologies for intelligent tutoring systems. *Journal of Artificial Intelligence and Education*, 4, 1993.

[41] S. Marsella, W. L. Johnson, and K. LaBore. Interactive Pedagogical Drama. In *Fourth International Conference on Autonomous Agents*, 2000.

[42] `http://fredm.www.media.mit.edu/people/fredm/projects/6270/`.

[43] F. Martin. *Robotic Explorations: A Hands-On Introduction to Engineering*. Prentice Hall, 2000.

[44] `http://www-scf.usc.edu/~csci445/`.

[45] M. J. Mataric. Behavior-based robotics as a tool for synthesis of artificial behavior and analysis of natural behavior. *Trends in Cognitive Science*, 2(3), March 1998.

[46] A. Mbala and A. G. N. Anyouzoa. A multi-agent system to support users in online distance learning. In *Agent-based Systems for Human Learning, AAMAS Workshop*, 2005.

[47] `http://www.legomindstorms.com/`.

[48] `http://ccl.northwestern.edu/netlogo/`.

[49] S. Papert. Situating constructionism. *Constructionism*, 1991.

[50] S. Parsons and E. Sklar. Teaching AI using LEGO Mindstorms. In *AAAI Spring Symposium 2004 on Accessible Hands-on Artificial Intelligence and Robotics Education*, 2004.

[51] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey. *Human-Computer Interaction*. Addison-Wesley, 1994.

[52] D. V. Pynadath and S. C. Marsella. PsychSim: Modeling Theory of Mind with Decision-Theoretic Agents. In *International Joint Conference on Artificial Intelligence*, 2005.

[53] `http://repast.sourceforge.net/`.

[54] A. Repenning and W. Citrin. Agentsheets: Applying grid-based spatial reasoning to human-computer interaction. In *IEEE Symposium on Visual Languages*, 24-27 1993.

[55] D. Richards, M. Kavakli, and M. Dras. Training for high risk situations. In *Agent-based Systems for Human Learning, AAMAS Workshop*, 2005.

[56] `http://www.robocup.org`.

[57] `http://www.ceeo.tufts.edu/robolabatceeo/`.

[58] K. Sehaba and P. Estraillier. A multi-agent system for rehabilitation of children with autism. In *Agent-based Systems for Human Learning, AAMAS Workshop*, 2005.

[59] M. Sei, S. C. Marsella, and D. V. Pyndath. Thespian: Using multi-agent fitting to craft interactive drama. In *International Conference on Autonomous Agents and Multi-Agent Systems*, 2005.

[60] K. Selvarajah and D. Richards. The use of emotions to create believable agents in a virtual environment. In *International Conference on Autonomous Agents and Multi-Agent Systems*, 2005.

[61] P. Sengupta and U. Wilensky. Niels: An emergent multi-agent based modeling environment for learning physics. In *Agent-based Systems for Human Learning, AAMAS Workshop*, 2005.

[62] M. Si, S. C. Marsella, and D. V. Pynadath. THESPIAN: An Architecture for Interactive Pedagogical Drama. In *International Conference on Artificial Intelligence in Education*, 2005.

[63] E. Sklar. *CEL: A Framework for Enabling an Internet Learning Community*. PhD thesis, Department of Computer Science, Brandeis University, 2000.

[64] E. Sklar. Agents for Education: When too much intelligence is a bad thing. In *Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2003.

[65] E. Sklar, S. Parsons, and P. Stone. Using RoboCup in university-level computer science education. *Journal on Educational Resources in Computing (JERIC), Special Issue on robotics in undergraduate education, part I*, 4(2), June 2004.

[66] P. Stone, G. Kuhlmann, M. E. Taylor, and Y. Liu. Keepaway Soccer: From Machine Learning Testbed to Benchmark. In *RoboCup-2005: Robot Soccer World Cup IX*. Springer Verlag, 2005.

[67] S. Sun, M. Joy, and N. Griffiths. An agent-based approach to dynamic adaptive learning. In *Agent-based Systems for Human Learning, AAMAS Workshop*, 2005.

[68] `http://www.swarm.org`.

[69] W. Swartout, R. Hill, J. Gratch, W. L. Johnson, C. Kyriakakis, C. LaBore, R. Lindheim, S. Marsella, D. Miraglia, B. Moore, J. Morie, J. Rickel, M. Thiebaux, L. Tuch, R. Whitney, and J. Douglas. Toward the holodeck: Integrating graphics, sound, character and story. In *Fifth International Conference on Autonomous Agents*, 2001.

[70] E. van den Broek. Empathic agent technology (eat). In *Agent-based Systems for Human Learning, AAMAS Workshop*, 2005.

[71] L. S. Vygotsky. *Mind in society: The development of higher psychological processes*. Harvard University Press, Cambridge, MA, 1978.

[72] V. Walia, A. Byde, and D. Cliff. Evolving market design in zero-intelligence trader markets. Technical report, Hewlett-Packard Research Laboratories, Bristol, England, 2003.

[73] U. Wilensky and M. Resnick. Thinking in levels: A dynamic systems perspective to making sense of the world. *Journal of Science Education and Technology*, 8(1), 1999.

[74] D. Wilkinson. The intersection of learning architecture and instructional design in e-learning. In *e-Technologies in Engineering Education: Learning Outcomes Providing Future Possibilities*, 2002.

[75] M. Wooldridge and N. R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2), 1995.

[76] `http://www.cs.uml.edu/~holly/91.450/`.

Reinforcement Learning for Declarative Optimization-Based Drama Management

Mark J. Nelson, David L. Roberts, Charles L. Isbell, Jr., Michael Mateas
College of Computing
Georgia Institute of Technology
Atlanta, Georgia, USA
{mnelson, robertsd, isbell, michaelm}@cc.gatech.edu

ABSTRACT

A long-standing challenge in interactive entertainment is the creation of story-based games with dynamically responsive story-lines. Such games are populated by multiple objects and autonomous characters, and must provide a coherent story experience while giving the player freedom of action. To maintain coherence, the game author must provide for modifying the world in reaction to the player's actions, directing agents to act in particular ways (overriding or modulating their autonomy), or causing inanimate objects to reconfigure themselves "behind the player's back".

Declarative optimization-based drama management is one mechanism for allowing the game author to specify a drama manager (DM) to coordinate these modifications, along with a story the DM should aim for. The premise is that the author can easily describe the salient properties of the story while leaving it to the DM to react to the player and direct agent actions. Although promising, early search-based approaches have been shown to scale poorly. Here, we improve upon the state of the art by using reinforcement learning and a novel training paradigm to build an adaptive DM that manages the tradeoff between exploration and story coherence. We present results on two games and compare our performance with other approaches.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Artificial Intelligence-Learning; J.5 [**Computer Applications**]: Arts and Humanities

Keywords

reinforcement learning, drama management, interactive entertainment

1. INTRODUCTION AND MOTIVATION

We are interested in adding narrative structure to large open-world games in a way that is not pre-scripted and allows the player to interact with and influence the story. Many modern games have rich, non-linear plotlines, with multiple endings, complex story branching and merging, and multiple subplots. We would like the player to be able to have a significant impact on what happens, rather than following along with a prewritten script, or being in control at only a few decision points. However, in contrast to completely open games, the story that results should exhibit global narrative coherence and reflect the game author's aesthetic goals.

Traditionally, story in games is guided by local triggers. Progress in a linear story depends solely on how much of the story has unfolded. In slightly more complex situations, the author can specify condition-action rules (*e.g.* "`if` the player is in the room `and if` the player is carrying a gun, `then` have the non-player character (NPC) hide behind the counter"). The author must specify rules for every possible combination of conditions, a tedious (and intractable) burden for stories of any complexity. Further, the author will find it difficult to achieve a globally cohesive, pleasing story purely from local rules.

This difficulty has been recognized for some time, and grows worse as players demand both true interactive agency and rich stories. One solution is a *drama manager* that watches a story progress, reconfiguring the world to fulfill the author's goals. A drama manager may notice a player moving along a currently undesirable subplot, and attempt to lure the player toward a better one. To do so, it requests that one of the agents in the world take some action; for example, it may instruct an NPC to start a conversation with the player. It may even make requests of objects that appear inert to the player, such as telling a door to lock itself (of course, reconfiguring inert objects must be done carefully without the player noticing). Thus the drama manager acts as a central director of the agents in the game world, making requests of them when necessary to maintain a globally coherent story-line.

We base our work on a specific framework in which a story is represented as a sequence of discrete *plot points*, and the drama manager has a set of requests it can make of agents, which we call *DM actions*.

Plot points are important possible events in the story, such as a player discovering information or an object. DM actions are any requests for which the game author has implemented an agent that can respond to the request. Plot points have a set of (boolean) prerequisites defined as other plot points. For example, the plot point *found book* may have the pre-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

requisite *found key*, if it's hidden in a locked room. Prerequisites restrict the set of plot-point sequences—and thus stories—to those that are physically possible. Note that in nonlinear games with many plot points there are still a large number of possible stories.

Plot points are abstractions of concrete events happening in the game world: For example, in a story that we use as a case study, *Anchorhead*, the player can find a safe in an old mansion. Finding the safe involves a number of concrete actions, such as entering the room in which it's located, reading a description of the room, possibly examining various objects in the room, searching the bookshelf on the wall, and finally picking up the book behind which the safe is hidden. The drama manager does not need to know the details of all these actions, since most do not impact the progression of the story; instead, it is only notified of the abstract plot point *found safe*. It is, however, an assumption of this approach that the abstract version of the story represented by a sequence of plot points is sufficient to judge story quality.

Drama-manager actions are similarly abstract, but correspond to sequences of concrete actions in the game world. For example, the DM action *hint safe location* may result in a complex series of concrete events, in which an NPC tries to draw the player into a conversation and let slip a hint. To the drama manager, what matters is the expected outcome of the request, so DM actions are given as an ordered pair (*action*, *plot point*), where *action* is one of: *cause*, *deny*, *temporarily deny*, *reenable*, or *hint*. These actions cause the plot point in question to, respectively, either: happen immediately (*e.g.* an NPC gives the player an item); become impossible for the duration (*e.g.* inconspicuously make an item disappear from the world entirely before the player has noticed it); become temporarily impossible (*e.g.* tell an NPC to refuse to speak to the player for now); become possible again (*e.g.* tell the NPC to be talkative again); or become more likely by leading the player towards the plot point (*e.g.* tell an NPC to hint about an item's location). Actions have constraints. They must be consistent (a plot point cannot be *reenabled* unless it has previously been *temporarily denied*), and there can also be constraints on when, from the author's perspective, the action makes sense (*e.g.* an action that *causes* something should only be taken at points in the story when it is plausible to cause its target plot point).

Finally, the author provides an evaluation function that takes a sequence of plot points and the history of drama-manager actions, and rates the completed story. This is the central feature of what we call *declarative optimization-based drama management* (DODM): The author specifies *what* constitutes a good story, and leaves to the drama manager the problem of *how* to bring it about in the complex environment of a continually-progressing game.

To specify an evaluation function, the author will generally annotate plot points or DM actions with some information and then specify a *feature* using that information, which is a numerical score rating how much a particular story exhibits some desired property. The final evaluation function is a weighted sum of these features, in accordance with the author's judgment of their importance. For example, the author may annotate plot points with the subplot they belong to, and assign a higher score to stories where the player progresses along one subplot for several plot points in a row; or, the author may annotate plot points with a description of the knowledge they provide to the player, and assign a higher score to stories where knowledge unfolds gradually and climaxes in a burst of revelations near the end (*e.g.* in a mystery story). Actions can be annotated with a judgment of how likely the player is to sense that the drama manager is fiddling with the world; stories would then be assigned scores in inverse proportion to how much total manipulation took place. These desired features may in many cases conflict in surprising and constantly changing ways; the drama manager is responsible for making these complex tradeoffs.

In short, the problem of drama management is to optimize the use of the available drama-manager actions in response to a player's actions, given the set of plot points and an evaluation function. This is well-modeled as a reinforcement learning (RL) problem. RL algorithms attempt to learn a policy that, given a world modeled by states and a (possibly unknown) transition function between states, specifies an action to take in each state in a manner that maximizes expected evaluation according to some function. In this case, state is specified by the sequence of plot points and drama-manager actions that have happened so far (order is important); actions are the drama-manager actions plus the null action (do nothing); transitions are the likelihood of plot points occurring in any given state, and are given by the combination of prerequisites for a plot point, the player, and any influence from drama-manager actions; and the evaluation function is the one provided by the author. Although nicely defined, this is a complicated reinforcement-learning problem: the player is difficult to model and highly stochastic, and the state space is extremely large.

In the rest of this paper, we present our approach to this problem. We review earlier search-based techniques, discuss our application of temporal-difference (TD) learning methods to the problem, and show how our novel approach to training, *self-adversarial / self-cooperative exploration* (SASCE), outperforms search-based methods and improves the performance of TD methods.

2. RELATED WORK

Using a drama manager to guide interactive drama was first proposed by Laurel [3]. The story and drama-manager formalism we use—and treating drama management as an optimization problem—was proposed by Bates as *search-based drama management* (SBDM) [1]. SBDM formulates the drama management problem similarly to a minimax game-tree search problem as has been used for games like chess; however, because the player in interactive drama is not truly an opponent, expectimax search is used, alternating maximizing nodes where the drama manager chooses its best available action with expectation-calculating nodes where the player is assumed to act according to some probabilistic model.

Weyhrauch [12] applied SBDM to a simplified version of the Infocom interactive fiction *Deadline*, named *Tea for Three*. Full-depth search is intractable, so he used shallow searches with static cutoff values calculated by sampling a small number of possible stories that could follow the cutoff and averaging their evaluations, achieving impressive results on *Tea for Three*. Lamstein & Mateas [2] proposed reviving the technique, but work by Nelson & Mateas [7] on a different story, *Anchorhead*, showed that the impressive performance of Weyhrauch's limited-depth sampling search did

not generalize to other stories.

In formulating drama management as an RL problem, temporal-difference (TD) learning [9] seems a natural candidate algorithm. In particular, the approach used by TDGammon [10] to learn a policy for playing backgammon as well as the best human players [11] seems a useful starting point. Some have argued (to some controversy) that these results are due in part to the particular stochastic properties of backgammon [8]. As we shall see, we address this concern by modifying the stochastic properties of the training regimen.

There have also been several other drama management systems that take quite different approaches.[1]

The Mimesis architecture [13] constructs story plans for real-time virtual worlds. The generated plans are annotated with a rich causal structure, and the system monitors for player actions that might threaten causal links, replanning or preventing player actions if a threat is detected. This approach differs substantially from ours in that authors specify concrete goals for the planner to achieve as part of the story, while an author in DODM specifies abstract and possibly conflicting features that the story as a whole should possess.

The Interactive Drama Architecture [4] takes a prewritten plot and tries to keep the player on the plot by taking corrective action according to a state-machine model of likely player behavior. This addresses a much different problem than ours, since we're interested in letting players' actions substantially affect the overall plot, rather than just incorporating their actions into variants of a prewritten plot.

The beat-based drama manager in *Façade* [6] sequences small pieces of story progression called dramatic beats, constructing the story out of these directly-chosen pieces. This is substantially different from the DODM approach, which for the most part lets the story run itself in the traditional way, stepping in to instruct the agents with DM actions only when it seems necessary. We hypothesize that this makes beat-based drama management better suited to tight story structures, where ideally all activity in the story world contributes to the story, while DODM lends itself more naturally to more open-ended games, especially in large worlds that may include much player activity not directly relevant to the story.

3. METHODS

In this paper, we focus on quantitatively evaluating the performance of the optimization component of DODM. Given a story world modeled by plot points, a set of available drama-manager actions, and an author-supplied evaluation function, we can determine to what extent the drama manager succeeds in maximizing story quality according to that evaluation function. To do so, we use the drama manager to guide simulated games many times, score each resulting game using the evaluation function, and plot the distribution of scores on a histogram. For a baseline distribution, we run the same simulations without a drama manager (equivalent to a null drama manager that never takes any DM actions). If the drama manager is successful, its distribution will be generally skewed towards the right compared to the baseline: high-quality stories will appear more often, and poor-quality stories will happen infrequently if at all. For the simulations, we use a fairly naive player model (proposed by Weyhrauch) that assumes the player is exploring the world; the player chooses uniformly from available plot points (plot points whose prerequisites are satisfied), with increased probability of doing something the drama manager has hinted at.

We compare three approaches to optimizing the selection of DM actions: the limited-depth sampling search (SAS+) proposed by Weyhrauch; standard temporal-difference (TD) learning; and TD learning with our modified training regimen (SASCE). We evaluate our results on two different stories: one a variant of the *Tea for Three* story used by Weyhrauch, and the other the *Anchorhead* story used by Nelson & Mateas; we also perform more detailed analysis on a subset of *Anchorhead*. We use the same plot points, DM actions, and evaluation functions specified and described by Weyhrauch and by Nelson & Mateas.

3.1 SAS+

Weyhrauch's search-based drama manager [12] chooses actions by performing expectimax search, a modification of standard minimax game-tree search in which the "opponent" (in this case the player) is assumed to be acting probabilistically according to the player model rather than adversarially. Since full-depth search in even fairly small stories is intractable, he proposes a sampling search, SAS+. Full-width search is performed to a fixed cutoff depth; from each node at the cutoff, a fixed number of possible plot-point sequences that could follow that node are sampled and scored; and the average of their scores is the evaluation of that node. To avoid dead ends, temporarily-denied plot points are reenabled prior to sampling, but the drama manager otherwise takes no action during sampling.

3.2 TD learning

Temporal-difference learning [9] is a reinforcement-learning technique that, over many thousands of simulated games, estimates the values of story states (a story state is either a partly- or fully-completed story). For completed stories, the state value is simply what the evaluation function says it is; for partial stories, it is the expected value of the complete story (*i.e.* the average of all stories that are continuations of that partial story, weighted according to likelihood). The drama manager's policy is then to simply pick whichever DM action leads to the highest-valued successor state.

A full description of TD learning is beyond the scope of this paper, but the general idea is that as a simulation run progresses, the value of each state encountered is estimated based on the current estimate of its successor. Since at the end of each story we can use the evaluation function to get the real value for the last state, the real values propagate backwards, and all states' estimates should eventually converge to the true values. Since there are far too many possible story states to store the value estimates in a lookup table, we follow Tesauro [10] in training a neural network as a function approximator to estimate a function from states to their values.

In story-based games, the order of plot points is of course important, so the state cannot simply include what has happened, but also the order in which it happened. We encoded state as an $n \times n$ precedence matrix, M, where n is the number of plot points and $M_{i,j}$ indicates whether plot point i

[1]The interested reader is directed to [5] for a more complete review of the drama-management literature (as of 1997).

happened after plot point j.[2] For example, in a story where there are four possible plot points, and in which plot points 3, 4, and 2 have occurred so far (in that order), the precedence matrix would look like:

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

3.3 SASCE

Self-adversarial / self-cooperative exploration (SASCE) is our modified training regimen for TD learning. Our experiments on TD learning both with and without SASCE were identical except for the player model used for training. Typically in machine learning we assume that training data is drawn from the same distribution as the test data. In this domain the TD learner would therefore be trained on a player similar to the one it is expected to encounter when deployed. In our case, this is the player exploring the world.

Instead, the SASCE player model uses the drama manager's current value function to select its actions. Because the SASCE player model uses the agent's state-value estimates, it can choose to act cooperatively or adversarially. A cooperative player model would select actions that put the player in the highest-rated subsequent state, while the adversarial player would move to lower-rated states. The basic idea behind SASCE is to convert the asymmetric non-adversarial drama management problem into a *pseudo-adversarial* or *pseudo-cooperative* problem. This feedback loop is intended to force meaningful exploration of the state space.

We implemented the player model to select states based on an exponential distribution:

$$p(s_i) = \frac{e^{\alpha\beta(s_i)V(s_i)}}{\sum_{s_j \in S} e^{\alpha\beta(s_j)V(s_j)}}.$$

Here s is a state; $V(s)$ is the agent's current state-value estimate; α is a parameter controlling the "desire" of the model to cooperate (large positive values are more strongly cooperative and large negative more adversarial); and $\beta(s)$ is function that responds to the "desires" of the drama manager. In our case, $\beta(s) = 1.0$ for all states except for those the manager has explicitly attempted to increase the likelihood of (*e.g.* by providing a hint). In those cases, $\beta(s)$ is some constant $c_a > 1$ associated with action a.

In practice, we train using a *mixed strategy*. The mixed-strategy player has three parameters. It has both cooperative and adversarial values for α and a probability (q) that the player will use the adversarial value of α. The goal of the mixed strategy is to keep from learning a worst-case policy: Since we don't expect the player to *actually* be acting adversarially, we don't want the policy to optimize only for that case.[3]

[2]This bears some resemblance to the more common bigram encoding, which specifies for each pair of events i, j whether that *sequence* appeared in the history. The precedence matrix trades off some information about exact sequences for a gain in information about global ordering.

[3]The reader may wish to improve upon our results by using our mixed strategy only as a bootstrapping mechanism. Although space limitations preclude a lengthy discussion, an attempt to simulate this by gradually annealing α and q towards an exploring player provided no extra benefit.

	Tea for Three		***Anchorhead* subset**	
Method	Mean	Median	Mean	Median
SAS+	**96.3%**	**96.7%**	42.5%	50.5%
TD	81.2%	81.8%	80.4%	70.5%
TD/SASCE	91.3%	93.3%	**82.8%**	**83.6%**

Table 1: Summary of results using SAS+ search and TD learning with and without SASCE. Numbers are in percentiles relative to the non-drama-managed distribution of story qualities (see text).

4. RESULTS

We first present results on *Tea for Three* and a subset[4] of *Anchorhead*, demonstrating that TD learning outperforms SAS+ on complex domains, and further that using SASCE improves TD learning's performance on both relatively simple and complex domains. We analyze this subset of *Anchorhead* extensively to understand why the policies behave as they do. We then present some results on a larger portion[5] of *Anchorhead* in which none of the policies perform well, but demonstrate that adding more DM actions improves performance markedly, illustrating some potential authorship pitfalls.

4.1 *Tea for Three* and an *Anchorhead* subset

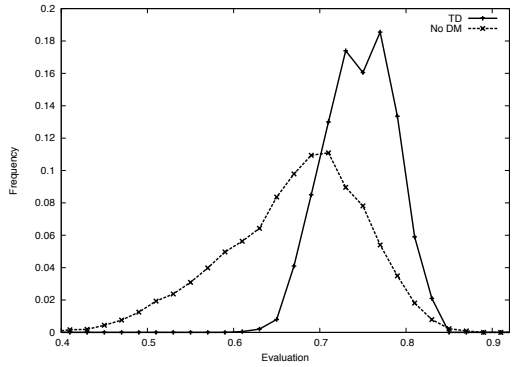
Table 1 summarizes the performance of SAS+ sampling search, TD learning trained conventionally, and TD learning trained using the SASCE training regimen, each evaluated on both *Tea for Three* and a subset of *Anchorhead*. Performance is summarized by calculating the mean and median story quality of each drama-managed distribution, and reporting that in terms of a percentile of the non-drama-managed distribution (*e.g.* a percentile of 70% for the mean would say that the average story quality using drama management is better than 70% of stories without using drama management).

There are three interesting things to note. First and foremost, training TD using SASCE consistently improved its performance by all measures on both stories. Secondly, while search does marginally better than SASCE on the simpler *Tea for Three* story, its performance degrades significantly on the *Anchorhead* subset—which has a more complicated relationship between story structure and evaluation—while the RL methods continue to perform well. Finally, SASCE greatly improves median performance on the *Anchorhead* subset in comparison to conventionally-trained TD learning.

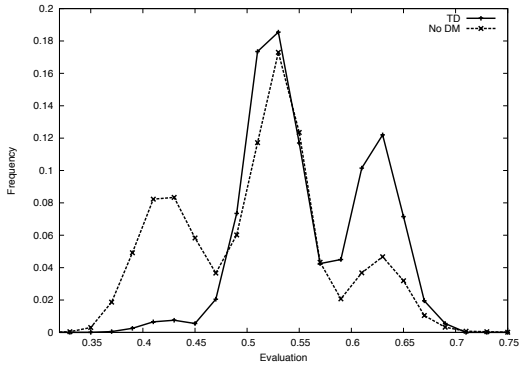
Figure 1 shows the detailed distributions on each story both without drama management and using TD learning without SASCE. Notice in the non-drama-managed distributions that the stories are qualitatively quite different: *Tea for Three* has a relatively smooth unimodal distribution, while the *Anchorhead* subset is distinctly trimodal. This suggests that there are points in the *Anchorhead* subset that have significant influence on the ultimate outcome. It is at these points that SASCE learns to make better decisions.

[4]The subset includes all the plot points from one major subplot, and all DM actions relevant to those plot points.

[5]The larger portion is the one used by Nelson & Mateas [7], itself a subset of the very large original interactive fiction piece.

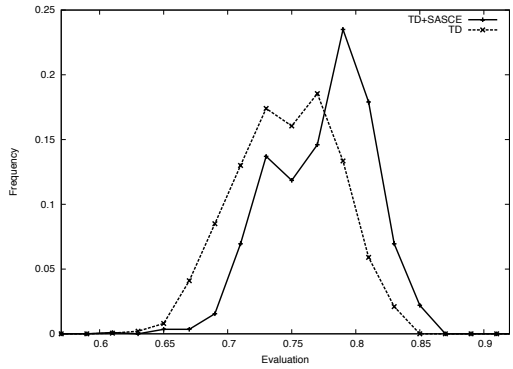


(a) *Tea for Three*

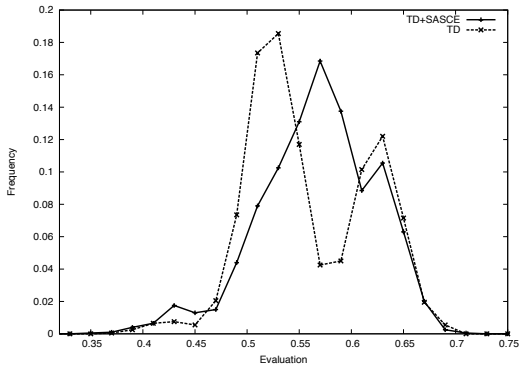


(b) *Anchorhead* subset

Figure 1: Story-quality distributions for conventionally-trained TD learning versus no drama management.



(a) *Tea for Three*



(b) *Anchorhead* subset

Figure 2: Story-quality distributions for TD learning trained using SASCE versus trained conventionally.

This also suggests why search performs so poorly in the *Anchorhead*-subset domain: its relatively short horizon doesn't allow it to effectively make long-term global decisions. As a result it only performs well in "easy" domains where local decisions can result in globally good results, as noted in [7]. We were able to reproduce these results, but also show that reinforcement learning is able to make global decisions in both types of domains effectively.

While not shown in a figure, the story-quality distribution using search on *Anchorhead* remains trimodal like the distribution of randomly-generated stories, lending support to the hypothesis that search fails to find the important decision points that steer between the different types of stories. By contrast, the TD-learned policy's distribution is bimodal and almost entirely missing the stories that form the lowest of the three modes (see Figure 1(b)), indicating that it successfully learns how to avoid the worst batch of stories.

The policy trained with SASCE does even better, with a unimodal distribution that is shifted higher (see Figure 2(b)), indicating that it is not missing any major decision points. To give a more detailed idea of how SASCE improves TD learning, Figure 2 shows the full distributions of story qualities using each method on both stories we evaluated. Remember, qualitatively the goal is to shift the distribution to the right as far as possible, reducing the frequency of lower-scored stories and increasing that of higher-scored stories. Notice how in Figure 2(a), SASCE is consistently better: On the lower side of the distribution (below the mean) its frequencies are lower, while on the higher side they're higher. In Figure 2(b), the improvement takes on a different shape, keeping roughly the same proportion of highly-scored stories, but increasing the quality of the lower- and middle-scored stories significantly, pushing the main peak (and the median) towards the upper end of the range.

Recall that, using SASCE, a player model can be adversarial (negative α), cooperative (positive α), or mixed (choos-

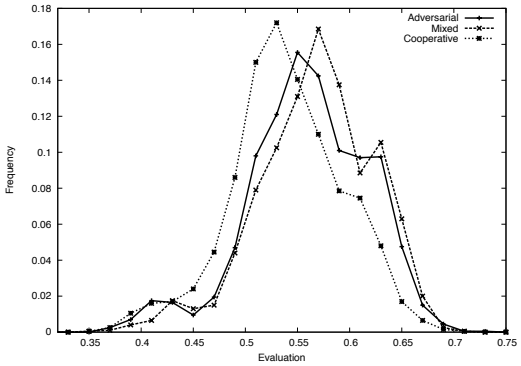


Figure 3: Story-quality distributions on the *Anchorhead* subset using different values of q for training. The three values shown are $q = 0.0$ (cooperative), $q = 0.5$ (mixed), and $q = 1.0$ (adversarial).

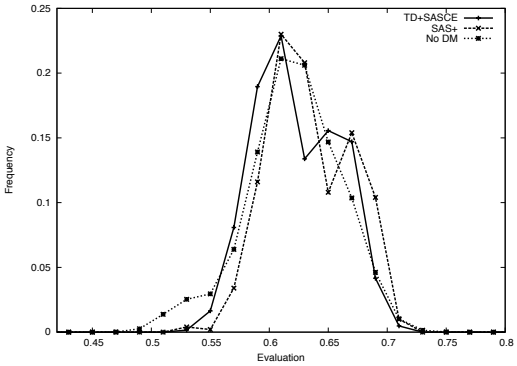


Figure 4: Story-quality distributions on *Anchorhead* for SASCE-trained TD learning, SAS+ search, and no drama management.

Adversarial			Mixed		TD
Mixed	TD	Search	TD	Search	Search
5.6	5.1	1.4	4.8	1.4	1.4

Table 2: Average number of plot points occurring in the *Anchorhead*-subset story before policies disagree on which action to take.

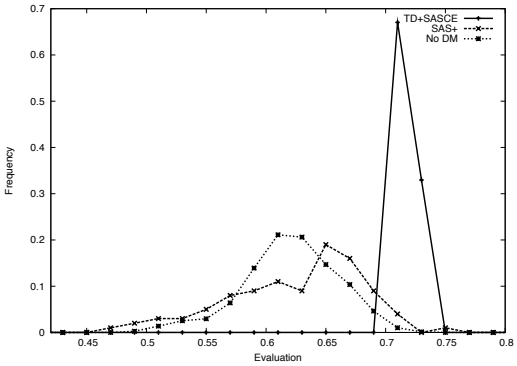


Figure 5: Story-quality distributions on *Anchorhead* with synthetic actions (see text) for SASCE-trained TD learning, SAS+ search, and no drama management.

ing an adversarial player with probability q, and a cooperative player with probability $1 - q$). Interestingly, all mixed models except the fully cooperative player ($q = 0.0$) improved TD learning, both quantitatively and qualitatively. In the examples shown here, as in most of our experiments, training on the mixed player gave the best performance, as shown in Figure 3, although the mean and median are not significantly different. Training on the fully cooperative player consistently gave the worst performance. All types of SASCE players display a roughly unimodal histogram, in stark contrast to the bimodal histogram seen in conventionally-trained TD learning, and the trimodal distribution of the unmanaged stories (Figure 1(b)). This lends some further support to our hypothesis that the modes are due to distinct decision points.

Intuitively, training against an (even occasionally) adversarial player forces the drama manager to learn how to maximize the worst-case performance. That is, the manager learns how to take actions that will prevent the player from arriving in a state where it can then easily do harm to the story. This proves useful for a wide variety of actual player types, including players who are exploring. A cooperative player acts as an antidote for those cases where such states can also lead to remarkably good outcomes.

It is worthwhile to try to understand in more detail how the policies learned by the various methods differ. To that end, we use them simultaneously to guide our simulated player until they reach a point at which they disagree on which action to take. Table 2 shows the average number of plot points occurring prior to disagreement for each pair of policies. Search is making different decisions very early. TD without SASCE diverges from SASCE-trained TD around the fifth plot point. Interestingly, the policies trained with adversarial and mixed players diverge not much later, but have much more similar distributions, suggesting that the important decisions have already been made by the fifth or sixth plot point (on average).

Finally, while space prevents a full discussion, it is worth noting that when there was a disagreement between policies on which action to take, we kept track of the plot point that preceded the disagreement and the pair of conflicting actions. We found that there were a small number of plot points after which most of these disagreements occurred, and the disagreements themselves were mostly between a few pairs of conflicting actions. That suggests that there are likely several key decision points directing stories into one of the three major classes in the trimodal story-quality histogram, and these decision points are where the policies differed most noticeably.

4.2 *Anchorhead*

On the larger portion of the *Anchorhead* story, which in-

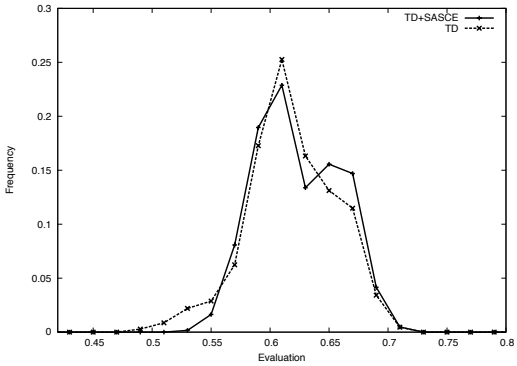


Figure 6: Story-quality distributions on *Anchorhead* for TD learning with and without SASCE.

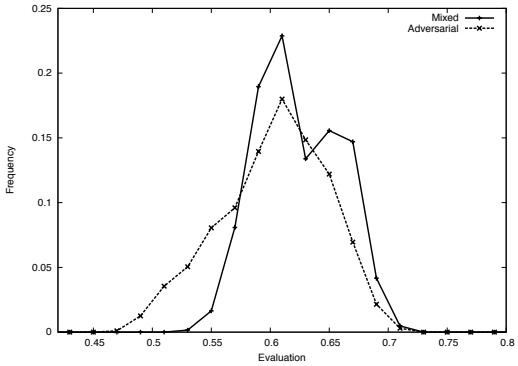


Figure 7: Story-quality distributions on *Anchorhead* for TD learning with mixed-strategy SASCE training ($q = 0.6$ in this case) and completely adversarial SASCE training ($q = 1.0$).

cludes two subplots that may interleave leading to two possible endings, none of the policies do particularly well, as illustrated in Figure 4.

In order to determine whether the policies were performing poorly because of their inability to find a good policy, or because the available DM actions in this state space simply did not permit for a very good policy, we ran a "synthetic" test in which there was a causing, hinting, temp-denying, and reenabling DM action for every plot point ("synthetic" because many of these DM actions could not plausibly be implemented in the real story). This ought to give the drama manager complete control, and indeed as shown in Figure 5, the TD-learned policy is now consistently very good; SAS+ search, however, still does rather poorly, as it is unable to navigate the now even larger state space to make effective use of the added actions. While this does not *prove* that the poor performance without these added actions is due to a lack of sufficient actions, rather than inability to learn a good policy, it does lend some support to that hypothesis, and suggest that the drama manager should be given more actions to work with.

The shape of Figure 5 illustrates a further issue: Although the goal as specified is to maximize story quality, it is not clear that this type of narrow distribution is the sort we actually want, especially since when we examined the plots produced, we found them to be nearly all identical: the drama manager found a very small range of good stories, and used DM actions to cause those stories to always happen. However, we would prefer there be a range of possible stories that could emerge during gameplay, especially if replay value is important. In non-synthetic examples this may turn out to not be an issue most of the time, but it does suggest that perhaps we should extend the evaluation function so that it takes into account the diversity of possible stories as an explicit goal, rather than rating each story in isolation.

Interestingly, SASCE still noticeably improved the performance of TD learning in this case, as illustrated in Figure 6. Furthermore, mixed-strategy training in particular is what improved performance: the completely adversarial version of SASCE did poorly, as shown in Figure 7. These consistent results across all the stories we tested lend some support to our hypotheses that SASCE forces the right type of exploration for this domain, and that a mixed-strategy version of SASCE in particular balances forcing exploration with an adversarial player without learning a policy optimized only for the adversarial case.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a declarative optimization-based drama manager. This style of drama management centrally coordinates the behavior of agents in a game by making requests of them (*DM actions*) in a manner that maximizes story quality according to some author-specified evaluation function. We demonstrated the use of reinforcement learning to learn, ahead of time, policies for taking these actions, rather than performing online search in the time-constrained environment of a real game. We showed that reinforcement learning performs on par with search in simple domains, and far outperforms search in more complex domains. Furthermore, we presented a training method, SASCE, that trains on player models that use the drama manager's current value estimates in such a way as to force exploration useful for this domain; we demonstrated that it improves learning performance in all cases we examined.

One avenue of future work involves a more complete understanding of why training on the sort of player model presented here yields improved performance over training on the same player used for evaluation (a "known correct" player model).

There are other player types in addition to the ones we have explored here. For example, there are players who really want to move through the story quickly, and players who are simply trying to delay ending the game (while an explorer, this type of player is not quite captured by our model). Initial experiments on other players suggest results similar to those presented here, but further investigation is warranted.

Ultimately, we would like to test our learned policies on dozens of actual human players interacting in the concrete world of our chosen stories, and see what distribution this induces in the abstract world of plot point sequences. Unsurprisingly, because players are experiencing concrete events in the story world, what seems like a logical model of player

behavior in the abstract world has the potential to be inaccurate. For example, it is reasonable to expect a player to unlock a door shortly after finding a key; however, if the door is in a dark corner of the room, the player may not see it. In the future we plan to build an agent that represents the player in the concrete world rather than in our abstract plot point world so that we can model these types of influences.

In any case, developing and using a wider range of player models for evaluation would allow us to better understand the relationship between the player model used for training and that used for evaluation. In particular, we are interested in how robust policies are when used with different sorts of players. For example, if we determine that policies trained under varying player models perform better or worse against different player models used for evaluation, we hope to develop techniques to identify in real-time the best policy to use to optimize a given player's experience.

Finally, we are interested in further evaluating the authorial affordances offered by this style of drama management, both in easing the design of large games and in offering possibilities for new types of games difficult or impossible to design without a system of this sort.

Acknowledgments

Work on this project was supported in part by a National Science Foundation Graduate Research Fellowship and a grant from the Intel Foundation. We also wish to thank Victor Bigio for his help coding and running experiments.

6. REFERENCES

[1] J. Bates. Virtual reality, art, and entertainment. *Presence: The Journal of Teleoperators and Virtual Environments*, 2(1):133–138, 1992.

[2] A. Lamstein and M. Mateas. A search-based drama manager. In *Proceedings of the AAAI-04 Workshop on Challenges in Game AI*, 2004.

[3] B. Laurel. *Toward the Design of a Computer-Based Interactive Fantasy System*. PhD thesis, Drama department, Ohio State University, 1986.

[4] B. Magerko. Story representation and interactive drama. In *Proceedings of the First Annual Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-05)*, 2005.

[5] M. Mateas. An Oz-centric review of interactive drama and believable agents. In M. Woodridge and M. Veloso, editors, *AI Today: Recent Trends and Developments. Lecture Notes in AI 1600*. Springer, Berlin, NY, 1999. First appeared in 1997 as Technical Report CMU-CS-97-156, Computer Science Department, Carnegie Mellon University.

[6] M. Mateas and A. Stern. Integrating plot, character, and natural language processing in the interactive drama Façade. In *Proceedings of the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE-03)*, 2003.

[7] M. J. Nelson and M. Mateas. Search-based drama management in the interactive fiction Anchorhead. In *Proceedings of the First Annual Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-05)*, 2005.

[8] J. B. Pollack and A. D. Blair. Why did TD-Gammon work? *Advances in Neural Information Processing Systems*, 9:10–16, 1997.

[9] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[10] G. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8:257–277, 1992.

[11] G. Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[12] P. Weyhrauch. *Guiding Interactive Drama*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1997. Technical Report CMU-CS-97-109.

[13] R. M. Young, M. O. Riedl, M. Branly, A. Jhala, R. J. Martin, and C. J. Saretto. An architecture for integrating plan-based behavior generation with interactive game environments. *Journal of Game Development*, 1(1), 2004.

Learning to Cooperate in Multi-Agent Social Dilemmas

Enrique Munoz de Cote
Politecnico di Milano
Department of Electronics and Information
piazza Leonardo da Vinci 32, I-20133 Milan, Italy
munoz@elet.polimi.it

Alessandro Lazaric
Politecnico di Milano
Department of Electronics and Information
piazza Leonardo da Vinci 32, I-20133 Milan, Italy
lazaric@elet.polimi.it

Marcello Restelli
Politecnico di Milano
Department of Electronics and Information
piazza Leonardo da Vinci 32, I-20133 Milan, Italy
restelli@elet.polimi.it

ABSTRACT

In many Multi-Agent Systems (MAS), self-interested agents need to cooperate in order to maximize their own utilities in time. The goal of this work is to improve cooperation among agents that use best-response Reinforcement Learning (RL) algorithms (Q-Learning), by the introduction of two new principles (*Change or Learn Fast* and *Change and Keep*) that foster the reaching of Pareto efficient stable outcomes.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

1. INTRODUCTION

In this paper we focus on problems of cooperation among *self-interested agents*, where no explicit communication is possible and agents can only perceive the actions taken by the other agents at the previous time step.

Several algorithms [5, 2, 3] have been proposed to make agents able to learn Pareto efficient (PE) solutions.

In Section 2, we introduce two new principles to foster cooperation. Section 3 presents comparative results in a multi-agent social dilemma and Section 4 contains conclusions and future research directions.

2. MAKING Q-LEARNING COOPERATIVE

A matrix game is a tuple $\langle \mathcal{N}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, \{R_i\}_{i\in\mathcal{N}} \rangle$, where $\mathcal{N}$ is a collection of n agents, $\mathcal{A}_i$ is the set of actions available to agent i, and R_i is its payoff matrix. At each iteration t, let $\mathbf{a}^t = [a_1^t, a_2^t, ..., a_n^t]$ be the executed joint action, a_i^t the action taken by the i-th agent and $\mathbf{a}_{-i}^t$ the joint action of all the players but i, r_i^t is the reward received by i-th agent.

In this paper, we adopt a version of Q-learning where states are represented by the previous joint action ($\mathbf{a}^{t-1}$).

The desired outcome of our algorithms is to let a MAS of Q-learners collectively learn equilibrium points that payoff dominate the best response dynamics of normal Q-learners.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

The main problem of pairing best-response agents in a repeated game is that they may end up in cyclic or suboptimal behaviors. In fact, many parallel learning processes may cause the environment to be non-stationary, thus preventing agents from predicting the correct outcome of their actions. In the following we introduce two variants of the Q-learning algorithm, aimed at improving cooperation of self-interested non-communicating Q-learning agents by taking into account the non-stationarity induced by the learning processes.

2.1 CoLF Principle

Algorithm 1 COLF – Change Or Learn Fast

Let $\alpha_S > \alpha_{NS}$, and λ be learning rates
$P(\mathbf{a}, a_i) \leftarrow S(\mathbf{a}, a_i) \leftarrow 0, Q(\mathbf{a}, a_i) \leftarrow \frac{r_{max}}{1-\gamma} \forall \mathbf{a} \in \mathcal{A}, a_i \in \mathcal{A}_i$
choose a random action a_i^0
execute a_i^0
read the joint action $\mathbf{a}^0$
$t \leftarrow 1$
for all steps **do**
  choose action a_i^t according to exploration strategy
  execute a_i^t and get the payoff r_i^t
  read the joint action $\mathbf{a}^t$
  $\Delta r_i^t \leftarrow |r_i^t - P(\mathbf{a}^{t-1}, a_i^t)|$
  if $\Delta r_i^t > S(\mathbf{a}^{t-1}, a_i^t)$ **then**
    $\alpha \leftarrow \alpha_{NS}$
  else
    $\alpha \leftarrow \alpha_S$
  end if
  $Q(\mathbf{a}^{t-1}, a_i^t) \leftarrow (1-\alpha) Q(\mathbf{a}^{t-1}, a_i^t) + \alpha \cdot \left(r_i^t + \gamma \cdot \max_{a_i} Q(\mathbf{a}^t, a_i)\right)$
  $S(\mathbf{a}^{t-1}, a_i^t) \leftarrow (1-\lambda) S(\mathbf{a}^{t-1}, a_i^t) + \lambda \cdot \Delta r_i^t$
  $P(\mathbf{a}^{t-1}, a_i^t) \leftarrow (1-\lambda) P(\mathbf{a}^{t-1}, a_i^t) + \lambda \cdot r_i^t$
  $t \leftarrow t+1$
end for

The CoLF (Change or Learn Fast) principle is inspired by the work of Bowling and Veloso [1], where a variable learning rate is considered. To foster cooperation, we propose to modify the learning rate of a Q-learning algorithm according to the following rule: if the payoff achieved by an agent is unexpectedly changing, then learn slowly, otherwise learn quickly. This principle aids in cooperation by giving less importance to "unexpected" payoffs, while allowing to speed up learning when the most of the agents are playing near-stationary strategies.

Algorithm 1 summarizes how the Q-learning algorithm changes with the introduction of the CoLF principle. The P-values are exponential averages of the collected payoffs with weight factor λ, while the S-values are exponential averages of the absolute differences between the current payoff and the respective P-value.

2.2 Change&Keep Principle

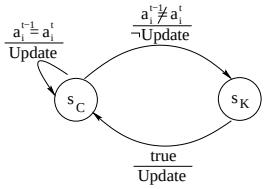


Figure 1: CK finite-state machine. For each state transition are reported both the trigger condition (above the line) and whether the update phase occurs or not (below the line). In state s_C the agent performs the usual action selection, while in state s_K it repeats the previous action.

The Change&Keep (CK) principle is based on the following observation: when an agent, due to either learning or exploration, decides to choose a different action, it typically collects an uninformative payoff. In fact, these (non-stationary) changes cannot be foreseen by other agents, and the related payoffs may be misleading, thus negatively affecting cooperation.

The idea of the CK principle is to discard the payoff received in correspondence to a change in the action selection (thus suspending the Q-value update), repeat the same action, and use the corresponding payoff for performing the suspended update. In this way the agent gives time for the other agents to react to its new action, thus using a more informative payoff for the update of its Q-table. Fig. 1 illustrates the two-states finite-state machine that describes the CK principle.

3. EXPERIMENTAL RESULTS

In this section, we compare the performance (in self-play) of Q-learning with those obtained by our variants (CK, CoLF, and CK-CoLF, which is obtained by combining both CK and CoLF principles together) in the MASD game [5]. In MASD, N agents hold M resource units each. At each iteration, let $a_i \in [0, M]$ be the amount of units contributed by agent i towards a group goal G. The utility of agent i given the joint action $\mathbf{a}$ is $P_i(\mathbf{a}) = \frac{\left[\frac{1}{N}\sum_{j=1}^{N} a_j\right] - k a_i}{M(1-k)}$, where $k \in \left(\frac{1}{N}; 1\right)$ is a constant that indicates how much each agent estimates its contribution towards the selfish goal. The payoff function is such that when all the agents put M units in the group goal, each agent is rewarded with 1 and when nobody puts units in the group goal, a payoff of 0 is produced. If each agent adopts a random strategy the expected average payoff is 0.5. All the algorithms used an ϵ-greedy exploration (where $\epsilon = \max(0.2 - 0.00006t, 0)$) joined with a relaxation search (as proposed in [5]) obtained by setting the initial values of the Q-table to high values.

In the learning rate performance comparison we used the following parametrization: $\gamma = 0.95$, and, for the CoLF principle, $\lambda = 0.1$. In Fig. 2 we report the results with different learning rates. The use of low learning rates in Q-learning (see Fig. 2(a)) allows to get higher payoffs but, as a drawback, the time required to reach a cooperative solution considerably increases. The results of CK (Fig. 2(b)) with the same learning rates show that it converges almost in the same time as Q-learning but the learning curve is steeper and outperforms Q-learning for every α. In CoLF, we adopted $\alpha_{NS} = 0.1$ and $\alpha_S = 4 \cdot \alpha_{NS}$. As can be noticed in Fig. 2(c), the CoLF principle makes agents able to achieve higher payoffs than those obtained by Q-learning with constant learning rate. The algorithm succeeded in exploiting both the low learning rate in terms of convergence payoff performance and the high learning rate in terms of learning speed. Finally, Fig. 2(d) show how the introduction of the CoLF principle into the CK algorithm, improves the learning speed, while still reaching the PE solution.

We have experimentally verified (see Fig. 3) that high discount factors allow to achieve better performances, but with a lower learning speed. In fact, using 0.8 as discount factor, Q-learning and CoLF are not able to achieve a cooperative solution, which is obtained by adding the CK principle. On the other hand, all the algorithms achieve a cooperative solution, but they require much longer time to converge. We have also tested the performances of the four algorithms in different versions of the MASD problem obtained by varying the k factor. For higher values of k the agents become more selfish and the reaching of the cooperative solution is more difficult. The sensitivity analysis in Fig. 4 shows the performances of the four algorithms with different values of k. None of the four algorithms is able to cooperate with $k = 0.9$ since the selfish behavior is highly fostered. The problem becomes easier and easier as k decreases.

4. DISCUSSION

As shown by the previous experiments, the CoLF principle is mainly concerned with non-stationarity caused by the other agents. It learns slow when gathering unexpected payoffs and fast otherwise. On the other hand, the CK principle deals with non-stationarity induced in the MAS by the behavior of the agent itself. Since actions selected following a non-stationary learning policy may not be foreseen by other agents, whenever an agent changes its action, according to the CK principle, it does not perform the update and repeats the same action and updates the related value with the last received payoff. The experiments carried out in the MASD framework show that the proposed principles applied to a best-response learning algorithm (Q-learning) largely improve its cooperation capabilities in self-play, both in terms of performance and learning speed.

5. ADDITIONAL AUTHORS

Additional authors: Andrea Bonarini (Politecnico di Milano, email: `bonarini@elet.polimi.it`).

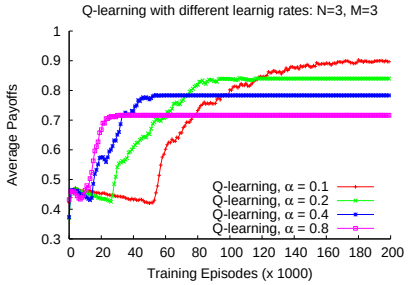
6. REFERENCES

[1] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.

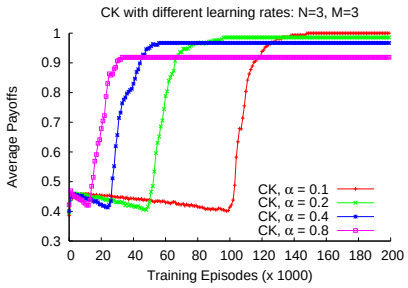
[2] J. W. Crandall and M. A. Goodrich. Learning to compete, compromise, and cooperate in repeated general-sum games. In *Proc. of ICML 2005*, to appear.

[3] M. W. Macy and A. Flache. Learning dynamics in social dilemmas. *Proceedings of the National Academy of Sciences*, 72(9):29–36, 2002.

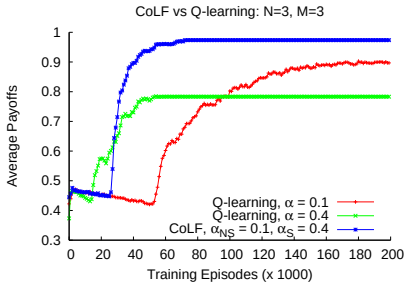
[4] J. L. Stimpson and M. A. Goodrich. Learning to cooperate in a social dilemma: A satisficing approach to bargaining. In *Proceedings of ICML*, 2003.



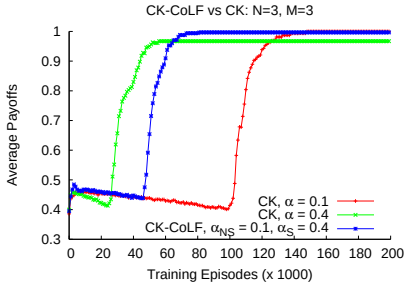
(a)



(b)

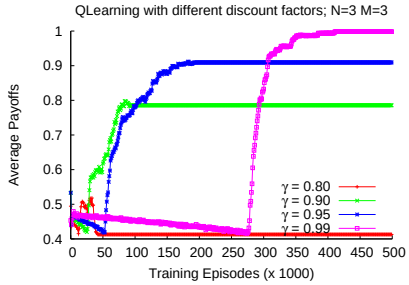


(c)

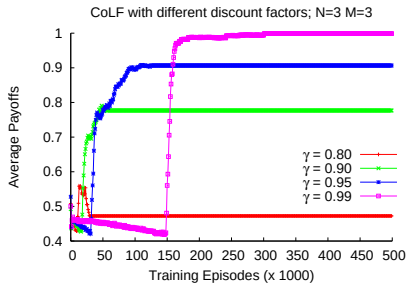


(d)

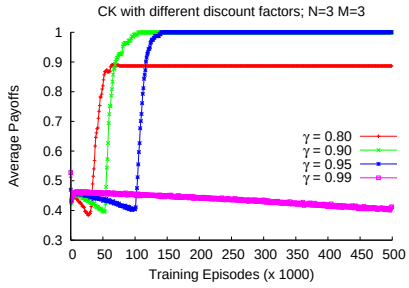
Figure 2: Plots showing the effect of different learning rates in the MASD problem with 3 agents, 4 actions, and $k = 2/3$.



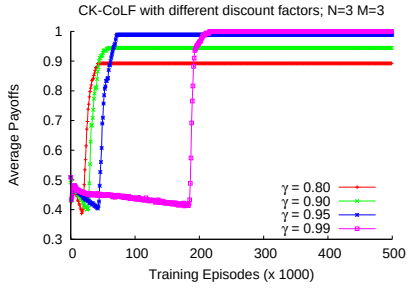
(a)



(b)

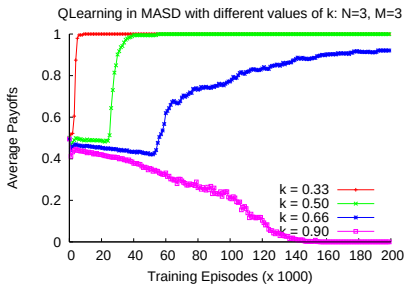


(c)

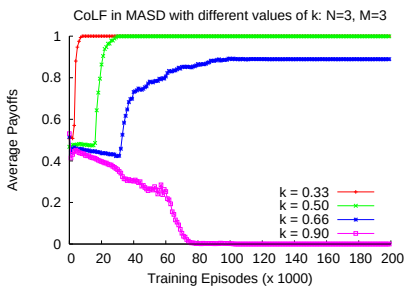


(d)

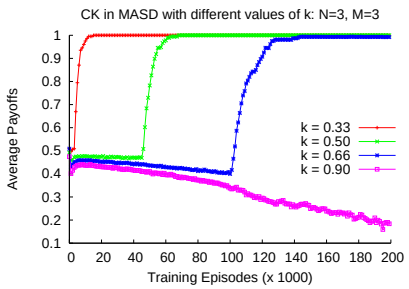
Figure 3: Plots showing the effect of different discount factors in the MASD problem with 3 agents, 4 actions, and $k = 2/3$.



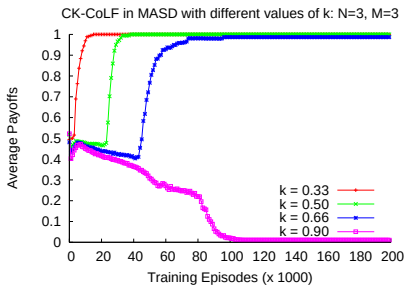
(a)



(b)



(c)



(d)

Figure 4: Plots showing the effect of different values of k in the MASD problem with 3 agents, 4 actions.

[5] J. L. Stimpson, M. A. Goodrich, and L. C. Walters. Satisficing and learning cooperation in the prisoner's dilemma. In *Proc. of IJCAI 2001*, 2001.

An Adaptive Plan-Based Dialogue Agent: Integrating Learning into a BDI Architecture

Anh Nguyen
School of Computer Science and Engineering
University of New South Wales
Sydney NSW 2052, Australia
anht@cse.unsw.edu.au

Wayne Wobcke
School of Computer Science and Engineering
University of New South Wales
Sydney NSW 2052, Australia
wobcke@cse.unsw.edu.au

ABSTRACT

We consider the problem of dialogue adaptation in our Smart Personal Assistant (SPA), which uses a plan-based dialogue model. We present a novel way of integrating learning into a BDI architecture so the agent can learn to select the most suitable plan amongst those applicable, enabling the SPA to tailor its responses according to the conversational context and the user's physical context, device and preferences.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*natural language interfaces*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*intelligent agents*

General Terms

Human Factors

Keywords

BDI agents, dialogue management, adaptive dialogue systems, symbolic machine learning

1. INTRODUCTION

The Smart Personal Assistant (SPA) [9] is an application for desktops and PDAs that allows users to conduct spoken dialogues with the system to remotely access and manipulate their e-mail and calendar. The SPA consists of a dialogue manager agent and a collection of task assistants, each specializing in a particular task domain. The dialogue manager [5] is implemented using a BDI agent architecture, which has a set of plans for processing user requests. This BDI approach allows a high level of abstraction in designing the dialogue model, which consists of complex but modular plans, each associated with a communicative goal or dialogue aspect. There are also plans for handling domain-specific tasks. The control flow of the dialogue is not specified directly, but derived automatically using the plan selection mechanism as part of the BDI interpretation cycle. This approach preserves modularity within the dialogue model at the level of plans, allowing domain-independent discourse-level plans to be reused for other domains. The belief state of the dialogue manager contains the past and current states of the dialogue as a stack of conversational acts performed by the user, the dialogue manager and other task assistants. These conversational acts constitute a hierarchy of dialogue segments, each having a segment purpose which represents a task-specific intention of the user.

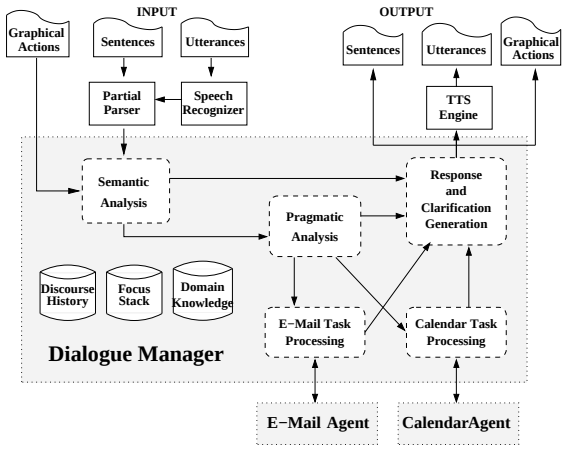


Figure 1: Dialogue manager's plans and beliefs

The dialogue manager has about forty plans, which can be roughly grouped into five classes according to their function, as in Figure 1. The first group, *semantic analysis*, is for classifying the user's requests into a task domain and formulating its semantic representation using the agent's linguistic knowledge. The second group, *pragmatic analysis*, is for determining the user's underlying conversational act and associated task-specific intention, which may involve resolving references. The *e-mail* and *calendar task processing* groups contain domain-specific plans for performing domain tasks, which requires interaction with the other task assistants. The last group is for generating responses to the user, including system requests for clarification.

In this paper, we consider the problem of dialogue adaptation in the SPA. The motivation is that the usability of the SPA on a mobile device with small screen and limited band-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

width can be enhanced by adaptively tailoring responses according to the conversational context and the user's physical context, device and preferences. For example, e-mail messages are often too long to fit on one PDA screen, thus showing only a message summary could sometimes be more desirable. Similarly, in different conversational contexts, the same information might be presented to the user in different ways, e.g. using a different modality, format or layout. Because the SPA's dialogue manager uses plans for generating responses, the problem of dialogue adaptation becomes the problem of learning which dialogue plan should be selected and executed in a given context.

There has been comparatively little work in the area of dialogue adaptation; existing work tends to be concerned with either adapting the system's questions for eliciting user preferences [7] or learning strategies in a game-theoretic approach to dialogue modelling such as the Email Voice Interactive System (ELVIS) by Walker *et al.* [8] and the work by Jokinen *et al.* in the Interact project [3]. In the game-theoretic approach, training dialogues are used to build an adaptivity model using reinforcement learning. However, the dialogue models are less complex than that of the SPA.

There have been very few attempts to incorporate learning within BDI models of agency [1, 6], of which none has looked at the problem of how the agent learns to select the most appropriate plan amongst those applicable in a given context. Typically, this work provides a loose coupling of the agent architecture and the learner, while for the problem of learning plan selection, a closer integration is required.

2. AN ADAPTIVE DIALOGUE AGENT

To build an adaptive plan-based dialogue manager, there are two problems to be solved: (i) defining the possible system actions in any given situation, and (ii) defining the mechanism for the system to decide when to choose which action. Using the plan-based approach to dialogue management, system modularity is facilitated through the use of plans, each handling some dialogue aspect or domain task. Hence the possible choices of system action can be modelled as different plans applicable in the same situation, and the learner is used for the selection mechanism to choose the most suitable plan according to the agent's beliefs, which represent the user's context and/or preferences. Thus the learner accepts as input a set of possible plans and some of the agent's beliefs and outputs the most suitable plan(s).

The dialogue manager of the SPA is implemented using a Java-based PRS-style agent platform, JACK Intelligent Agents^{TM} [2]. JACK provides meta-level reasoning, posting a special `PlanChoice` event, as a way to overwrite the agent's plan selection mechanism, making the integration with learning easier. We have chosen to integrate the Alkemy symbolic decision tree learner [4] into the dialogue manager to develop an adaptive dialogue agent. In Alkemy, the learning individuals are represented in a typed, higher-order logic, which supports representation of domain data with complex structure. Moreover, Alkemy provides a highly-expressive predicate rewrite system to constrain the hypothesis space for the learner.

Figure 2 shows the learning problem for the meta plan to decide which plan to use to display a set of e-mail messages to the user. Decision tree learners are not able to learn a set of values, and in our learning problem it is necessary to predict more than one possible plan, so given the current

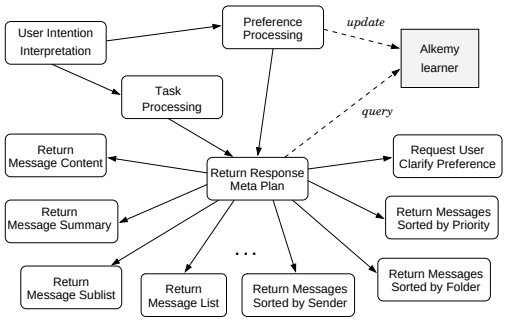


Figure 2: Example learning plan selection

context, for each available `Return Response` plan, a learning individual is generated that includes information about the current context and the plan name. This is classified as either `True` or `False`. If the result is `True`, the plan is considered an option; otherwise, the plan does not meet the user's preferences is not an option. If there is more than one plan classified `True`, they are all options, and in this case the `Request User Clarify Preference` plan is executed, asking the user to specify the preferred system action (i.e. the most preferred plan) and then updating the learner accordingly. The Alkemy specification for this learning problem includes the following data constructors:

$PDA,\ Desktop : Device\ ;$
$Speech,\ Text,\ None : Mode\ ;$
$Search,\ Read,\ Show,\ ...,\ Notify : Task\ ;$
$Low,\ High,\ Normal : Priority\ ;$
$Length = Int\ ;$
$Folder = String\ ;$
$ReturnMsgList,\ ...,\ ReturnMsgSortedBySender : Plan\ ;$
$True,\ False : Class\ ;$

The `Mode` element indicates the current modality of the dialogue, where `Text` corresponds to typed text and `None` means the user is currently using only the GUI. The dialogue result included in the learning individual includes a set of e-mail messages. Each message is represented by four components: the sender, the length, the folder and the priority. Thus each learning individual (training example) is of the `Product` type defined as follows:

$$Ind = Device \times Task \times Mode \times (Set\ Email) \times Plan;$$

Each learning example is pre-computed and sent as input to the learner. The individuals include information about the current context, e.g. the modality of interaction, the device, etc., the result set of e-mail messages and the name of the plan to be considered. The learning problem is to predict whether a plan is an option in the current context according to the user's preferences. Thus the function that needs to be learned is $ResponsePlan : Ind \rightarrow Class$.

Alkemy uses a predicate rewrite system for constructing the hypothesis space. Predicates are composed from basic functions called transformations. For example, a transformation on `Email` to extract its length is a projection of its second component as follows:

$$projLength : Email \rightarrow Int;$$
$$projLength : project(1);$$

The following predicate is true iff the e-mail set contains exactly one message whose length is less than thirty lines:

$$and\ (projEmails \circ numOfEmails(true) \circ eq1)$$
$$(projEmails \circ setEmailExists(projLength \circ lt30))$$

A part of the predicate rewrite system for our learning

problem is shown in Figure 3. The predicates are constructed by applying the predicate rewrites to expand the predicate tree starting from the root. For example, replacing *top* in (1) by (7) results in a predicate which is true iff the user's device is a PDA. Predicate rewrite (3) is useful when there is no e-mail message in the result. Expanding (4) or (5) results in predicates for checking the length or the priority of the only e-mail message in the result. The rule (6) is used for constraining the total number of messages. Note that defining this predicate rewrite system requires the expression of domain-specific knowledge. Furthermore, constraining the learner's hypothesis space is necessary because of the limited availability of training data.

(1) $top \rightarrowtail projDevice.top$;
(2) $top \rightarrowtail projPlan.top$;
(3) $top \rightarrowtail and\ (projEmails.numOfEmails(true).eq0)(top)$
(4) $top \rightarrowtail and\ (projEmails.numOfEmails(true).eq1)$
$(projEmails.setEmailExists1(projLength.top))$;
(5) $top \rightarrowtail and\ (projEmails.numOfEmails(true).eq1)$
$(projEmails.setEmailExists1$
$(projPriority.top))$;
(6) $top \rightarrowtail projEmails.numOfEmails(true).top$;
(7) $top \rightarrowtail eqDevicePDA$;
(8) $top \rightarrowtail eqPriorityHIGH$;
(9) $top \rightarrowtail lt30$;

Figure 3: The predicate rewrite system

We first show an example dialogue without adaptation:

```
User   Is there any new mail from Wayne?
SPA    You have one new message from Wayne Wobcke.
SPA    Displays the message from Wayne Wobcke.
User   Find all messages about meeting in the Inbox.
SPA    There are twenty messages about meeting in your Inbox.
       Displays the headers of the twenty messages.
User   Show me the one from John.
SPA    Here you are.
       Shows the content of the message from John Lloyd.
```

Now assume that the user is using a PDA for interacting with the SPA, with dialogue adaption enabled; the example dialogue above would become:

```
User   Is there any new mail from Wayne?
SPA    You have one new message from Wayne Wobcke.
       The message is more than thirty lines, should I
       show you just the summary?
User   Yes please.
SPA    Displays the summary of message from Wayne Wobcke.
       SPA learns to display only the summary
       if the message length is more than 30 lines.
User   Find all messages about meetings in the Inbox.
SPA    SPA has learned in the past to show only the first few
       messages if there are too many messages in the result.
       The message headers should also be sorted by sender.
       I am showing the first ten of twenty messages
       sorted by sender.
       Displays the first 10 message headers sorted by sender.
User   Show me the one from John.
SPA    Here is the summary of the message from John Lloyd.
       SPA shows user the summary of the message according
       the user's preferences that have been learned so far.
```

Note from the dialogue above how the dialogue manager has learned to adapt its responses according to the user's preferences. When the user requests the summary of the message from Wayne, the system has learned to show only the summary if the message body is more than thirty lines. Hence later on in the dialogue, the system immediately shows the summary of the e-mail message from John Lloyd because its length is also more than thirty lines. A typical example of a decision list representing the user's preferences that results in such adaptive behaviour is as follows:

```
if numOfEmail is 0 then
  if plan is ReturnResponse then True
else if numOfEmail is 1 and length is greater than 30 then
      if plan is ReturnSummaryContent then True
      else if numOfEmail is 1 and priority is HIGH then False
    else
      if numOfEmail is less than 15 then
        if plan is ReturnMessageSortedBySender then True
        else if plan is ReturnSubList then True
```

3. CONCLUSION

We have proposed a way of incorporating learning into a BDI agent architecture that allows the agent to learn the appropriate plan to select in a given context. The technique can be applied to plan-based dialogue systems to achieve dialogue adaptation, which is particularly useful in dialogue-based applications for mobile devices, where the interaction should be tailored to suit the current conversational context, the physical environment and the user's device and preferences.

4. ACKNOWLEDGEMENTS

This work was funded by the CRC for Smart Internet Technology. We would like to thank Agent Oriented Software Pty. Ltd. for the use of their JACK agents platform. Many thanks to Kee Siong Ng and Joshua Cole for supplying Alkemy and helping us resolve technical problems arising in the integration of Alkemy and the SPA.

5. REFERENCES

[1] A. Guerra-Hernández, A. E. Fallah-Seghrouchni and H. Soldano. Learning in BDI Multi-agent Systems. In *CLIMA IV*, pages 218–233. Springer-Verlag, Berlin, 2004.

[2] N. Howden, R. Rönnquist, A. Hodgson and A. Lucas. JACK Intelligent Agents™– Summary of an Agent Infrastructure. *Paper presented at the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, 2001.

[3] K. Jokinen, A. Kerminen, M. Kaipainen, T. Jauhiainen, G. Wilcock, M. Turunen, J. Hakulinen, J. Kuusisto and K. Lagus. Adaptive Dialogue Systems - Interaction with Interact. In *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialogue*, 2002.

[4] J. W. Lloyd. *Logic for Learning: Learning Comprehensible Theories from Structured Data*. Springer-Verlag, Berlin, 2003.

[5] A. Nguyen and W. Wobcke. An Agent-Based Approach to Dialogue Management in Personal Assistants. In *Proceedings of the 2005 International Conference on Intelligent User Interfaces*, pages 137–144, 2005.

[6] T. Phung, M. Winikoff and L. Padgham. Learning within the BDI Framework: An Empirical Analysis. In R. Khosla, R. J. Howlett and L. C. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems, Part III*, pages 282–288. Springer-Verlag, Berlin, 2005.

[7] C. A. Thompson, M. H. Göker and P. Langley. A Personalized System for Conversational Recommendations. *Journal of Artificial Intelligence Research*, 21:393–428, 2004.

[8] M. A. Walker. An Application of Reinforcement Learning to Dialogue Strategy Selection in a Spoken Dialogue System for Email. *Journal of Artificial Intelligence Research*, 12:387–416, 2000.

[9] W. Wobcke, V. Ho, A. Nguyen and A. Krzywicki. A BDI Agent Architecture for Dialogue Modelling and Coordination in a Smart Personal Assistant. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 323–329, 2005.

Multi-agent Reinforcement Learning Algorithm to Handle Beliefs of Other Agents' Policies and Embedded Beliefs

Takaki Makino
Dept. of Frontier Sciences and Science Integration, the University of Tokyo
5-1-5 Kashiwa-no-ha,
Kashiwa, Chiba, 277-8568 Japan
mak@cb.k.u-tokyo.ac.jp

Kazuyuki Aihara
Institute of Industrial Science, the University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo 153-8505 Japan
ERATO Aihara Complexity Modelling Project, JST
3-23-5 Uehara, Shibuya-ku, Tokyo 151-0064 Japan
aihara@sat.t.u-tokyo.ac.jp

ABSTRACT

We have developed a new series of multi-agent reinforcement learning algorithms that choose a policy based on beliefs about co-players' policies. The algorithms are applicable to situations where a state is fully observable by the agents, but there is no limit on the number of players. Some of the algorithms employ embedded beliefs to handle the cases that co-players are also choosing a policy based on their beliefs of others' policies. Simulation experiments on Iterated Prisoners' Dilemma games show that the algorithms using on policy-based belief converge to highly mutually-cooperative behavior, unlike the existing algorithms based on action-based belief.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms, Theory

Keywords

multi-agent reinforcement learning, policy-based belief, embedded belief, Iterated Prisoners' Dilemma

1. INTRODUCTION

Each member's effort to maximize its own benefit constitutes social behaviors, such as competition and cooperation. One promising way to implement such behaviors on computers is to extend a reinforcement learning algorithm into multi-agent environments, in order to provide independent trial-and-error learning for each agent. If the agents are smart enough, such social behaviors would emerge spontaneously.

Nevertheless, past studies on N-agent reinforcement learning have limited to specific structure of games [1, 2, 3]. Such limitations are required due to the way to extend reinforcement learning framework. These studies rely on the following update rule, which is the same as the classical TD-learning except action is replaced by a vector of actions:

$$Q_i(s(t),\vec{a}(t)) \leftarrow (1-\alpha)Q_i(s(t),\vec{a}(t)) + \alpha\big[r_i(t)+\gamma V_i(s(t+1))\big], \quad (1)$$

where $s(t)$ is the state at time t, $\vec{a}(t) = (a_1(t),\ldots,a_N(t))$ is the vector of actions for all players taken at time t, $Q_i(s,\vec{a})$ and $V_i(s)$ are the expected action-state value and state value for agent i ($1 \le i \le N$), $r_i(t)$ is the reward given to agent i as a result of actions $\vec{a}(t)$, α is the learning rate, and γ is the discount factor.

A Major problem of this formulation is that it cannot deal with learning co-player agents. Equation 1 assumes that the expected value $V_i(s)$ for agent i only depends on the current state s [6]. This is equivalent to assume stateless co-players, i.e., the states of other agents are independent of the expected value. Learning agents change its behavior based on the learned information; it is hard to find the best policy against such learning co-players under the assumption of statelessness.

Recently, Weinberg and Rosenschein have tried to break this situation by proposing the NSCP-learner [7] that finds the best-response policy to other players' policies. In their algorithm, agent i maintains a model $\langle\pi_j\rangle_i$ of another agent j ($1 \le j \le N, j \ne i$)[1], and calculates V_i by using the models:

$$V_i^{\pi_i,\langle\vec{\pi}_i\rangle_i}(s) = \max_{\pi_i} \sum_{a_1,\ldots,a_N} \pi_i(s,a_i) \cdot \prod_{1\le j\le N, j\ne i} \langle\pi_j\rangle_i(s,a_j) \cdot Q_i(s,\vec{a}), \quad (2)$$

where the policy $\pi_i(s,a_i)$ is the probability of agent i's taking action a_i on condition of state s and $\langle\vec{\pi}_i\rangle_i$ is the vector of beliefs about policies of the co-player agents for agent i, i.e., $\langle\vec{\pi}_i\rangle_i = (\langle\pi_1\rangle_i,\ldots,\langle\pi_{i-1}\rangle_i,\langle\pi_{i+1}\rangle_i,\ldots,\langle\pi_N\rangle_i)$. In short, the agent uses models to anticipate the actions of the co-players and calculates the state value using the belief on their expected actions.

However, we'd like to point out that Weinberg's extension is not sufficient. Suppose two possible beliefs $\langle\pi_j\rangle_i, \langle\pi'_j\rangle_i$ about policies of agent j, which choose the same action a_j for the immediate response to state s but different actions afterwards. Then the state values for these policies, $V_i^{(\ldots,\langle\pi_j\rangle_i,\ldots)}(s)$ and $V_i^{(\ldots,\langle\pi'_j\rangle_i,\ldots)}(s)$ should be able to be different, whereas eq. 2 cannot.

In this paper, we propose a new way to extend the action-value function Q with the vector of *policies* to solve this problem. Although an agent cannot know the current policies of the co-player agents, one can develop a *belief* of policies of the co-player agents and use it for the extended action-value function. This extension also lets us to introduce an embedded policy belief, i.e., a belief

[1]Weinberg *et al.* used the notation $\hat{\pi}^j$, but in this paper, we use the angle-bracket notation to clarify whose model is on whose policy.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Initialize $Q_i(s,a)$
Repeat (for each episode):
  Observe the initial state s
  Repeat (for each step):
    Derive π_i from Q_i
    Choose a_i from π_i and s
    Take action a_i, observe $\vec{a}, r, s'$
    $Q_i(s,a) \leftarrow (1-\alpha)Q_i(s,a) + \alpha\left[r + \gamma \max_{a_i'} Q_i(s',a_i')\right]$
    $s \leftarrow s'$;
  until s reaches the end of the episode

Figure 1: Algorithm of a Level-0 agent of index i (equivalent to the original Q-learning algorithm).

Initialize $Q_i(\langle\vec{\pi_i}\rangle_i, s, a)$ and $\langle\vec{\pi_i}\rangle_i$
Repeat (for each episode):
  Observe the initial state s
  Repeat (for each step of episode):
    Derive Π_i from Q_i
    $\pi_i = \Pi_i\left(\langle\vec{\pi_i}\rangle_i\right)$
    Choose a_i from π_i and s
    Take action a_i, observe $\vec{a}, r, s'$
    $\langle\vec{\pi_i}\rangle_i' \leftarrow$ New belief, obtained by updating $\langle\vec{\pi_i}\rangle_i$ with s and $\vec{a}$
    $Q_i(\langle\vec{\pi_i}\rangle_i, s, a) \leftarrow (1-\alpha)Q_i(\langle\vec{\pi_i}\rangle_i, s, a) + \alpha\left[r + \gamma \max_{a_i'} Q_i(\langle\vec{\pi_i}\rangle_i', s', a_i')\right]$
    $s \leftarrow s'$; $\langle\vec{\pi_i}\rangle_i \leftarrow \langle\vec{\pi_i}\rangle_i'$;
  until s reaches the end of the episode

Figure 2: Algorithm of a Level-1 agent of index i.

about the co-player's policy belief, for better estimation of the co-player's policy. The simulation results on Iterated Prisoners' Dilemma (IPD) games show that our algorithm show better results than agents based on existing algorithms. We expect that such an algorithm would lead us to study mathematical structure of communication, such as mutual-estimation structure [4].

2. ALGORITHMS

As a starting point, we suppose a simple agent called Level-0, which uses a single-agent Q-learning algorithm (Fig. 1). This agent treats the other agents as a part of an environment and pays no attention to them. As a result, such an agent can only derive a very primitive learning in a multi-agent environment.

In the following, we discuss agents that use information of other agent's policies. Since agent i cannot directly see policy π_j of another agent j, agent i needs to obtain an estimate $\langle\pi_j\rangle_i$ of j's policy. If agent j has a stationary policy, some kind of statistical method can accurately estimate the policy of agent j by collecting a sufficiently large sample of the co-player's actions. Such a statistical method can be also applied to an co-player agent whose policy changes but converges to a limit [7].

2.1 Level-1: Using Policy-based Beliefs

We can extend the algorithm of a Level-0 agent to choose actions with regard to the policy beliefs $\langle\pi_i\rangle_i$ of other agents. One simple way to do so is to regard the tuple of policy belief and the original state $(\langle\vec{\pi_i}\rangle_i, s)$ as an extended state. Accordingly, agent i has an extended policy $\pi_i^+(\langle\vec{\pi_i}\rangle_i, s; a)$ and extended action-value function $Q_i^{\pi_i^+}(\langle\vec{\pi_i}\rangle_i, s; a)$. Assuming that the belief is accurate and that the change to the policies of the other agents has the Markov property, the agent can learn to choose an optimal policy. Thus we can design a Level-1 agent, which obtains $\langle\pi_i\rangle_i$ by using some statistical estimation and use $Q_i^{\pi_i^+}$ to choose an action.

We can have another viewpoint on this algorithm, in which a Level-1 agent i chooses a policy, instead of just an action, by using its belief about the other agents' policies. To choose a policy, the agent uses a *metapolicy* Π_i, which maps the policy belief of the other agents $\langle\vec{\pi_i}\rangle_i$ to a policy π_i. Interestingly, a metapolicy Π_i is just a notational variant of an extended policy π_i^+, since we can obtain a policy from π_i^+ by fixing $\langle\vec{\pi_i}\rangle_i$. Figure 2 shows the algorithm of a Level-1 agent in the new notation.

Initialize $Q_i(\langle\vec{\pi_i}\rangle_i, s, a)$, $\langle\vec{\Pi_i}\rangle_i$, and $\langle\langle\vec{\pi}\rangle\rangle_i$
Repeat (for each episode):
  Observe the initial state s
  Repeat (for each step of episode):
    Derive Π_i from Q_i
    $\langle\vec{\pi_i}\rangle_i \leftarrow \langle\vec{\Pi_i}\rangle_i\left(\langle\langle\vec{\pi}\rangle\rangle_i\right)$; $\pi_i \leftarrow \Pi_i\left(\langle\vec{\pi_i}\rangle_i\right)$
    Choose a_i from π_i and s
    Take action a_i, observe $\vec{a}, r, s'$
    $\langle\vec{\Pi_i}\rangle_i' \leftarrow$ New belief, obtained by updating $\langle\vec{\Pi_i}\rangle_i$ with $\langle\langle\vec{\pi}\rangle\rangle_i$, s and $\vec{a}$
    $\langle\langle\vec{\pi}\rangle\rangle_i' \leftarrow$ New embedded belief, obtained by updating $\langle\langle\vec{\pi}\rangle\rangle_i$ with s and $\vec{a}$
    $\langle\vec{\pi_i}\rangle_i' \leftarrow \langle\vec{\Pi_i}\rangle_i'\left(\langle\langle\vec{\pi}\rangle\rangle_i\right)$
    $Q_i(\langle\vec{\pi_i}\rangle_i, s, a) \leftarrow (1-\alpha)Q_i(\langle\vec{\pi_i}\rangle_i, s, a) + \alpha\left[r + \gamma \max_{a_i'} Q_i(\langle\vec{\pi_i}\rangle_i', s', a_i')\right]$
    $s \leftarrow s'$; $\langle\vec{\Pi_i}\rangle_i \leftarrow \langle\vec{\Pi_i}\rangle_i'$; $\langle\langle\vec{\pi}\rangle\rangle_i \leftarrow \langle\langle\vec{\pi}\rangle\rangle_i'$
  until s reaches the end of the episode

Figure 3: Algorithm of a Level-2 agent of index i.

Assuming that the policy beliefs are accurate and their change has the Markov property, agent i can learn a metapolicy, which is optimal in the sense of choosing the best policy in response to the policies of the other agents. However, in playing with other Level-1 agents, this assumption becomes unrealistic, since the policy selected by a metapolicy changes rapidly.

2.2 Level-2: Using Embedded Beliefs

A more elaborated agent, which we call Level-2, develops a belief about metapolicies of other agents. As mentioned before, a metapolicy is a map from policy beliefs to a policy; to estimate the policy of agent j using a metapolicy, agent i needs to have an embedded belief within agent j. We denote $\langle\langle\pi_k\rangle_j\rangle_i$ to represent agent i's belief about agent j's belief about agent k's policy. Agent i can obtain a belief of agent j's policy, given metapolicy belief $\langle\Pi_j\rangle_i$ and embedded policy belief $\langle\langle\vec{\pi_j}\rangle_j\rangle_i$ about agent j:

$$\langle\pi_j\rangle_i = \langle\Pi_j\rangle_i\left(\langle\langle\vec{\pi_j}\rangle_j\rangle_i\right) \quad . \tag{3}$$

We denote the agent i's embedded beliefs of all the other agents as

$$\left\langle\overrightarrow{\langle\vec{\pi}_{\cdot}\rangle_{\bar{i}}}\right\rangle_i = \left(\left\langle\langle\vec{\pi_1}\rangle_1\right\rangle_i, \ldots, \left\langle\langle\vec{\pi}_{\overline{i-1}}\rangle_{i-1}\right\rangle_i, \left\langle\langle\vec{\pi}_{\overline{i+1}}\rangle_{i+1}\right\rangle_i, \ldots, \left\langle\langle\vec{\pi}_{\overline{N}}\rangle_N\right\rangle_i\right). \tag{4}$$

In a perfect information game, every agent shares the same information, and every agent knows that the information is shared. Thus, agent i has no reason to distinguish the embedded belief $\langle\langle\pi_k\rangle_j\rangle_i$ for agent j from the one $\langle\langle\pi_k\rangle_{j'}\rangle_i$ for agent j', and as a result, we can denote them as $\langle\langle\pi_k\rangle\rangle_i$. This reduces the representation of embedded beliefs from $\left\langle\overrightarrow{\langle\vec{\pi}_{\cdot}\rangle_{\bar{i}}}\right\rangle_i$ to $\langle\langle\vec{\pi}\rangle\rangle_i$.

A Level-2 agent obtains an embedded belief $\langle\langle\pi_k\rangle\rangle_i$ by using the same statistical estimation method as the one for Level-1's belief policy; the only difference is that Level-2 needs $\langle\langle\pi_i\rangle\rangle_i$, the embedded estimation of the policy of agent i itself within another agent.

Using $\langle\langle\vec{\pi_j}\rangle_j\rangle_i$, Level-2 agent i is able to develop a belief about the metapolicy $\langle\Pi_j\rangle_i$ of agent j. Assuming the space of the embedded policy is finite, one can perform statistical estimations for every possible $\langle\langle\vec{\pi_j}\rangle_j\rangle_i$. By using the embedded policy belief $\langle\langle\vec{\pi}_{\cdot}\rangle_{\bar{i}}\rangle_i$ and the metapolicy belief $\langle\Pi_j\rangle_i$, a Level-2 agent can derive a policy belief $\langle\vec{\pi_i}\rangle_i$ and perform reinforcement learning on $Q_i(\langle\vec{\pi_i}\rangle_i, s; a_i)$. Note that this extended action-value function is the same as the one in the Level-1 agent.

Figure 3 shows the algorithm of the Level-2 agent. The algorithm is the same as that of the Level-1 agent (Fig. 2), except that $\langle\vec{\pi_i}\rangle_i$ is now calculated from $\langle\langle\vec{\pi}\rangle\rangle_i$ and $\langle\vec{\Pi_i}\rangle_i$.

2.3 Level-n: Using Deeper Embedded Beliefs

We can go along this line to create higher-lever agents successively. For instance, a Level-3 agent maintains $\langle\langle\langle\vec{\pi_l}\rangle_k\rangle_j\rangle_i$, $\langle\langle\Pi_k\rangle_j\rangle_i$,

Table 1: Pay-off matrix for agents X and Y

X \ Y	C	D
C	3 \ 3	5 \ 0
D	0 \ 5	1 \ 1

and $\langle \Pi_j \rangle_i$; A Level-4 agent maintains $\langle\langle\langle\langle \vec{\pi}_m \rangle_l \rangle_k \rangle_j \rangle_i$, $\langle\langle\langle \Pi_l \rangle_k \rangle_j \rangle_i$, $\langle\langle \Pi_k \rangle_j \rangle_i$, and $\langle \Pi_j \rangle_i$; and so on. In any case, the innermost embedded policy belief is obtained by a statistical estimation, and applied to metapolicy beliefs to obtain a, presumably better, policy belief.

Although the number of embedded policy beliefs and metapolicy beliefs seems to increase exponentially, we can reduce it by assuming a perfect information game as in the Level-2 agent's case. Since agent i has no reason to distinguish $\langle\langle \Pi_k \rangle_j \rangle_i$ from $\langle\langle \Pi_k \rangle_{j'} \rangle_i$, we can combine them into $\langle\langle \Pi_k \rangle\rangle_i$. Similarly, $\langle\langle\langle \vec{\pi}_l \rangle_k \rangle_j \rangle_i$ for any k and j can be reduced to $\langle\langle\langle \vec{\pi}_l \rangle\rangle\rangle_i$. This sort of reduction limits the number of beliefs to a linear increase.

3. SIMULATION EXPERIMENTS

We conducted a series of simulation experiments, in which the proposed agents play the Iterated Prisoners' Dilemma [**?**], a simple two-player game. In every turn, each agent would independently choose an action, either cooperation (C) or defection (D). The agents would then observe the result, the choice of the other agent and the pay-off given as in Table 1. Each agent would strive to maximize the sum of the pay-offs for itself over many turns. Noise in action is introduced with the probability of ε ; thus an agent chooses unwanted action with a probability of $\varepsilon/2$.

In the experiments, we applied reinforcement-learning algorithms to the game. A state refers to the last step of the action history of the two agents, i.e., $s(t) = (a_{\text{other}}(t-1), a_{\text{self}}(t-1))$; thus there are four possible states. Since we only used a deterministic policy, a policy becomes a four-dimensional vector of C or D. In other words, the policy space has only $2^4 = 16$ elements.

We specified an agent's belief space of another's policy to be equal to the policy space of the agent itself, that is, a four-dimensional vector of C or D. The statistical estimation method of the policy belief is simple; whenever the agent observes an action different from the policy belief, the agent replaces the corresponding element of the belief vector to make the belief match the observation.

We also provide two series of variants for comparison with our algorithm. One is Level-0(H3) agent, which is the same as Level-0 agent except that it uses three steps of action history as the state of reinforcement learning so that this agent has the same table size as the Level-n agents ($n \geq 1$) without using beliefs. The other is Level-n(A) agents, which uses action beliefs $\langle \vec{a}_j \rangle_i$ instead of policy beliefs $\langle \vec{\pi}_j \rangle_i$. Metapolicy reduces to a map from expected opponent's action $\langle \vec{a}_j \rangle_i$ to a policy, and so do metapolicy beliefs. Note that the algorithm of the Level-1(A) agent is equivalent to the NSCP-learner algorithm [7] (see eq. 2). To distinguish from these variants, we denote agents based on our proposed algorithms as Level-n(P).

In all of the simulations described below, we fixed $\alpha = 0.03$ and $\gamma = 0.99$. Since our interest is in the ability of agents playing against peer agents, we focused on simulations that cover only the case that two equivalent agents play IPD games.

3.1 Results

Figure 4 shows averaged pay-off over 2,000 trials of the IPD game by two same-level agents. After the initial learning time, the proposed Level-n(P) agents ($n \geq 1$) obtained average pay-offs very close to 3, the best pay-off that can be obtained only by mutual cooperation. This shows clear contrast to the pay-offs obtained by Level-0, Level-0(H3), and Level-n(A) agents; in fact, in the experiments of each of these agents, the agents failed learning to act cooperatively in more than 30% of the trials. These results indicate that our Level-n(P) agents have better ability to recognize the situation with peer learning agent and to learn the optimal policy.

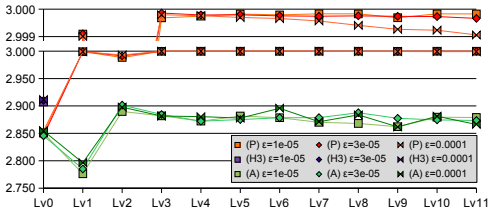


Figure 4: Average pay-off after learning with various ε values. Pay-off is averaged for 5×10^8 steps after 5×10^8 learning. The upper graph enlarges the top part of the lower graph.

Effects of having deeper embedded beliefs are not clear, except slight loss of pay-off is observed in case of $\varepsilon = 0.0001$. This indicates that there is some penalty to keep deeply embedded beliefs in noisy environments. However, since the penalty is very small, we can say that our algorithm is robust to the noise.

4. SUMMARY

We developed a new series of multi-agent reinforcement-learning algorithms that choose a policy based on beliefs about co-players' policies. The algorithms are applicable when a state is fully observable, and there is no limit on the number of players or a particular configuration of pay-offs. By assuming that the co-players also use policy-based beliefs to choose a policy, we also presented algorithms that use various depths of embedded policy beliefs.

We compared the algorithms of different embedding depths and algorithm variants through a series of simulation experiments on the IPD game. The results showed that the algorithms using policy-based belief converged to mutually cooperative behavior, unlike the existing algorithms based on action-based beliefs.

Acknowledgments

This study is partially supported by the Advanced and Innovational Research program in Life Sciences and Grant-in-Aid for Scientific Research on Priority Areas 17022012 from MEXT of Japan.

5. REFERENCES

[1] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proc. of AAAI-98*, pages 746–752. 1998.

[2] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proc. of ICML 1998*, pages 242–250. 1998.

[3] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. of ICML 1994*, pages 157–163. 1994.

[4] T. Makino and K. Aihara. Self-observation principle for estimating the other's internal state. Mathematical Engineering Technical Reports METR 2003–36, the University of Tokyo, Oct. 2003.

[5] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11:387–434, 2005.

[6] Y. Shoham, R. Powers, and T. Grenager. On the agenda(s) of research on multi-agent learning. In *Proc. of Artificial Multiagent Learning: Papers from the 2004 AAAI Fall Symposium, Technical Report FS-04-02*. 2004.

[7] M. Weinberg and J. S. Rosenschein. Best-response multiagent learning in non-stationary environments. In *Proc. of AAMAS'04*, pages 506–513, 2004.

Rule Value Reinforcement Learning for Cognitive Agents

Chris Child and Kostas Stathis
City University, London, UK
{c.child, k.stathis}@city.ac.uk

ABSTRACT

RVRL (Rule Value Reinforcement Learning) is a new algorithm which extends an existing learning framework that models the environment of a situated agent using a probabilistic rule representation. The algorithm attaches values to learned rules by adapting reinforcement learning. Structure captured by the rules is used to form a policy. The resulting rule values represent the utility of taking an action if the rule's conditions are present in the agent's current percept. Advantages of the new framework are demonstrated, through examples in a predator-prey environment.

Categories and Subject Descriptors

I.2.1 [**AI/GEN**]

General Terms

Algorithms

Keywords

Reinforcement learning, perception, action, planning, situated agents, stochastic, environment, probabilistic logic.

1. INTRODUCTION

The overall aim of our research is to build agents that can learn to act autonomously in a stochastic environment through experience gathered from interaction with the environment. Acquired stochastic logic rules are used to provide a compact model of the effects of agent action in the environment, and reinforcement learning techniques are used to plan within that model. RVRL provides a method for planning and action within this context.

If the environment the agent is modelling can be described in terms of a set of state variables, a factored state-model can be used. This describes the environment in terms of the dependencies between state variables and the evolution of these variables with respect to the actions taken by an agent.

The method used in this research is to create planning operators from experience of interactions with the environment. These rules predict how the environment will change when the agent takes an action. The general form of a stochastic planning operator is expressed as a rule of the form:

p: $e \leftarrow a, c$

p is the probability that the effects (e) of this operator will become true given the conditions (a, c) of the operator hold. a is the action taken by the agent, and c is a set of state variables representing the context of the agent's perception of the environment for the operator. Both a and c may be empty. In order to restrict the number of possible operators, e is defined to be a *single* state variable for each operator.

The process of building a rule set from experience requires the identification of conditions relevant to the effects of a rule. An effective method of building planning operators from experience is to use statistical significance to identify whether additional conditions are relevant to the outcome. This is the method used to create the rule sets in this work using ASDD [2]. ILP has also been used to learn rules of this form [3]. Precedence between rules is used to resolve conflicts in situations where two or more rule-sets match the conditions for the same output variable (for full details of the "precedence" algorithm, see [2]). This paper is an abridged version of the work presented in [1].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

2. RULE VALUE REINFORCEMENT LEARNING

Section 1 outlined the use of rules to model an environment. The next task is to use this rule model to develop an effective policy. One method of achieving this is to use Q-Learning [1]. The update function for Q-learning is as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[R_s, + \gamma \max_{a'} Q(s', a') - Q(s, a)] \quad (2.1)$$

s and a are the states and actions. $Q(s,a)$ indicates the current Q value for the state action pair. This update rule gradually improves estimates on the target function Q. The α parameter is a step-size, indicating how quickly the new estimate should change the old one. γ indicates the discount factor, determining the influence of future rewards on the current state. If we use this function and take sample results (i.e. s' is taken to be the random result after taking action a in state s) the learning is one-step temporal difference (TD) learning.

2.1 The Rule Value Update Function

RVRL uses the same principle as TD learning to update a value associated with each rule, rather than each state. The main advantages of using a state-based aggregation method, such as RVRL, over standard reinforcement learning are that the agent:

a) Does not store a value-map containing every state-action.

b) Can generalize over many states, allowing each value to represent states with similar properties.

If a model of the environment is available, full backup values can be used. Rather than taking a random sample for s_{t+1}, the probability (P) of reaching each possible next state (s') given that action (a) was taken in state (s) can be used in the equation, and the best next action taken as the maximum action (a) for each possible next state, as in dynamic programming (DP). The update function for DP [4] is:

$$Q(s, a) \leftarrow$$

$$Q(s, a) + \sum_{s'} P_{ss'}^{a} \alpha[R_{s'} + \gamma \max_{a'} Q(s', a') - Q(s, a)] \quad (2.2)$$

The stochastic planning operators act as a model in RVRL. It is therefore possible to use an adaptation of equation (2.2). The rule values for operators cannot be updated directly using this equation because more than one rule will match the next state (`s'`) and would therefore be used to generate consecutive states due to several output variables being present. The rule learning function, therefore, replaces `Q(s',a')` with an average value for all matching rules which have precedence (and would therefore be used in generation of the successor state). The rules with precedence are used to give the most accurate representation of the dynamics of the environment in state s'.

`Q(s,a)` is replaced by the value of the rule which will be updated. All matching rules are updated in turn by the update function because their estimated value will be improved by the update, whether they have precedence or not. The update function for RVRL is defined as:

$$forEach\ (rule \in MatchingRules\ (s, a))$$

$$\left\{ \begin{array}{l} Q(rule) \leftarrow Q(rule) + \\ \sum_{s'} P_{ss'}^{a} \alpha[R_{s'} + \gamma \max_{a'} AvgQ(WinningRules(s', a')) \\ -Q(rule)] \end{array} \right\} \quad (2.3)$$

`AvgQ(WinningRules(s',a'))` returns the average rule value for rules which have precedence in state `s'` if action `a'` is taken. `AvgQ(WinningRules(s',a'))` finds the average value of the winning rules and returns the value. `MatchingRules(s,a)` returns all rules whose conditions match the current state and action. The values of all the returned rules are updated by equation (2.3).

2.2 Iterative Rule Value Evaluation

The process of building successor states described in [2], combined with the rule-value update function (2.3) allows continuous generation of next states from an initial state and updates the rule values until satisfactory values have been generated. This process is described by the following algorithm:

```
Initialise Q(rule) = 0, for all rule ∈ rules;
Repeat {
  Initialise s = random state, a = random action;
  Generate next states, s' and prob(s') for s,a
  totalValue = 0; totalReward = 0;
  For each s' ∈ successor states {
    totalReward += reward(s') * prob(s');
    maxActionValue = -∞;
    maxAction = null;
    For each a' ∈ actions {
       actionValue = AvgQ(WinningRules(s',a'));
       if (actionValue > maxActionValue)
         maxActionValue = actionValue; }
    totalValue += maxActionValue * prob(s'); }
  For each rules ∈ matchingRules(s,a) {
    Q(rule) = Q(rule) +
              α[totalReward + γ*totalValue;
              -Q(rule)]; }
} for n steps
```

The sampling (TD learning) equivalent to this method would take a sample next state `s'` rather than calculating the probability of each next state. The process is otherwise the same.

3. EXPERIMENTATION

We tested RVRL in a predator prey environment, consisting of a 4x4 grid surrounded by a wall and containing a predator agent (P) and a prey agent (A). P has caught A if A is on the same square as P at the end of A's move, represented as A being under P (or `Agent_U`). A selects a random action every move. Both P and A have four actions: move north (`Move_N`), east (`Move_E`), south (`Move_S`) and west (`Move_W`). An action moves an agent one square in the direction, unless there is a wall, in which instance there is no effect. P and A move alternate turns. An agent's percept gives the contents of the four squares around it and the square under it. Each square can be in one of three states: empty (`E`), wall (`W`) or agent (`A`).

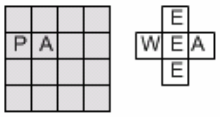


Figure 3-1: P, A in 4x4 grid. P's percept is shown to the right.

The task for RVRL was to construct an effective policy under the above, allowing the predator to catch the prey with optimal frequency. The task is complicated by the fact that the predator (P) is only adjudged to have captured the prey (A) if it moves into, or remains in, the predator's square at the end of its turn. P and A will continue to move after the prey is caught (continuous task).

In experiments it was found that, with a small amount of experience in the environment, P will tend to move next to A but not on-top of it. This is a good tactic as A is likely to move onto P and be caught. The optimal tactic, however, was found to be one in which P moves into A's square every move. This enables the predator (P) to be constantly in sight of the prey (A) and catch it whenever it moves into a wall. An example of a rule which captures this behaviour is: `Agent_N, Move_N`. Our experiments showed that RVRL gives high value to this rule and the `S, E` and `W` equivalents. Rules which attain higher value than the above have conditions such as: `{Agent_U, Move_N, Wall_N, Wall_E}`, in which P is on-top of the A in the NE corner of the map and choose action move north, into a wall (giving P a 50% chance of catching A). `Wall_N` and `Wall_E` are state variables representing that there is a wall north and east. The "effects" of the rules are not shown, because equal value will be learned for all rules with the same conditions. A sample of the final rule weights from rules learned from 60,000 moves experience after RVRL was run on the rule set for 15,000 iterations is given in Table 1.

Rules 1 and 2 have the same conditions but different actions (P is in the SW corner of the grid, with A underneath it). The rule has a positive value if P moves into a wall (1) and a negative value if the P moves away from the wall (2). P would, therefore, move into the wall and thus have the highest chance of catching A (50% chance of A moving into a wall).

Rules 3 and 4 have the same weight. If P takes the move north action (3), it will be on-top of A and will catch it if it moves into the wall to the west (25% chance). If P takes the move north

action in (4) it will move into the wall and therefore stay on-top of A. P will then catch A if it moves into the wall to the north (25% chance). These two situations should be of equal utility to the agent, which has been successfully learned by RVRL.

Table 1: Sample rule weights for rules learned from 60,000 moves experience and 15,000 iterations of RVRL.

No.	*Conditions*	*Value*
1	Move_W, Wall_W, Wall_S, Agent_U	0.43
2	Move_E, Wall_W, Wall_S, Agent_U	-0.03
3	Move_N, Wall_W, Agent_N	0.11
4	Move_N, Wall_N, Agent_U	0.11
5	Move_E, Agent_E	-0.07
6	Move_W, Agent_E	-0.21
7	Move_S	-0.28
8		-0.28

Rules 5 and 6 show the weights for moving east onto the prey to the east and moving west away from a prey to the east. Moving onto the prey has higher weight as expected. Rules 7 and 8 have the same value. Rule 7 is the value of moving south with no other information. Rule 8 has no conditions and is thus the value of taking a random move in the environment. These rules are both effectively random and thus have equal weight.

The performance of RVRL was compared with:

a) Dyna-Q: Q-Learning reinforcement learner with frequency based environment model.

b) SR-Q: A stochastic rule based model of the environment was used to build a state, action value map. This is the equivalent of running Q-learning, using the rule based model for experience to build the Q(s, a) map. This method is described in [2].

RVRL builds a Q(rule) map, assigning value to each rule. Table 2 gives a comparison of the three methods.

In each test case the methods were given the same experience with which to build the model. The predator and prey were run for a set number of steps, taking random moves at each step. Using the model, each method ran Q-learning (in the first two cases), or RVRL for 15,000 iterations to build a value map. Once the map had been created, each method ran for 40,000 steps in the predator prey environment, selecting the action with the highest utility at each step. The number of times P caught A was then recorded. Average moves taken for P to capture A is given in Table 2.

The two Q-learning based methods selected the best action at each step picking the highest valued action from all matching Q(s,a) values for the current state (s). RVRL picked the highest valued action from all matching `AvgQ(WinningRules(s,a))`.

Table 2: Moves per capture: Dyna-Q, Stochastic Rule model Q (SR-Q) and Rule Value Reinforcement Learning (RVRL).

Method	100	500	1000	10000	15000	30000	60000
Dyna-Q	17.5	16.4	12.0	8.8	7.4	6.2	4.6
SR-Q	13.1	13.4	11.5	9.2	8.8	7.1	4.7
RVRL	13.2	12.7	11.3	9.3	8.1	7.0	4.7

Moves per capture for P taking random moves is 16.01. A trail was also run on a "perfect" model (Dyna-Q built from 400,000 moves). In this instance P averaged 4.32 moves per capture.

The results in Table 2 for 100, 500 and 1000 moves training data show that RVRL is more effective than Dyna-Q when little experience has been gathered. In this case the Dyna-Q agent is forced to take a random move in many of the states encountered in the test, because it has no experience which matches the situation. With this limited model, Dyna-Q "expected" the prey to move in the same way as it did in the training data, resulting in picking a poor action. The RVRL agent, however, was able to make generalisations in two ways: first to generalise the *model* using the stochastic logic rules, which allows the system to predict future states from the current state, even when this state has not been seen before; second, RVRL learned values are applicable across multiple states, allowing learned values to be applied in unseen states. This allows the small amount of experience gathered to be generalised and used, which is demonstrated by the improved performance under these conditions. SR-Q is only able to make use of the first of these generalisations, and therefore performed slightly better than Dyna-Q, but not as well as RVRL.

As the state action map gains a larger amount of experience (10,000, 15,000 and 30,000 steps), its model becomes closer to a perfect model in this test environment, while the generalisations made by the rule learner become less effective. This is due largely to shortcomings in the ASDD modelling method with this level of training data [2] which is reflected in the similar performance of the SR-Q results, rather than shortcomings in the RVRL algorithm. When the learned rules become a near perfect representation of the environment (at 60,000 steps training data), the results show that RVRL is capable of learning near perfect valued rules, and thus the utility of taking an action in the current state, again demonstrating that the rule values are capable of capturing a policy at least as effectively as a state action model under these conditions.

4. CONCLUSIONS

This paper has presented the Rule Value Reinforcement Learning (RVRL) method. RVRL is a state-based aggregation technique, in that states which behave in a similar way with respect to a given action sequence and goal are given the same value. Results in our experimentation are extremely encouraging in that the algorithm is able to learn rule-values which accurately capture the utility of actions in the predator-prey environment without the need for a state-action map.

5. REFERENCES

[1] Child, C. and Stathis, K. "Learning to Act with RVRL agents". City University Technical Report. (2006).

[2] Child, C. and Stathis, K. "The Apriori Stochastic Dependency Detection (ASDD) Algorithm for Learning Stochastic Logic Rules", *In Proceedings of the 4th International Workshop on Computation Logic in Multi-agent Systems (CLIMA-04),* J. Dix, J. Leiter (Eds), Florida, Jan, (2004).

[3] Pasula, H. M, Zettlemoyer, L.S. and Kaelbling, L.P. "Learning Probabilistic Relational Planning Rules." *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling,* ICAPS, 73-82, (2004).

[4] Sutton, R.S., and Barto, A.G. "Reinforcement Learning: An Introduction". A Bradford Book, MIT Press, (1998).

[5] Watkins, C. J. C. H. "Learning from Delayed Rewards." PhD thesis, Cambridge University, (1989).

Learning Executable Agent Behaviors from Observation

Andrew Guillory, Hai Nguyen, Tucker Balch, Charles Lee Isbell, Jr.
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
{guillory, haidai, tucker, isbell}@cc.gatech.edu

ABSTRACT

We present a method for learning a human understandable, executable model of an agent's behavior using observations of its interaction with the environment. By *executable* we mean that the model is suitable for direct execution by an agent. Traditional models of behavior used for recognition tasks (*e.g.*, Hidden Markov Models) are insufficent for this problem because they cannot respond to input from the environment. We train an Input/Output Hidden Markov Model where the output distributions are mixtures of learned low level actions and the transition distributions are conditional on features detected by the agent's sensors. We show that the system is able to learn both the behavior and human-understandable structure of a simulated model.

Categories and Subject Descriptors

I.5 [**Computing Methodologies**]: Pattern Recognition

General Terms

Algorithms, Verification

Keywords

Behavior modeling, Input/Output Hidden Markov Models

1. INTRODUCTION

An ethologist studying the behavior of an animal such as an ant or bee typically starts by defining a set of low-level actions the animal performs. Having defined such low-level actions, the scientist then determines high-level structure; that is, she models what causes the animal to switch between low-level actions. Similarly, a roboticist or agent developer might program a behavior by coding low-level control modes corresponding to primitive actions and combining these low-level control modes together through some kind of high-level action selection mechanism. In this paper we present a method for learning both low-level actions and a high-level switching model. The result is human-understandable, but also *executable* in the sense that it can be used as a control program in a simulation or as part of an agent.

More formally we assume we are given a model of the agent's perception, a set of perceptions that may cause the agent to switch actions, and example trajectories of the agent interacting in its environment (tracking data), some of which has been labeled by a human expert designating the action the agent is performing at a given time step. We further assume that the agent acts according to a Markov Process with inputs. Our task is then to compute an executable model of the agents behavior composed of controllers for low-level actions and rules for switching between them.

Our approach is to train an Input/Output Hidden Markov Model (IOHMM) where the output distributions are mixtures of learned low-level actions and the transitions are conditioned on perceptions of the agents. We first learn from labeled data a set of low-level actions that may be assigned to states in the model. Having learned these low-level actions, the problem is then to learn from unlabeled data a mapping between states and low-level actions as well as the transition rules between states. By dividing the problem this way, the human expert can simply label portions of the data where the agent is obviously performing a particular action (not necessarily data in which the agent switches actions).

Our work is in contrast to *activity recognition*. Although models of behavior learned in activity recognition are often generative, they are not usually suitable for re-creation. Hidden Markov Models (HMMs), for example, are generative models but are usually limited in their ability to recreate real activities because they cannot respond to input. Goldberg and Matarić use an HMM based model to learn a robot's behavior patterns online [5]. Our work is more similar in goal to *imitation learning*. The notion of primitives in [3] is similar to our notion of low-level actions. Such techniques generally differ from our work by focusing more on learning and detecting low-level actions while constrained by the kinematics of particular robots. Our work is most similar to that of [4]. The authors approach the problem from the perspective of control theory and recover from unlabeled data both low-level control modes and a high-level control program in the form of a motion description string. Other related work includes [7] where the authors learn decision trees for low-level control modes.

In order to test our learning algorithms, we programmed by hand a model of foraging inspired by the behavior of social insects, using the TeamBots simulation platform and motor schema-based control [1]. Our experimental method

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

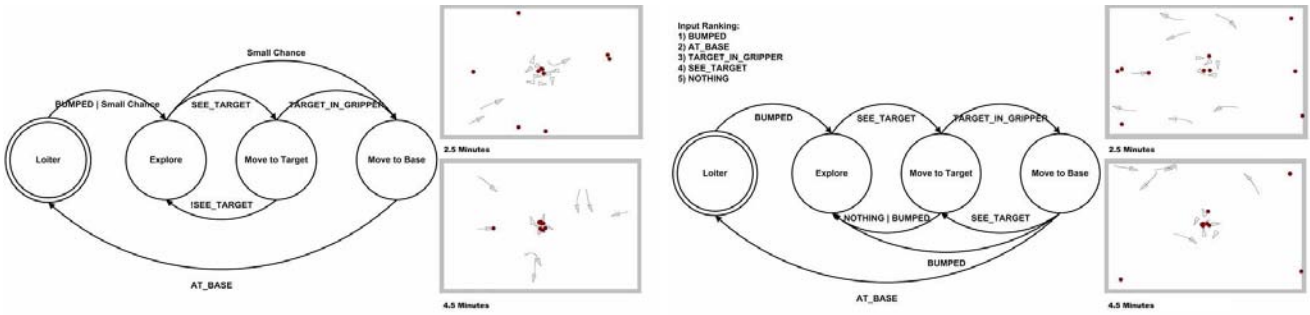


Figure 1: On the left, our original model, including screenshots from the simulation. On the right, a high likelihood learned model, showing the most likely transitions for every input, excluding self transitions.

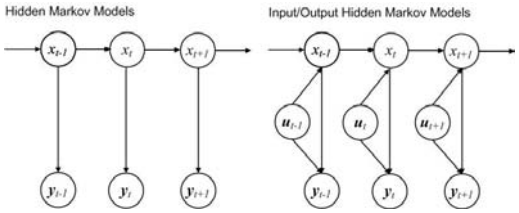


Figure 2: Networks for IOHMMs and HMMs

was to generate data from this known model, then see if we are able to automatically recover the behavior from the data. Figure 1 shows our model. From the simulation we created two data sets, one labeled with the action the agent is performing at each time step and one unlabeled. For both data sets we ran the model in simulation with 12 agents at 33 frames per second, waiting for all of the targets to be carried to base, recording at each frame the positions and orientations of all agents as well as the positions of all targets. Our goal is then to, from the labeled data set, learn the low-level actions of the agent and, from the unlabeled data set, learn the switching behavior.

2. IOHMMS AS EXECUTABLE MODELS

HMMs [6] represent the probability of an observation sequence $P(\mathbf{y}_1^T)$ where $\mathbf{y}_1^T = \mathbf{y}_1 ... \mathbf{y}_T$ is a sequence of observation vectors. IOHMMs [2] are a generalization of this model that represent the conditional probability of an observation sequence given an input sequence $P(\mathbf{y}_1^T | \mathbf{u}_1^T)$ where $\mathbf{u}_t$ is the input vector for time t. Figure 2 shows the dependency assumptions of the two models, where x_t is the discrete state of the model at time t. The essential difference between the two models, is that in an IOHMM the transition distributions $P(x_t | x_{t-1} = i, \mathbf{u}_t)$ and output distributions $P(\mathbf{y}_t | x_t = i, \mathbf{u}_t)$ are both conditional on the current input vector. The standard algorithm for training IOHMMs is an Expectation Maximization (EM) algorithm that is a straightforward extension of Baum-Welch for HMMs. After training, the model can be interactively executed on an input sequence in a manner similar to a finite state machine.

In our case, the input $\mathbf{u}_t$ is the tracking data at time t, including the position of the agent. The corresponding output $\mathbf{y}_t$ is the next position of the agent after moving. By using the model interactively in a simulator environment we can recreate the global behavior of the agent. Although consistent with previous papers on IOHMMs, our use of variable names and the terms *input* and *output* are different than their use in dynamic systems literature (where input often refers to input into the environment).

We model output distributions as mixtures over already learned low-level actions. Mixture distributions marginalize a distribution over a discrete variable, in this case a_t, the low-level action the agent is performing at time t. Normally, a mixture distribution in an IOHMM would have mixing weights conditioned on $\mathbf{u}_t$ as well as x_t. For our application, input-conditional mixing weights are undesirable because they complicate the meaning of a state–there is not a simple mapping between states and low-level actions. Our approach is to use unconditional mixing weights but also for the purposes of parameter estimation model the output distributions as generating both $\mathbf{u}_t$ and $\mathbf{y}_t$. This allows us to represent information about $\mathbf{u}_t$ without introducing input-conditional mixing weights. We further assume $\mathbf{u}_t$ and $\mathbf{y}_t$ are independent of x_t given a_t (that the low level actions are the same for each state). The overall model is something like an HMM/IOHMM hybrid: the transition distributions are conditional while the output distributions are not. With this model, assuming we have already calculated the values for the mixture components at each time step, which we factor into $P(\mathbf{y}_t | a_t, \mathbf{u}_t)$ and $P(\mathbf{u}_t | a_t)$, only the mixing weights need to be re-estimated during EM training.

We must also represent our transition distributions. In our experiments we avoid some complexity and maintain human understandability by conditioning the transition distributions on a set of binary sensory features calculated from $\mathbf{u}_t$, just as we have done in our simulated domain. We can then represent the transition distributions as look up tables. We experimented with ranking our binary features. We then took as our input the most important triggered feature. This ranking method greatly reduces the number of parameters and constrains the model to resemble hand-made models where only a single binary feature triggers a transition. Unfortunately, choosing a feature ranking requires domain knowledge. We also experimented with a standard binary encoding, allowing for a different transition rule for every possible combination of feature values.

When estimating probabilities in look up tables, it is standard to add a small constant to the counts for each entry to allow some probability mass everywhere. In our case, it is useful to add a small number to only the entries corresponding to self-transitions. In other words, given little evidence, an agent should remain in its current state.

3. LEARNING LOW-LEVEL ACTIONS

To estimate the mixing weights during EM from unlabeled data we need to estimate values for $P(\mathbf{y}_t|a_t, \mathbf{u}_t)$ for each time step and low-level action. To execute the resulting model, we also need to be able to sample from this distribution. Our approach to this problem is to learn a controller for each low-level action from labeled data.

From labeled data, we can calculate sensory features and corresponding motor commands for each low-level action (example inputs and outputs for each controller). In our experiments we solve the function approximation problem by using a modified version of k-nearest neighbor (KNN), randomly choosing from the k nearest neighbors with the probability of each weighted by the point's distance from the query point. By randomly choosing in this way we can model random actions like the Explore action. With the learned controller and a simulation environment, we can sample from $P(\mathbf{y}_t|a_t, \mathbf{u}_t)$ (*i.e.* simulate a single time step of the agent's motion). From this we can also estimate values for the distribution for unlabeled data using kernel density estimation. For each time step and action we run a simulation many times to produce distributions of points predicting the next position of the agent. We then compare these point distributions to the actual next position to decide which action the agent is performing.

We also learn a model for $P(\mathbf{u}_t|a_t)$ from labeled input vectors. We are not interested in sampling from this distribution as during execution our model does not generate $\mathbf{u}_t$, but we need to estimate its values in order to calculate the mixing weights, since we modeled the output distributions as joint distributions over $\mathbf{u}_t$ and $\mathbf{y}_t$. In our experiments, we again reduce $\mathbf{u}_t$ to a set of binary features, using the same features and encoding as we did for the transition distributions, and learn look up tables.

4. RESULTS

We used k-nearest neighbors on the agent's sense-action pairs to learn the low level controllers. For the sensory features we used distance and angle to the closest obstacle, closest target, closest agent, and the base as well as whether or not the agent is currently carrying a target. For the motor outputs we calculated the distance and angle the agent moved at each time step as well as whether the agent was carrying an object or not. We then tested the learned controllers in simulation using the original forage switching behavior. Our learned low-level controllers were able to recreate the original foraging behavior even on foraging arenas where the position of food items are moved from their original position as well as differently sized arenas with more attractors and more obstacles. Even though the agents were able to recreate the original foraging behavior by successfully gathering all the targets, there were still flaws. For example, the agents would frequently miss targets while moving towards them in the Move to Target controller, especially targets in corners.

With learned low-level actions, we trained 150 models with EM for both ranked and unranked input types, and 40 ranked and 32 unranked input models learned the correct structure of the behavior (that there was one state for each low level action). The likelihood scores for the models that learned the correct structure were all higher than the scores for the models that did not. Their likelihood scores were also greater than the original model's. Within these models, there were several clusters of models whose likelihood scores were identical within range of our convergence criteria. We tested models from each of these clusters. Figure 1 shows the model from the cluster with the highest likelihood score for the ranked input type. The unranked input type model is similar, except with more complicated conditions on the transitions. 14 ranked and 21 unranked models were within the maximum likelihood clusters.

All of the models that learned the correct structure were able to recreate the forage behavior in simulation when combined with the learned controllers. However, they also all had flaws that affected the behavior to varying degrees. For example, many models did not know what to do when they missed a target or another agent reached a target before them. Some would even remain stuck in Move to Target in this situation, in the worst case then becoming stuck against a wall. Subjectively the models with the maximum likelihood scores seemed to recreate the behavior the best.

5. CONCLUSIONS AND FUTURE WORK

We have presented a method for learning an executable model of behavior from observations. We tested our method on simulated data generated from a hand-constructed model. The learned models recreated the behavior of the original model and exhibited the same basic structure. We hope to next apply these techniques to actual insect tracking data.

6. ACKNOWLEDGMENTS

Thank you to Sridhar Mahadevan, Zia Khan, and Adam Feldman for helpful discussions. This work was supported in part by NSF.

7. REFERENCES

[1] R. C. Arkin and T. R. Balch. Aura: Principles and practice in review. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 9(2-3):175–188, April 1997.

[2] Y. Bengio and P. Frasconi. Input-output HMM's for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249, September 1996.

[3] D. C. Bentivegna, A. Ude, C. G. Atkeson, and G. Cheng. Humanoid robot learning and game playing using pc-based vision. In *2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2449–2454, 2002.

[4] F. Delmotte and M. Egerstedt. Reconstruction of low-complexity control programs from data. In *43rd IEEE Conference on Decision and Control*, volume 2, pages 1460–1465, December 2004.

[5] D. Goldberg and M. J. Matarić. Detecting regime changes with a mobile robot using multiple models. In *2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS01)*, volume 2, pages 619–624, 2001.

[6] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, February 1989.

[7] C. Sammu, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. In *Ninth International Conference on Machine Learning*, 1992.

$RV_{\sigma(t)}$: A unifying approach to performance and convergence in online multiagent learning

Bikramjit Banerjee
Department of Electrical Engineering & Computer Science
Tulane University
New Orleans, LA 70118. USA
http://www.eecs.tulane.edu/Banerjee
banerjee@eecs.tulane.edu

Jing Peng
Department of Electrical Engineering & Computer Science
Tulane University
New Orleans, LA 70118. USA
http://www.eecs.tulane.edu/Peng
jp@eecs.tulane.edu

ABSTRACT

We present a new multiagent learning algorithm ($RV_{\sigma(t)}$) that can guarantee both no-regret performance (all games) and policy convergence (some games of arbitrary size). Unlike its predecessor ReDVaLeR , it (1) does not need to distinguish whether its opponents are self-play or otherwise non-stationary, (2) is allowed to know its portion of *any* equilibrium that, we argue, leads to convergence in some games in addition to no-regret. Although the regret of $RV_{\sigma(t)}$ is analyzed in continuous time, we show that it grows slower than in other no-regret techniques like GIGA and GIGA-WoLF. We show that $RV_{\sigma(t)}$ can converge to coordinated behavior in coordination games, while GIGA, GIGA-WoLF may converge to poorly coordinated (mixed) behaviors.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Theory, Performance

Keywords

Multiagent Learning, Game Theory

1. INTRODUCTION

Multiagent learning (MAL) in a reinforcement learning setting has been an active field of study recently. The problem is simply of multiple controllers trying to learn individually "optimal" control policies in a shared Markov Decision Process (MDP), often called a *stochastic game* or a *Markov game*. The difficulty arises from the fact that all agents are learning simultaneously, which means the MDP faced by each agent is essentially *non-stationary*. Hence, the concept of "optimal" policy becomes ill-defined, and depends on the collective behavior of the other agents. Previous research has attempted to tackle this problem by considering various *opponent classes* such that there is a well-defined "optimal policy" for the learner *for each class of opponents*. Typically the research contributions in this aspect has been to design learning algorithms that learn the appropriate behaviors for the corresponding class of opponents, *without any access to the class information*. We follow this line of research to present a new MAL algorithm for repeated games, with the general philosophy of policy convergence against some classes of opponents but otherwise ensuring high payoffs.

Most MAL algorithms are guaranteed to learn the best response strategies against (eventually) stationary opponents, i.e., those that play stationary strategies after an unknown of rounds. Even single agent learning techniques like Q-learning can achieve this. Building on this property, some MAL algorithms (e.g., WoLF-IGA [4], AWESOME [5]) ensure, in addition, convergence to a Nash equilibrium strategy in *self-play*. i.e., when all agents are following the *same* learning algorithm. However, there is a significantly large class of opponents that is not covered by either of these two cases, i.e., opponents that are neither stationary, nor self-play. In order to fill this void, we proposed [1] that a MAL algorithm should additionally ensure that if the opponent belongs to neither of the above classes, then it will achieve *no-regret* (defined in section 2) payoffs - something that can be construed as reasonably good performance when facing uncertain opponents. Although we had presented an algorithm, ReDVaLeR [1], that achieved all three properties, there were some limiting knowledge requirements such as

- The ReDVaLeR agent needed to know whether its opponents are self-play, or otherwise non-stationary. In this paper, we study a variant of ReDVaLeR , called $RV_{\sigma(t)}$ [2], that eliminates this assumption.
- In case of self-play, all ReDVaLeR agents need to know their portions of the *same* Nash equilibrium. $RV_{\sigma(t)}$ eliminates this requirement for some classes of games. Specifically, $RV_{\sigma(t)}$ agents only need to know their portions of *any* equilibrium in any 2×2 game, and in a game of any size with a unique mixed equilibrium (established before in [2]). In this paper, we argue theoretically and empirically, that this also holds in some coordination games with multiple equilibria.

We also show that the expression (in continuous time) for regret of $RV_{\sigma(t)}$ can have a slightly better form than those (discrete time) of GIGA [6] and GIGA-WoLF [3]. In a class of coordination games, GIGA, GIGA-WoLF are found to converge to mixed equilibria that make miscoordination highly likely besides yielding poor expected payoff to both agents, while $RV_{\sigma(t)}$ is found to always achieve strict coordination with Pareto-dominating payoffs.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. MULTIAGENT REINFORCEMENT LEARNING

A Multiagent Reinforcement Learning task is usually modeled as a Stochastic Game (SG, also called *Markov Game*), which is a Markov Decision Process with multiple controllers. We focus on stochastic games with a single state, also called repeated games, since we are more interested in dealing with uncertainties due to the other learners' changing behaviors. This refers to a scenario where a matrix game – given by an n-tuple of matrices, $\langle \mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_n \rangle$ where $\mathbf{R}_i$ is a payoff matrix of dimension $|A_1| \times |A_2| \ldots \times |A_n|$, such that the payoff of the ith agent for the joint action $(a_1, a_2, \ldots, a_n)$ is given by the entry $R_i(a_1, a_2, \ldots, a_n)$, $\forall i$ – is played repeatedly by multiple agents. We represent the action space of the ith agent as A_i. Table 1 shows examples of bimatrix games of a special class that we call *Simplified Coordination Games*. It is "simplified" because only coordinated action profiles have non-zero payoffs. This game has n pure equilibria (plus some mixed equilibria) given by the matrix diagonals, but players may prefer different ones (here row player prefers 3rd row, column player prefers 1st column).

Table 1: Simplified Coordination Games

Actions	C_1	C_2	C_3
R_1	(2,100)	(0,0)	(0,0)
R_2	(0,0)	(51,51)	(0,0)
R_3	(0,0)	(0,0)	(100,2)

A *mixed policy*, vector $\boldsymbol{\pi}_i \in \Delta(A_i)$ for agent i, is a probability distribution over A_i. We write the probability of the jth action of the ith agent at time t as $\pi_i^t(j)$. If the entire probability mass is concentrated on a single action (some actions), it is also called a *pure policy* (*partially mixed policy*). The joint policies of the opponents of the ith agent will be given by the vector $\boldsymbol{\pi}_{-i}$. We shall usually refer to the ith agent as the learner and the rest of the agents as the opponents. The expected payoff of the jth action of the learner against the opponent's current policy, $\boldsymbol{\pi}_{-i}^t$ is written as $V_i(j, \boldsymbol{\pi}_{-i}^t)$.

For a given time range $t = 0 \ldots T$, the regret of a learner (agent i), Rg_i^T is given by

$$Rg_i^T = \max_{\boldsymbol{\pi}_i} \sum_{t=1}^{t=T} V_i(\boldsymbol{\pi}_i, \boldsymbol{\pi}_{-i}^t) - \sum_{t=1}^{t=T} V_i(\boldsymbol{\pi}_i^t, \boldsymbol{\pi}_{-i}^t).$$

This compares the total payoff of the actual sequence of policies of the learner to the best response against the empirical distribution of the opponent.

3. OUR APPROACH: ReDVaLeR WITH VARIABLE σ

We make the usual assumptions about the boundedness of the game payoffs (only non-negative in our case); the learner's ability to use vanishing step sizes ($\eta \to 0$), and that the learner can observe its payoff vector $\mathbf{V}_i$, i.e., $V_i(j, \boldsymbol{\pi}_{-i}^t)$ for all actions j at any round. It *does not* need to observe the opponents' payoffs or policies/actions.

We use the ReDVaLeR algorithm from [1] with a time-varying schedule for σ in place of a constant. The discrete form of the algorithm (slightly different from [1]) is

$$\pi_i^{t+1}(j) = \frac{\pi_i^t(j) + \eta \pi_i^t(j) l_i^t(j) V_i(j, \boldsymbol{\pi}_{-i}^t)}{1 + \eta \sum_j l_i^t(j) \pi_i^t(j) V_i(j, \boldsymbol{\pi}_{-i}^t)} \quad (1)$$

for any j, η being a small step size and initial condition: $\pi_i^0(j) > 0$.

In contrast with [1], the learning rates $(l_i^t(j))$ for our algorithm, $RV_{\sigma(t)}$, are defined as

$$l_i^t(j) = \begin{cases} 1 + \sigma(t) & \text{if } \pi_i^t(j) < \pi_i^*(j) \\ 1 - \sigma(t) & \text{if } \pi_i^t(j) \geq \pi_i^*(j) \end{cases} \quad (2)$$

for a suitable σ-schedule as defined below, and $\boldsymbol{\pi}_i^*$ being the learner's portion of a Nash equilibrium [2] policy.

DEFINITION 1 (σ-SCHEDULE). *A time decaying schedule for $\sigma(t)$ is defined by the 3 conditions:*

1. *$\sigma(t)$ is continuous and $1 \geq \sigma(t) \geq 0, \forall t$,*
2. *$\sigma(t) \geq \sigma(t'), \forall t' \geq t$, and*
3. *$\sigma(t) \to 0$ as $t \to \infty$.*

3.1 Properties of $RV_{\sigma(t)}$

The following theorems (proofs can be found in [2]) summarize the main properties of $RV_{\sigma(t)}$ for $\eta \to 0$. We call games of arbitrary size having unique mixed Nash equilibrium as UME games.

THEOREM 1 (EVENTUALLY STATIONARY OPPONENTS). *If there exists a time τ_1 (unknown to the learner) such that all other agents play stationary policies at all times $t > \tau_1$, then for a given $\epsilon > 0$, there exists a time $\tau > \tau_1$, such that after τ an $RV_{\sigma(t)}$ agent i using σ-schedule in Definition 1, is guaranteed to converge to the set of ϵ-best response policies (i.e. payoffs are within ϵ of the best response payoff).*

THEOREM 2 (NO-REGRET). *If an $RV_{\sigma(t)}$ agent i uses the decaying σ-schedule of Definition 1 and $\pi_i^0(j) = 1/|A_i|$, then $\lim_{T\to\infty} \frac{Rg_i^T}{T} \leq 0$, i.e., the algorithm is asymptotically no-regret.*

THEOREM 3 (SELF-PLAY IN 2×2 GAMES). *In any 2×2 game, there exists a σ-schedule satisfying Definition 1, such that $RV_{\sigma(t)}$ will converge to a Nash equilibrium profile in self-play, even if they are given their portions of different equilibria.*

THEOREM 4 (SELF-PLAY IN UME GAMES). *There exists a σ-schedule satisfying Definition 1, which when followed by n $RV_{\sigma(t)}$ agents guarantees the convergence of their policies to the equilibrium profile in all UME games of any size, provided each agent knows appropriate loose-bounds on (1) the maximum game payoff of any agent, $R_{\max} = \max_i \bar{r}_i$, (2) the maximum size of action space among all agents, $\max_i |A_i|$, (3) the minimum equilibrium payoff among all agents, $\min_i V_i^*$, and (4) the total number of agents, n.*

3.2 Regret growth of $RV_{\sigma(t)}$

It has been argued in [2] that the σ-schedule $\sigma(t) = \frac{1}{1+\beta\sqrt{t}}$ for a small constant β (proportional to $1/(\bar{r}_i|A_i|)$) satisfies Definition 1 and all of the theorems simultaneously. This allows us to deduce a more direct expression for the regret in terms of the time variable, using the following lemma. Note that this lemma immediately leads to Theorem 2 for *any* σ-schedule satisfying Definition 1, and particularly for $\sigma(t) = \frac{1}{1+\beta\sqrt{t}}$.

LEMMA 5. *The regret of an $RV_{\sigma(t)}$ agent (for $\eta \to 0$) is given by $Rg_i^T \leq 2\bar{r}_i \int_0^T \sigma(t)dt + \log|A_i|$.*

Thus our unifying choice of $\sigma(t) = \frac{1}{1+\beta\sqrt{t}}$ yields a regret bound of

$$Rg_i^T \leq \left(\frac{4\bar{r}_i}{\beta}\right)\left(\sqrt{T} - \frac{\log(1+\beta\sqrt{T})}{\beta}\right) + \log|A_i|,$$

whereas GIGA-WoLF has

$$Rg_i^T \leq (2 + \bar{r}_i^2|A_i|)\sqrt{T} - |A_i|\bar{r}_i^2$$

and half that for GIGA. The presence of the log term in our expression for Rg_i^T might make it grow slower, though our analysis is in continuous time as opposed to discrete time analysis of GIGA and GIGA-WoLF. In order to make a more direct comparison, we experiment with all these methods *in discrete time*.

4. SIMPLIFIED COORDINATION GAMES

The following two lemmas (proofs omitted) and the experimental results suggest that $RV_{\sigma(t)}$ in self-play converges to playing coordinated actions even if they were given different pure equilibrium policies (e.g., the ones they prefer) at start. GIGA and GIGA-WoLF on the other hand can converge to poorly coordinated behavior with poor payoffs. Although the definition of this class of games and the following results are presented for two agent scenarios, it can be trivially extended to the multi-player case.

LEMMA 6. *Two $RV_{\sigma(t)}$ agents cannot converge asymptotically to uncoordinated pure policies in the SCGs.*

LEMMA 7. *If at a large enough time in the SCG, both $RV_{\sigma(t)}$ agents assign their highest probabilities to the same action (j), then the probability of both players playing j increases.*

Our experiments in this class of games used the game shown in Table 1 and various completely mixed starting policy profiles. Each $RV_{\sigma(t)}$ player was given its portion of the equilibrium policy that it would prefer, e.g., player 1 was given $[0, 0, 1]$ and player 2 was given $[1, 0, 0]$.

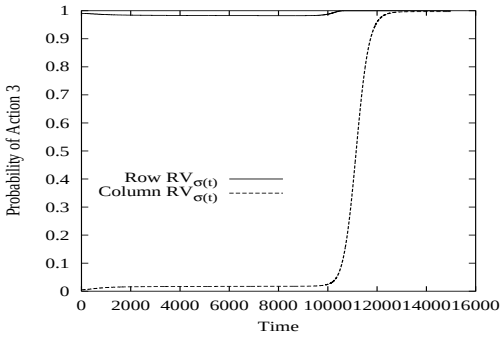


Figure 1: The probabilities of R_3 and C_3 of two $RV_{\sigma(t)}$ agents starting at $\langle[0.005, 0.005, 0.99], [0.99, 0.005, 0.005]\rangle$, in the coordination game in Table 1. Each agent used $\beta = 0.2$ and $\eta = 2 \times 10^{-3}$.

Two $RV_{\sigma(t)}$ agents always converged to a coordinated action profile, sometimes even to action profiles $\langle R_1, C_1\rangle$ and $\langle R_3, C_3\rangle$ depending on the starting point, in both games. Figure 1 show both $RV_{\sigma(t)}$ agents converging to their 3rd actions (i.e., R_3, C_3) even when each starts close to its desired policy and uses its desired equilibrium. Essentially, one agent learns to compromise while the other maximizes its payoff, reaching a coordinated profile. Which agent compromises depends more on the starting policies than the learning dynamics.

GIGA and GIGA-WoLF converge to either coordinated action profiles or to mixed equilibria e.g., $\langle[1/51, 0, 50/51], [50/51, 0, 1/51]\rangle$. This happens when they start close to their preferred policies like the $RV_{\sigma(t)}$ agents. Figure 2 show the sum

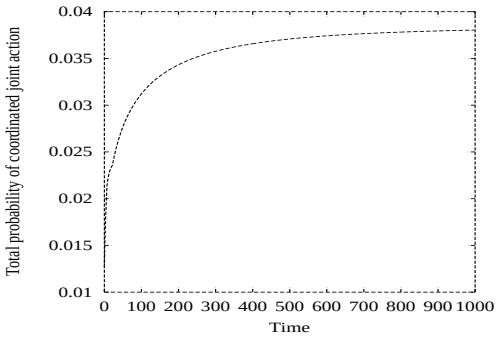


Figure 2: The sum of the joint probabilities of diagonal actions when two GIGA-WoLF agents start at $\langle[0.005, 0.005, 0.99], [0.99, 0.005, 0.005]\rangle$, in the coordination game in Table 1. Each agent used $\eta_1 = 2 \times 10^{-3}$.

of the coordinated joint actions of two GIGA-WoLF agents, i.e., $\sum_s \pi_1^t(s)\pi_2^t(s)$, which converges to the inner product of the mixed equilibrium profile that the agents converge to (in this case 0.038446751). If the agents had converged to a purely coordinated profile, then $\sum_s \pi_1^t(s)\pi_2^t(s)$ would have converged to 1. Thus figure 2 also shows the low probability of coordination of the resulting profile (3.8%). This not only means that the agents would play uncoordinated actions most of the times, but also that even the expected payoffs of *both* agents ($\approx$ 1.96 for both) are lower than at any coordinated action profile. Note that GIGA-WoLF converges to this same undesired equilibrium whether it uses $\eta_1 = 1$ or $\eta_1 = 2 \times 10^{-3}$ (like $RV_{\sigma(t)}$), where $\eta_t = \eta_1/\sqrt{t}$.

5. ACKNOWLEGDEMENT

We thank Mike Bowling for helping with the code of GIGA-WoLF. This work was supported in part by Grants from Army Research Office W911NF-05-1-0377 and Louisiana BOR LEQSF(2002-05)-RD-A-29.

6. REFERENCES

[1] B. Banerjee and J. Peng. Performance bounded reinforcement learning in strategic intercations. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-04)*, pages 2 – 7, San Jose, CA, 2004. AAAI Press.

[2] B. Banerjee and J. Peng. Convergence of no-regret learning in multiagent systems. In *Proceedings of the First International Workshop on Learning and Adaptation in Multiagent Systems (LAMAS)*, Utrecht, The Netherlands, 2005. Held in conjunction with AAMAS-05.

[3] M. Bowling. Convergence and no-regret in multiagent learning. In *Proceedings of NIPS 2004/5*, 2005.

[4] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215 – 250, 2002.

[5] V. Conitzer and T. Sandholm. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In *Proceedings of the 20th International Conference on Machine Learning*, 2003.

[6] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, Washington DC, 2003.

Lenient Learners in Cooperative Multiagent Systems

Liviu Panait
lpanait@cs.gmu.edu

Keith Sullivan
ksulliv@cs.gmu.edu

Sean Luke
sean@cs.gmu.edu

Department of Computer Science, George Mason University
4400 University Drive, MSN 4A5, Fairfax, VA 22030, USA

ABSTRACT

In concurrent learning algorithms, an agent's perception of the joint search space depends on the actions currently chosen by the other agents. These perceptions change as each agent's action selection is influenced by its learning. We observe that agents that show lenience to their teammates achieve more accurate perceptions of the overall learning task. Additionally, lenience appears more beneficial at early stages of learning, when the agent's teammates are merely exploring their actions, and less helpful as the agents start to converge. We propose two multiagent learning algorithms where agents exhibit a variable degree of lenience, and we demonstrate their advantages in several coordination problems.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Experimentation

Keywords

Multiagent Learning, Cooperation, Coordination

1. INTRODUCTION

This paper focuses on applications where multiple agents concurrently learn how to better interact with one another. Imagine a simple scenario where two agents learn to coordinate. The task is for the agents to each independently choose one action with the goal of maximizing the joint reward that they receive. Figure 1 illustrates a search space of joint rewards for a simple two-agent domain. The figure shows two peaks of different sizes. The lower peak represents a globally-suboptimal solution, and the wide coverage of that peak implies that solution quality changes only a little when either agent chooses a slightly different action. The higher peak represents the global optimum; its smaller coverage implies that the solution quality may change rapidly if either agent chooses a slightly different action.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

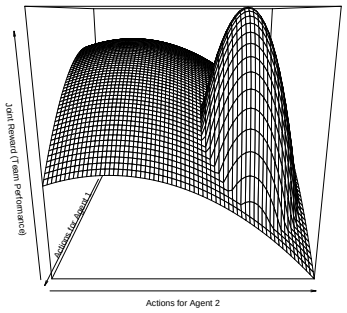


Figure 1: A bimodal search space for the possible rewards received by a two-agent team. Wider peaks may attract many search trajectories, even though such peaks may be globally suboptimal.

In concurrent multiagent learning, each agent is usually afforded only a partial glimpse of the search space. Specifically, each agent can only detect the rewards it receives for its own actions. However, these rewards also depend on the actions chosen by its teammate, and as a consequence, an agent's perception of the search space depends on what actions its teammate is currently choosing.

An agent's perception of the joint space is additionally affected by the approach it uses to aggregate the rewards obtained for each of its actions (when combined with different actions for the teammate). For example, Figure 2 shows different perceptions of the joint search space in Figure 1 for two aggregation methods. We observe that if an agent estimates the quality of its action as the *average* reward obtained when combined with multiple actions chosen by its teammate, the agent fails to perceive the potential of actions corresponding with the globally optimal peak. On the other hand, an agent that shows *lenience* towards its teammate by assessing the quality of its actions as the *maximum* reward received over multiple actions chosen by its teammate (thus ignoring the lower rewards usually associated with miscoordinations) perceives the higher potential of actions corresponding to the global optimum. If the goal is to discover globally optimal solutions, the agents should be able to tell that actions associated with such solutions are superior to others. In this paper, we focus on lenient learning as an approach to doing this.

We are particularly interested in *varying* the degree of lenience as time passes. As a consequence, we propose two algorithms that decrease the learners' degree of lenience with time. There are several reasons to vary the degree of lenience an agent exhibits. While an extremely high degree of lenience may be useful at early stages of learning to identify promising actions, ignoring many rewards may also be wasteful (in terms of evaluations) at later stages of learning once the agents start to converge. Additionally, ignoring rewards

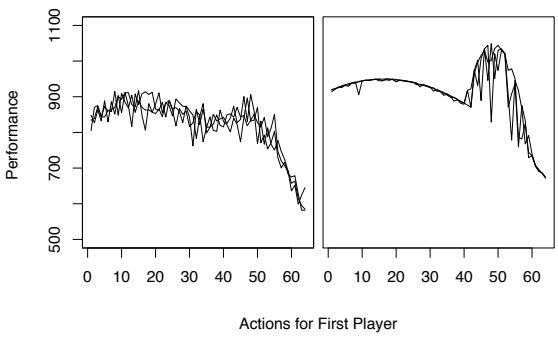


Figure 2: Two possible perceptions of the domain in Figure 1, for random actions of the teammate. (left) The average of ten joint rewards. (right) The maximum of ten joint rewards.

as the agents start to converge may be harmful to estimating the actual expected value in domains where agents receive noisy reward information. Finally, lenience may interfere with the learners' attempts to choose one of multiple equally-good solutions, if several exist.

2. LENIENT COEVOLUTION

Cooperative coevolutionary systems [6] are variants of evolutionary computation (a stochastic optimization technique) which apply multiple parallel learners (optimization processes) to work jointly on different aspects of the problem. This makes them a good fit for multiagent learning by decomposing the joint problem into several, likely intertwined, individual agent subproblems.

A standard approach to applying cooperative coevolutionary algorithms (CCEAs) to multiagent learning assigns each agent its own set (*population*) of higher-rewarded actions. The quality (*fitness*) of an action is determined by testing it in combination with actions from the other agent (as sampled from its current population either randomly or based on their performance during the past evaluation phase). When combined with these actions (called *collaborators* in coevolution parlance), an agent's action receives a reward equal to the joint reward for the entire team of agents. The fitness of the candidate action is then computed by aggregating multiple such rewards (e.g., by taking the average or the maximum, similar to the process used for Figure 2). Aside from this collaborative assessment, each agent follows its own independent evolutionary process in parallel with other agents.

Most cooperative coevolutionary algorithms assume that each action is evaluated as the maximum joint reward obtained with a fixed number of collaborators [8]; of them, one is usually the best action at the previous learning stage (*generation*), and the others are chosen at random. In terms of our earlier discussion, using the maximum joint reward translates into a constant level of lenience: the quality of the action is assessed by ignoring lower rewards. Given the observed variance in the impact that lenience can have on an agent's perception, we argue that a varying degree of lenience might be beneficial for CCEAs, especially when given fixed computation capabilities.

To do this, we experiment with a trivial ad-hoc setting: the fitness of each individual is assessed as the maximum reward obtained with ten collaborators for the first five generations; after that, only the maximum reward with *two* collaborators is used. This implies that the degree of lenience varies with time: the worst nine out of ten rewards will be ignored for each action in an agent's population during the first five generations, while only one out of two rewards will be ignored thereafter. While we found this trivial setting to be useful for this proof of concept investigation, readers with further interest in time-dependent collaboration schemes for CCEAs are referred to [5].

Experiments. To test our hypothesis that a variable degree of lenience is helpful, we applied CCEAs to four simple coordination games based on benchmark optimization problems with well-known properties: Two-Peaks (as in Figure 1), Rosenbrock, Griewangk, and Booth [7]. All domains were discretized such that each of the two agents had a set of 1024 actions. Each agent maintained a population of 32 actions. Agents kept unmodified their best action from one generation to another, and the remaining population of actions was created by mutating actions chosen via tournament selection of size 2 (two random actions were picked with replacement from the population, and the fitter of the two was selected). Mutation worked as follows: a coin was repeatedly tossed, and the action (an integer number) was increased or decreased (the direction chosen at random beforehand) until the coin came up heads, making sure it did not go outside the allowed bounds. The coin was biased such that it came up heads with probability 0.05. One of the collaborators was always set to the best action from the other agent's population at the previous generation; the others were chosen by a tournament selection of size 2.

We fixed the budget to 17600 evaluations of joint rewards. When choosing this budget, we felt that too small of a value might prevent differentiations among the algorithms because they would not be allowed to search enough. Similarly, too large of a budget might diminish the differences between methods that waste evaluations and methods that use them effectively. The value we chose seemed to be a good compromise.

The experiments were performed using the ECJ library [4], and they involved 250 runs per method. Performance was computed as the average of the best performing pair of actions (one per each agent) at the last generation. The results indicated that the CCEA with a variable degree of lenience was significantly better in all four problem domains than CCEAs involving a fixed degree of lenience (statistical significance was verified using t-tests assuming unequal variances at 95% confidence).

3. LENIENT MULTIAGENT REINFORCEMENT LEARNING

Drawing inspiration from dynamic programming concepts, reinforcement learning (RL) methods update the estimates of utilities for performing actions in various states of the environment, or for being in those states themselves. These utilities are used for both the exploration of the space, and for the exploitation of the agent's knowledge about the environment. As the memory requirements of traditional RL grow exponentially with the number of agents, multiagent reinforcement learning reduces the memory consumption by decomposing the utilities of joint actions into per-agent utilities of actions. We assume for simplicity that the environment has a single state, and we only focus on computing the utility of choosing different actions; this is similar to the analysis of multiagent RL in [1, 2, 3].

The proposed lenient multiagent reinforcement learning algorithm (LMRL) is based on the following idea: if at early stages of learning an agent receives rewards $r_1, r_2, ..., r_k$ when choosing action a_1 at various times, the agent only updates the utility of a_1 based on the maximum of $r_1, r_2, ..., r_k$, ignoring the others. The reason for this is that those rewards were obtained while the other learning agent selected some actions $b_1, ..., b_k$, most of which we poor choices to begin with and will be abandoned by the other

agent in the future due to low utility. But later, after each agent has largely converged to one or a few actions, it becomes more important to achieve accurate utility estimates. The agents therefore will decrease their degree of lenience to one another by ignoring fewer of the lower utilities.

To implement this idea, an agent always updates the utility of the action if the current reward exceeds the utility of that action. Agents associate a temperature with each action, and the level of lenience is inversely proportional to the temperature: the temperature is initially high and agents are thus more likely to ignore lower rewards. The temperatures of actions decrease as those actions are selected, and as a consequence the agents become more likely to incorporate the lower rewards into the utility estimates. As agents have non-zero probabilities of selecting an action at each time step, there is also a small (0.01) probability of ignoring small rewards at all times. Agents choose actions via Boltzman selection.

`Lenient Multiagent RL`
Parameters
$MaxTemp$: maximum temperature
α: temperature multiplication coefficient
β: exponent coefficient
δ: temperature decay coefficient
λ: learning rate
N: number of actions
Initial Settings
For each action i
$U_i =$ random value between 0 and 0.001
$Temp_i = MaxTemp$
Algorithm
Repeat
// Action Selection
$MinTemp = 10^{-6} + \min_{i=1}^{N} Temp_i$
$W_i = \frac{e^{\frac{U_i}{MinTemp}}}{\sum_{j=1}^{N} e^{\frac{U_j}{MinTemp}}}$
Use probability distribution W_1, ..., W_N to select action i
$Temp_i = Temp_i * \delta$
// Utility Update
Perform action i and observe reward r
$RandVal =$ random value between 0 and 1
If $(U_i \leq r)$ or $\left(RandVal < 10^{-2} + \beta^{-\alpha * Temp_i}\right)$ Then
$U_i = \lambda * U_i + (1-\lambda) * r$

Experiments. We tested LMRL in the Climb and Penalty domains introduced in [1], and in the two stochastic variations of the Climb domain discussed in [2]. We performed a preliminary sensitivity study for the parameters: as a result, we set $MaxTemp = 500$, $\alpha = 2$, $\beta = 2.5$, $\delta = 0.995$, and $\lambda = 0.95$. The agents spent 7500 moves learning. We compared LMRL's performance with that of FMQ [2] and with a straightforward application of RL to concurrent learning (also in [2]). We ran each algorithm 10000 times in each of the four problem domains.

The lenient multiagent RL algorithm consistently converged to the global optimum in the Climb, Penalty, and Partially-Stochastic Climb domains. According to [2], this is equivalent to the performance of FMQ in these problem domains, and it is also significantly better than the performance of traditional Q-learning approaches, as well as to an algorithm previously proposed in [3]. However, the lenience also helped the agents learn the global joint action in the Fully-Stochastic Climb domain in more than 93.5% of the runs. Contrast this to FMQ's poor performance in this domain as mentioned in [2]. We likewise found that FMQ converged to the global optimum solution in only around 40% of runs in this difficult domain, despite an extensive sensitivity study for parameter values.

4. CONCLUSIONS

This paper argues that multiple agents that learn concurrently can benefit from showing lenience to each other, especially during early interactions. We illustrated this concept with perceptions of the joint search space that each agent would perceive during different stages of learning. We then extended two popular multiagent learning algorithms, namely cooperative coevolution and multiagent reinforcement learning, to include lenience in the agents' decision processes, and we showed the superiority of these extensions in several coordination games.

5. REFERENCES

[1] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 746–752, 1998.

[2] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 2002.

[3] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 535–542. Morgan Kaufmann, 2000.

[4] S. Luke. ECJ 10: An Evolutionary Computation research system in Java. Available at http://www.cs.umd.edu/projects/plus/ec/ecj/, 2003.

[5] L. Panait and S. Luke. Selecting informative actions improves cooperative multiagent learning. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi Agent Systems – AAMAS-2006*, 2006.

[6] M. Potter and K. De Jong. A cooperative coevolutionary approach to function optimization. In Y. Davidor and H.-P. Schwefel, editors, *Proceedings of the Third International Conference on Parallel Problem Solving from Nature (PPSN III)*, pages 249–257. Springer-Verlag, 1994.

[7] H. Schwefel. *Evolution and Optimum Seeking*. John Wiley and Sons, New York, 1995.

[8] R. P. Wiegand, W. Liles, and K. De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In E. Cantu-Paz *et al*, editor, *Proceedings of the 2001 Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1235–1242, 2001.

Can Good Learners Always Compensate for Poor Learners?

Keith Sullivan Liviu Panait Gabriel Balan Sean Luke
{ksulliv, lpanait, gbalan, sean}@cs.gmu.edu

Department of Computer Science, George Mason University
4400 University Dr, MSN 4A5, Fairfax, VA 22030 USA

ABSTRACT

Can a good learner compensate for a poor learner when paired in a coordination game? Previous work presented an example where a special learning algorithm (FMQ) is capable of doing just that when paired with a specific less capable algorithm even in games which stump the poorer algorithm when paired with itself. We argue that this result is not general. We give a straightforward extension to the coordination game in which FMQ cannot compensate for the lesser algorithm. We also provide other problematic pairings, and argue that another high-quality algorithm cannot do so either.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Algorithms

Keywords

Reinforcement Learning, Multiagent Systems, Cooperative Games

1. INTRODUCTION

Concurrent learning is a subset of cooperative multiagent learning where the overall problem is divided into simpler subproblems such that each agent representing one subproblem, explores its space of actions with little or no control over its teammate's actions. Given concurrent learners, we might ask: what if each learner used a different algorithm? This is not implausible: for example, an agent (on the web, say) may not have control over the other learning agents. But while recent research on learning in competitive games has addressed heterogeneous learners [1], cooperative learning literature has largely assessed homogeneous learners.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

We are aware of only one paper that focuses on concurrent heterogeneous cooperative learners: Kapetanakis and Kudenko analyze different combinations of a traditional reinforcement learning algorithm and an extension called FMQ [6]. The authors report that a two-agent team using the traditional algorithm cannot consistently learn the globally optimal solution. However, the optimal solution is achieved when both agents use FMQ. Moreover, an agent using FMQ can help another agent using the traditional algorithm to learn the global optimum in several coordination games. The authors thus conclude that a "smart" learner (FMQ) can still be successful even when it must work with less smart learners (the traditional algorithm).

We will argue that this is not necessarily so. We extend the research in [6] by including a difficult coordination game, as well as two new algorithms. One of these algorithms is a modified evolutionary algorithm: this extends the analysis to combinations with agents that use *entirely* different learning techniques, not just variations of the same learning method. The results of the experiments suggest the opposite conclusion from that in [6]: in difficult domains, good results are usually obtained only if both agents are "smart" enough. That is because it takes both agents to converge to an optimal solution, and poor learners may force "smart" ones to converge to suboptima.

2. SINGLE-STAGE COOPERATIVE GAMES

Markov decision processes (MDPs) are widely used in multiagent reinforcement learning to account for the presence of other agents in the environment [4, 8]. Single-stage cooperative games are a variation of MDPs where all agents receive the same reward [2]. More specifically, each agent independently chooses an action from its action set, and the actions from all the agents are combined into a *joint action*. All agents receive the same reward or penalty depending on the joint action. For example, using the Climbing game as shown in Table 1(a), if agent 1 chooses action b and agent 2 choose action a, then both agents receive a reward of -30. This process of choosing a joint action is repeated until (hopefully) the agents learn to select better actions due to past interactions. We assume that agents do not explicitly communicate or observe teammates' actions; the only feedback mechanism is the reward received for the agent's action.

Table 1 shows four single-stage cooperative games: the Climbing game and the Penalty game introduced in [2], and the partially- and fully-stochastic variations of the Climbing game as proposed in [5]. Note that in the partially- and fully-stochastic games, when $p = 0.5$, the average reward for a joint action is the same as the reward for the same joint action in the regular Climbing game. In

Agent 1 \ Agent 2	a	b	c
a	11	-30	0
b	-30	7	6
c	0	0	5

(a)

Agent 1 \ Agent 2	a	b	c
a	10	0	k
b	0	2	0
c	k	0	10

(b)

Agent 1 \ Agent 2	a	b	c
a	11	-30	0
b	-30	14, 0	6
c	0	0	5

(c)

Agent 1 \ Agent 2	a	b	c
a	10, 12	5, -65	8, -8
b	5, -65	14, 0	12, 0
c	5, -5	5, -5	10, 0

(d)

Table 1: Joint reward matrices for the Climbing Game (a), Penalty Game (b), Partially Stochastic Game (c), and the Fully Stochastic Game (d). In the stochastic games, the first reward is returned with probability p, and the second reward is returned with probability $1-p$.

the regular, partially- and fully-stochastic Climbing games, the optimal joint action is (a,a). In the Penalty game, the known penalty, $k \leq 0$, produces three optimal joint actions — (a,a), (b,b), (c,c) — of which (a,a) and (c,c) are preferred since they have a higher reward. Researchers have used these games extensively to highlight the advantages of certain multiagent learning algorithms (for example, [2, 5, 7]).

Each of these games is challenging due to miscoordination penalties. The Climbing game has a severe penalty for choosing action a when the other agent chooses action b. However, there are no major miscoordination penalties associated with action c, potentially tempting the agents. The Penalty game introduces an additional miscoordination issue due to the presence of multiple optimal joint actions. While smaller values of k will make the individual actions a and c more attractive, there is still the problem of ensuring that the other agent will choose the same optimal action.

The stochastic variations of the Climbing game add more complexity due to the noisy reward function. As agents cannot perceive their teammates' actions, the different rewards they observe for the same action may be due to either (1) the other agent experimenting with various actions, or (2) the noisy reward function. While the highest reward of 14 is sometimes achieved when both agents choose action b, they need to learn to separate the effects of (1) and (2), and to realize that the average reward for the joint action (b,b) is lower than that of (a,a).

3. LEARNING ALGORITHMS

We experiment with four learning algorithms: three variations of reinforcement learning, and a genetic algorithm with a novel evaluation procedure. Two of these algorithms are not new: reinforcement learning and FMQ have been previously analyzed, in particular in [5]. We chose reinforcement learning as a standard benchmark, while FMQ appears to be one of the stronger algorithms to date. We devised the other two algorithms based on the notion of lenience of an agent towards its teammates.

3.1 Lenient Multiagent Reinforcement Learning

In an accompanying paper, we propose another RL algorithm, Lenient Multiagent Reinforcement Learning (LMRL), which selectively updates the utilities of actions based on *some* of the rewards. It is implemented as follows. We always update the utility of the action if the current reward exceeds the utility of the action. Otherwise, the utility is updated with a probability based on a current per-action *temperature*. If the temperature associated with an action is high, then agent is "lenient" towards low-reward pairings and so it does not update its utility to reflect them. At a lower temperature, the utility includes low-reward pairings with greater frequency.

The temperature of an action is decreased slightly every time that action is selected. As a consequence, actions that have been chosen more often have their utilities updated more often as well, while the utilities for actions that have been chosen rarely are mainly updated in response to higher rewards. This initially leads to an overoptimistic evaluation of the utility of an action. An agent may thus be temporarily fooled into choosing suboptimal actions. However, the utilities of such actions will decrease with time, and the agent is more likely to end up choosing the optimal action. There is also a small (0.01) probability of ignoring small rewards at all times: we found this to be important in our experimental setup because the agents have non-zero probabilities of selecting any action at each time step. Aside from these enhancements, the algorithm follows a traditional RL approach, including the Boltzman action selection based on the utility of each actions.

3.2 Lenient Evolutionary Algorithm

Evolutionary algorithms (EAs) are stochastic search techniques inspired by natural evolution [3]. EAs maintain a set of samples in the search space (referred to as a "population" of "individuals"). Each such individual is assigned a "fitness" (the quality of the individual). EAs then form a new population by repeatedly selecting, copying and modifying the highly fit individuals in the current population. The new population replaces the old one, and the cycle of fitness assessment and breeding continues until a termination criterion is met. Each iteration of this cycle is known as a "generation." See [3] for a more detailed discussion of evolutionary algorithms.

Cooperative Coevolutionary Algorithms (CCEAs) apply EAs to concurrent learning processes. A CCEA employs multiple (for our purposes, two) populations, each evolving independently. CCEAs pair populations to evaluate them together. We propose a new EA that, like CCEAs, evaluates individuals (representing actions) in combination with actions not just the other populations, but from other learning algorithms (such as RL or another EA).

This EA also allows an agent (individual) to show varying degrees of lenience to its teammate. For this purpose, the entire population is cloned multiple times and shuffled, and each clone is evaluated with an action chosen by the teammates. As the teammate is also co-adapting, the particular action it chooses may change at any time. The extra clones of the population are required only for evaluations and may be discarded afterwards. After each clone of an action receives multiple rewards, these rewards are aggregated into a single fitness for the original action. This fitness is computed as the average of the better K rewards that were obtained. Varying degrees of lenience are implemented by using the average of fewer of the better rewards at early generations (thus ignoring more of the rewards observed for an action). Initially, the best reward is used. As learning progresses, we compute the fitness as the average of more and more "top" rewards, thus reducing the lenience towards the teammate.

4. EXPERIMENTAL RESULTS

We ran each combination of learning algorithms in the four cooperative games. Experiments consisted of 30 trials of 1000 runs each. Each run lasted for 7500 joint action selections and their re-

	RL	FMQ	LMRL	EA
RL	468.5	998.4	0.8	218.5
FMQ	–	999.8	998.1	419.9
LMRL	–	–	998.0	214.7
EA	–	–	–	303.8

Table 2: Average number of iterations (out of 1000) that converged to the joint action for the partially stochastic game.

	RL	FMQ	LMRL	EA
RL	464.4	3.3	2.6	230.9
FMQ	–	141.5	4.3	108.0
LMRL	–	–	928.5	235.1
EA	–	–	–	197.0

Table 3: Average number of iterations (out of 1000) that converged to the joint action for the stochastic game.

wards. We set $k = -10$ in the Penalty game. For all the reinforcement learning algorithms, temperature decayed as $T = T * 0.9995$ and the learning rate is 0.95. For FMQ, we set $c = 10$; LMRL used a temperature multiple coefficient of 2 and a exponent coefficient of 2.5. For the lenient evolutionary algorithm, we used a population of size 10, and the evolution lasted for 150 generations; we also set $C = 5$ clones per population, and K started at 1 and increased every 15 generations until reaching the maximum value of 5. (See our accompanying paper for a more detailed discussion of the parameters.) The lenient EA employed selection via binary tournament, and breeding randomized the action with probability 0.001. At the end of each run we determined if the converged joint action was optimal. We then computed the number of runs (out of 1000) that converged to the joint action, and averaged over 30 trials.

In the Climbing game, all but three learning teams discovered the optimal action more than 91% of the time. The reasons: RL-RL's use of all rewards caused it to be attracted occasionally to (b,c). RL-LMRL in turn was attracted to (c,c) and EA-EA to (b,b). Overall, FMQ and LMRL were the best learners in the Climbing domain, and they also did well when combined with one another. FMQ had a slight edge over LMRL as it also performed very well when teamed with RL. All teams discovered the optimal strategy more than 98% of time in the Penalty game. This improvement in performance was partly due to the fact that the Penalty game has twice the number of optimal solutions to which learning may converge.

The results for the partially stochastic Climbing game (Table 2) were similar (although more extreme) to the ones in the deterministic Climbing game. FMQ and LMRL were again the "smarter" algorithms: they found the optimal joint action in almost all runs when paired with themselves or each other. FMQ was also able to help RL converge to the optimum, but LMRL could achieve that in almost any run. The EA's performance deteriorated significantly across the board.

So far, FMQ has done well when partnered with most other algorithms. But Table 3 shows a different result. Only one pair (LMRL-LMRL) efficiently solved this problem. But more importantly, *every method but EA did best by far when paired with itself.* In general, heterogeneous parings performed very poorly! Notably, while FMQ helped RL in the partially stochastic game, in the fully stochastic game FMQ *hinders* RL compared to RL-RL (and for that matter, FMQ-FMQ). That is to say, the claim in [6] that the "smarter" FMQ algorithm can help poorer RL find the optima must change to "FMQ helps RL to converge to the optimum *for this problem domain*, but hinders it in others."

5. CONCLUSIONS

We extended the work in [6] to include two new learning algorithms that exhibit lenience toward teammates, as well as a difficult coordination game characterized by stochastic rewards for every joint action. The experiments indicate that only the lenient multiagent reinforcement learning algorithm can achieve near-optimal performance, and only when paired with itself. A good learner cannot compensate if its teammate converges to a suboptimal action. Contrary to the findings in [6], we find that a pair of traditional RL algorithms performs better than both an FMQ-FMQ pair, as well as a combination of an FMQ and an RL learner. It is *not* the case that FMQ, or even the often-better LMRL, can compensate for a mismatched teammate algorithm.

The issue that remains is: are there well-defined classes of problems and subsets of learning algorithms for the teammate with which a given learning algorithms works well? For example, exactly where and why does FMQ pair well with various learners? This is not an easy question to answer, but investigation along these lines may reveal more clues about the relationship classes of algorithms have with one another, and why some features of algorithms may be at odds with features of other algorithms. Until then we must recommend a homogeneous approach to multiagent learning, if given the choice.

6. REFERENCES

[1] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[2] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of National Conference on Artificial IntelligenceAAAI/IAAI*, pages 746–752, 1998.

[3] K. De Jong. *Evolutionary Computation: A unified approach*. MIT Press, 2006.

[4] J. Hu and M. Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242–250. Morgan Kaufmann, San Francisco, CA, 1998.

[5] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI02)*, 2002.

[6] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in heterogeneous cooperative multi-agent systems. In *Proceedings of the Third Autonomous Agents and Multi-Agent Systems Conference (AAMAS 2004)*, 2004.

[7] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 535–542. Morgan Kaufmann, San Francisco, CA, 2000.

[8] M. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 157–163, New Brunswick, NJ, 1994. Morgan Kaufmann.

Learning Regularities with a Constructivist Agent

Filipo Studzinski Perotto
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Av Bento Gonçalves 9500 – Porto Alegre, Brasil
+(55) 51 3217 3301
fsperotto@inf.ufrgs.br

Luís Otávio Álvares
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Av Bento Gonçalves 9500 – Porto Alegre, Brasil
+(55) 51 3316 6843
alvares@inf.ufrgs.br

ABSTRACT
This paper presents an agent learning architecture based on the constructivist approach. After a brief introduction, we show details about the proposed model. Firstly, we explain how the agent represents the knowledge. We introduce an innovative learning method, inspired by the constructivist description of the human learning process. The kind of environment regularities that can be discovered by the method is analyzed. Finally, we discuss how an extension of this agent architecture can lead to the construction of abstract concepts.

Categories and Subject Descriptors
I.2.6 [**Learning**]

General Terms
Algorithms, Theory

Keywords
Constructivist AI, Inductive Learning, Agent Architecture

1. INTRODUCTION
The Constructivist Artificial Intelligence comprises all works on this science that refer to the Constructivist Psychological Theory [6]. The constructivist conception of intelligence was brought to the field of Artificial Intelligence in the early 90's, by the Drescher's pioneer work [2]. Relevant related works have been presented in [1], [3] and [7]. This paper proposes an agent learning architecture based on the constructivist paradigm. Our model differs from the previous works because it is guided by direct induction, instead of statistical correlation. This approach has two advantages: reduces computational costs and makes it easier to find high-level deterministic regularities. A previous description of our research has been presented in [5].

2. KNOWLEDGE REPRESENTATION
Our constructivist agent tries to construct knowledge to represent perceived regularities occurring in the environment. This representation is structured in a set of schemas. Each schema represents some regularity checked by the agent during its interaction with the world. A schema is composed of three vectors: context, action and expectation. The context and expectation vectors have the same length, and each of their elements is linked with one sensor. The action vector has their elements linked with the effectors.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

In a schema, the context vector represents the set of equivalent situations where the schema is applicable. The action vector represents a set of similar actions that the agent can carry out in the environment. The expectation vector represents the expected result after executing the given action in the given context. Each element vector can assume any value in a discrete interval defined by the respective sensor or effector. In addition, some elements can undertake an undefined value. For example, an element linked with a binary sensor must have one of three values: true, false or undefined (represented, respectively, by '1', '0' and '#'). The undefined value generalizes the schema because it allows ignoring some properties to represent a set of situations. For example, a schema which has the context vector = (100#) is able to assimilate the situations (1000) and (1001).

There is compatibility between a schema and a certain situation when the schema's context vector has all defined elements equal to those of the agent's perception. Compatibility does not compare the undefined elements. This strategy gives more flexibility when the agent needs to represent a set of world states as a same situation. So, there is no need of considering each possible perception combination, what reduces the problem complexity. The use of undefined values, and the consequent capability to create generalized representations to similar situations, enables the construction of a tree.

Each node in that tree is a schema, and relations of generalization and specialization guide its topology. The root node represents the most generalized situation, which has the context and action vectors completely undefined. Adding one level in the tree is to specialize one generalized element, creating a branch where the undefined value is replaced by different defined values. This specialization occurs either in the context vector or in the action vector. The structure of the schemas and their organization as a tree are presented in Figure 1.

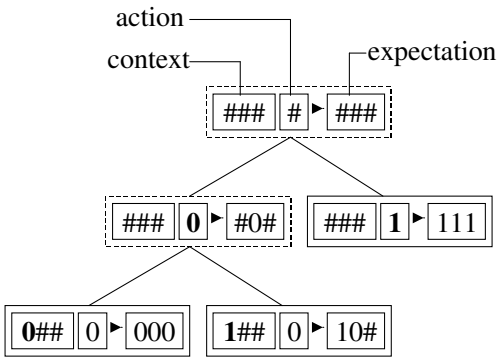


Figure 1. Schemas Tree. Each node is a schema composed of three vectors: context, action and expectation. The leaf nodes are decider schemas.

The context in which the agent is at a given moment (perceived through its sensors) is applied in the tree, exciting all the schemas that have a compatible context vector. This process defines a set of excited schemas, each one suggesting a different action to do in the given situation. The cognitive system chooses one to activate (based on an emotional value attributed to its expectations). Then the schema's action is performed (through the agent's effectors). This algorithm always chooses the compatible schema that has the most specific context, called "decider schema". Appended on each decider, there is a representation of the specific really experimented situations that happened in the past.

3. LEARNING METHODS

The learning process happens through the refinement of the set of schemas. The agent becomes more adapted to its environment as a consequence of that. After any situation, the agent's cognitive system checks if the result (context perceived at the instant following the action) is according to the expectation of the activated schema. If it fails in its prediction, the error between the result and the expectation serves as parameter either to correct the tree or to adjust the schema. If we consider context, action and expectation as a triple search space, then the learning problem is to induce a set of schemas in this search space that represents the environment regularities.

Our learning method combines top-down and bottom-up strategies. The context and action vectors are put together and this concatenated vector identifies the node in the schemas tree. The schemas tree grows up using the top-down strategy. The agent has just one initial schema. This root schema has the context vector completely general (without any differentiation) and expectation vector totally specific (without any generalization), created at the first situation experienced, according the result directly observed after the action. The context vector will be gradually specialized and differentiated. In more complex environments, the number of features the agent senses is huge, and, in general, only a few of them are relevant to identify the situation. The top-down strategy is better in this kind of environment, because it is a shorter way to begin with an empty vector and searching for these few relevant features, than to begin with a full vector and having to eliminate a lot of not useful elements. The expectation vector is equivalent to labels on the leaf nodes of the tree (called decider schemas).

The evolution of expectations uses the bottom-up strategy. Initially all different expectations are considered as different classes, and they are gradually generalized and integrated with others. The agent has two alternatives when the expectation fails. In a way to make the knowledge compatible with the experience, the first alternative is to try to divide the scope of the schema, creating new schemas with more specialized contexts. Sometimes it is not possible, then the only way is to reduce the schema expectation.

Three basic methods compose the learning function, namely: *differentiation*, *adjust* and *integration*. Differentiation is a necessary mechanism because a schema responsible for a too general context cannot have precise expectations. If a general schema does not work well, the mechanism divides it into new schemas, differentiating them by some perceptive element of context vector. In fact, the differentiation method takes an unstable decider schema and changes it into a two level sub-tree. The parent schema in this sub-tree preserves the context of the original schema. Its branches sustain the children, which are the new decider schemas. These deciders have their context vectors more specialized than their parent. They attribute a value to some undefined element dividing the scope of the original schema, each one engaging itself in a part of the domain. In this way, the previous correct knowledge remains preserved, distributed in the new deciders, and the discordant situation is isolated and treated only in its specific context. Differentiation is the method responsible to construct the schemas tree. Each level of the tree represents the introduction of some constraint into the context vector to describe compatible situations. Figure 2 illustrates the differentiation process.

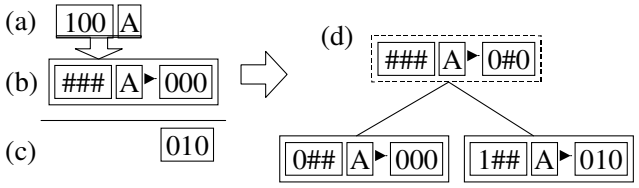


Figure 2. Differentiation method; (a) experimented situation and action; (b) activated schema; (c) real observed result; (d) sub-tree generated by differentiation.

The algorithm needs to choose what will be the differentiator element, and it could be from either the context vector or the action vector.

When some schema fails and it is not possible to differentiate it, then the cognitive system executes the adjust method. This method reduces the expectations of an unstable decider schema in order to make it more reliable. The algorithm simply compares the activated schema's expectation and the real result perceived by the agent after the application of the schema and changes the elements that are not compatible to each other for the undefined value '#'. As the mechanism always creates schemas with expectations totally determined and equal to the result of its first application, the walk performed by the schema is a reduction of expectations up to the point it reaches a state where only those elements that really represent the regular results of the action carried out in that context remain. Figure 3 illustrates that.

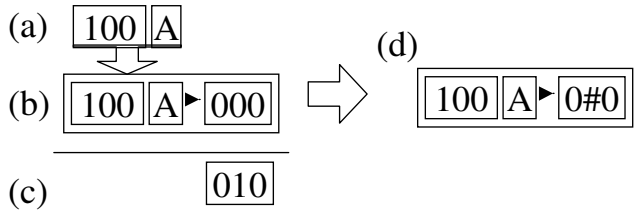


Figure 3. Adjust method; (a) experimented situation and action; (b) activated schema; (c) real observed result; (d) schema expectation reduction after adjust.

In this way, the schema expectation can change (and consequently the class of the situation represented by the schema), and the tree maintenance mechanism needs to be able to reorganize the tree when this change occurs.

Finally, successive adjustments in the expectations can reveal some unnecessary differentiations. When the cognitive system finds two schemas with similar expectations to approach different contexts, the integration method comes into action, joining these

schemas into only one, and eliminating the differentiation. The method operates as shown in figure 4.

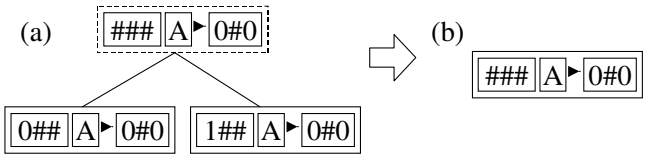


Figure 4. Integration method; (a) sub-tree after some adjust; (b) an integrated schema substitutes the sub-tree.

We have made some experiments in simple scenarios and exhaustive tests have shown that the agent ever converges to the expected behavior, constructing correct knowledge to represent the environment regularities, as well as the regularities of its body sensations, and also the regular influence of its actions over both. We may consider our results were successful, once the agent has learned about the consequences of its actions in different situations, avoiding emotionally negative situations, and pursuing those emotionally positive.

4. CONCLUSION AND FUTURE WORK

We have shown an agent architecture based on constructivist AI principles, which provides adaptive capability because the agent constructs its knowledge incrementally, by interaction with its environment. The agent has cognitive construction autonomy. We have presented a learning method composed by three functions: differentiation, adjust and integration. These functions operate over a schemas tree, which is transformed during the agent life in a way to describe the observed regularities of the world. Some experimental results have corroborated the mechanism ability to discover regularities and use this knowledge to adapt the agent behavior.

However, the learning method presented can be extended to be able to detect high-level regularities. Currently it can induce only stationary, deterministic, instantaneous and sensorimotor regularities. Therefore, we intend to introduce in our agent the following capabilities: temporal analysis, abstract concept creation, and probabilistic features. Furthermore, we want to improve the cognitive mechanism to chain the schemas, forming action sequences, and permitting the agent to create plans instead just reacting.

In the basic method, when some schema fails the first alternative is to differentiate it. If it is not possible, the mechanism repairs the schema by reducing its expectation. This alternative supposes that there is no regularity and the analyzed result is unpredictable. It is a possible explication to the fail, called "unpredictability hypothesis", but it is not necessarily true every time. In fact, there is no deterministic, instantaneous, sensorimotor regularity, but the agent must considering some alternative hypotheses, which represent other kind of regularities, and which can explain the fail.

First, the agent can postulate a “noise hypothesis”. It supposes that the error is caused by some perceptive noise, and it is a good explication to the schema’s fail if the error occurs with a very low frequency, and if the error is a random change in the element value. In this case, the best answer is to preserve the schema without modifications.

The second possible explication is the “stochastic regularity hypothesis” which says that a few different results are possible, each one with specific probabilities. The solution, if the hypothesis is right, is to create a special schema with stochastic expectations.

The third alternative is the “temporal context regularity hypothesis”. This resource permits the agent to perceive contexts distributed in the time, in addition to the capability of identifying regularities in the instantaneous perception. This hypothesis claims that is necessary to observe event sequences to explain the result of an action. It supposes that the result is a consequence of chained actions. The solution, in this case, is to utilize a special temporal schema, which improves the temporal window, analyzing more instants in the past to make its predictions.

The last option is the “abstract context regularity hypothesis”, which supposes that the error can be explained by considering the existence of some abstract property that is capable of differentiating the situation, but it cannot be directly perceived by the sensors. This abstraction requires the agent postulating the existence of a non-sensorial element. An abstract property in the agent's mind could represent two kinds of environment features: hidden properties (in partially observed worlds) and sub-environment identifiers (in non-stationary worlds).

With these new capabilities, the agent becomes able to overpass the sensorial perception constructing abstract concepts. By forming these new knowledge elements, the agent acquires new forms to structure the comprehension of its own reality in more complex levels.

After postulating these hypotheses, the agent needs to discover what of them is the correct one. It is not possible to do that a priori. The solution is to use a selectionist idea. During some time, the agent preserves all the hypotheses, and evaluates them. At the end, the best solution survives.

5. REFERENCES

[1] Chaput, H. 2004. The Constructivist Learning Architecture. PhD Thesis. University of Texas.

[2] Drescher, Gary. 1991. Mide-Up Minds: A Construtivist Approach to Artificial Intelligence. MIT Press.

[3] Holmes & Isbell. 2005. Schema Learning: Experience-based Construction of Predictive Action Models. In Advances in Neural Information Processing Systems, volume 17.

[4] Morrison, C. Oates, T. & King, G. 2001. Grounding the Unobservable in the Observable: The Role and Representation of Hidden State in Concept Formation and Refinement. In Working Notes of AAAI Spring Symposium

[5] Perotto, F. Vicari, R. Alvares, L. O. 2004. An Autonomous Intelligent Agent Architecture Based on Constructivist AI. AIAI. p.103-116

[6] Piaget, Jean. 1957. Construction of Reality in the Child. London: Routledge & Kegan Paul.

[7] Wazlawick, R. Costa, A. C. R. 1995. Non-Supervised Sensory-Motor Agents Learning. In: Artificial Neural Nets and Genetic Algorithms. New York: Springer-Verlag: 49-52.

Improving Reinforcement Learning with Context Detection

Bruno C. da Silva, Eduardo W. Basso, Filipo S. Perotto, Ana L.C. Bazzan, Paulo M. Engel
Instituto de Informática, UFRGS
Caixal Postal 15064 CEP 91.501-970
Porto Alegre, Brazil
{bcs,ewbasso,fsperotto,bazzan,engel}@inf.ufrgs.br

ABSTRACT

In this paper we propose a method for solving reinforcement learning problems in non-stationary environments. The basic idea is to create and simultaneously update multiple partial models of the environment dynamics. The learning mechanism is based on the detection of context changes, that is, on the detection of significant changes in the dynamics of the environment. Based on this motivation, we propose, formalize and show the efficiency of a method for detecting the current context and the associated model of prediction, as well as a method for updating each of the incrementally built models.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms

Keywords

Reinforcement learning, non-stationary environments, multi-model learning

1. INTRODUCTION

Dealing with non-stationary environments has always been a difficult task for learning algorithms. Specifically, non-stationarity affects standard reinforcement learning (RL) methods in a way that forces them to continuously relearn the policy from scratch. Our goal in this paper is to complement model-based RL algorithms with a method for dealing with a class of non-stationary environments in which the dynamics obeys a set of restrictions.

The non-stationary environments that we deal with consist of those composed by several different stationary dynamics. We call each type of dynamics a *context* and assume that it can only be estimated by observing the transitions and rewards. Our method creates models for these contexts and determines how to switch to the most appropriate model *on-the-fly*. Partial models have been used for this purpose by other authors, such as Choi [1] and Doya [2]. However, their approaches require a fixed number of models, and thus implicitly assume that the approximate number of different environment dynamics is known *a priori*. Since this assumption is not always realistic, our method tries to overcome it by incrementally building new models.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. MULTIPLE PARTIAL MODELS

Most RL problems are modeled as Markov Decision Processes (MDPs). MDPs are described by a set of states, $\mathcal{S}$, a set of actions, $\mathcal{A}$, a reward function $R(s,a) \rightarrow \Re$ and a probabilistic state transition function $T(s,a,s') \rightarrow [0,1]$. An experience tuple $\langle s,a,s',r \rangle$ denotes the fact that the agent was in state s, performed action a and ended up in s' with reward r. Given a MDP, the goal is to calculate the optimal policy π^*, which is a mapping from states to actions such that the future reward is maximized. In the next sections we assume that the reader is familiar with Q-Learning, a simple model-free RL algorithm, and with model-based Prioritized Sweeping. For more information, please refer to [3].

When dealing with non-stationary environments, both the model-free and the model-based RL approaches need to continuously relearn everything from scratch, since the policy which was calculated for a given environment is no longer valid after a change in dynamics. This causes a performance drop during the readjustment phase, and also forces the algorithm to relearn policies even for environment dynamics which have been previously experienced. Our hypothesis is that the use of multiple partial models makes the learning system capable of partitioning the knowledge into models, each model automatically assuming for itself the responsibility for "understanding" one kind of environment behavior. To each model, we assign a policy and a trace of prediction error of transitions and rewards.

3. RL WITH CONTEXT DETECTION

We propose that the creation of new models should be controlled by a continuous evaluation of the prediction errors generated by each partial model. In the following subsections we first describe how to learn contexts (i.e, estimate partial models), and then we show how to detect and switch to the most adequate model given a sequence of observations. Our method is called **RL-CD**, or Reinforcement Learning with Context Detection.

The class of non-stationary environments that we deal with is similar to the one studied by Hidden-Mode MDPs researchers [1]. We assume that the following properties hold: **1)** environmental changes are confined to a small number of contexts, which are stationary environments with distinct dynamics; **2)** the current context cannot be directly observed, but can be estimated according to the types of transitions and rewards observed; **3)** environmental context changes are independent of the agent's actions; and **4)** context changes are relatively infrequent.

3.1 Learning contexts

The mechanism for detecting context changes relies on a set of partial models for predicting the environment dynamics. A partial model m contains functions which estimate transition probabilities ($\hat{T}_m$) and rewards ($\hat{R}_m$). Standard model-based RL methods such as Prioritized Sweeping and Dyna can be used to compute the locally optimal policy $\pi_m(s)$.

Given an experience tuple $\varphi \equiv \langle s, a, s', r\rangle$, we update the current partial model m by adjusting its model of transition and rewards by $\Delta^{\hat{T}}_{m,\varphi}$ and $\Delta^{\hat{R}}_{m,\varphi}$, respectively. These adjustments are computed as follows:

$$\Delta^{\hat{T}}_{m,\varphi}(\kappa) = \frac{1}{\mathcal{N}_m(s,a)+1}\Big(\tau^{s'}_{\kappa} - \hat{T}_m(s,a,\kappa)\Big) \quad \forall \kappa \in \mathcal{S}$$

$$\Delta^{\hat{R}}_{m,\varphi} = \frac{1}{\mathcal{N}_m(s,a)+1}\Big(r - \hat{R}_m(s,a)\Big)$$

such that τ is the Kronecker Delta:

$$\tau^{s'}_{\kappa} = \begin{cases} 1, & \kappa = s' \\ 0, & \kappa \neq s' \end{cases}$$

The effect of τ is to update the transition probability $T(s,a,s')$ towards 1 and all other transitions $T(s,a,\kappa)$, for all $\kappa \in \mathcal{S}$, towards zero. The quantity $\mathcal{N}_m(s,a)$ reflects the number of times, in model m, action a was executed in state s. We compute $\mathcal{N}_m$ considering only a truncated (finite) memory of past M experiences:

$$\mathcal{N}_m(s,a) = min\Big(\mathcal{N}_m(s,a) + 1,\ M\Big) \tag{1}$$

A truncated value of $\mathcal{N}$ acts like a learning coefficient for $\hat{T}_m$ and $\hat{R}_m$, causing transitions to be updated faster in the initial observations and slower as the agent experiments more. Having the values for $\Delta^{\hat{T}}_{m,\varphi}$ and $\Delta^{\hat{R}}_{m,\varphi}$, we update the transition probabilities:

$$\hat{T}_m(s,a,\kappa) = \hat{T}_m(s,a,\kappa) + \Delta^{\hat{T}}_{m,\varphi}(\kappa), \quad \forall \kappa \in \mathcal{S} \tag{2}$$

and also the model of expected rewards:

$$\hat{R}_m(s,a) = \hat{R}_m(s,a) + \Delta^{\hat{R}}_{m,\varphi} \tag{3}$$

3.2 Detecting context changes

In order to detect context changes, the system must be able to evaluate how well the current partial model can predict the environment. Thus, an error signal is computed for each partial model. The *instantaneous error* is proportional to a *confidence value*, which reflects the number of times the agent tried an action in a state. Given a model m and an experience tuple $\varphi = \langle s, a, s', r\rangle$, we calculate the instantaneous error $e_{m,\varphi}$ and the confidence $c_m(s,a)$ as follows:

$$c_m(s,a) = \left(\frac{\mathcal{N}_m(s,a)}{M}\right)^2 \tag{4}$$

$$e_{m,\varphi} = c_m(s,a)\Big(\Omega(\Delta^{\hat{R}}_{m,\varphi})^2 + (1-\Omega)\sum_{\kappa\in\mathcal{S}} \Delta^{\hat{T}}_{m,\varphi}(\kappa)^2\Big) \tag{5}$$

where Ω specifies the relative importance of the reward and transition prediction errors for the assessment of the model's quality. Once the instantaneous error has been computed, the trace of prediction error E_m of the partial model is updated:

$$E_m = E_m + \rho\Big(e_{m,\varphi} - E_m\Big) \tag{6}$$

where ρ is the adjustment coefficient for the error.

The error E_m is updated after each iteration for every partial model m, but only the active model is corrected according to equations 2 and 3. A plasticity threshold λ is used to specify until when a partial model should be adjusted. When E_m becomes higher than λ, the predictions made by the model are considered sufficiently different from the real observations. In this case, a context change is detected and the model with lowest error is activated. A new partial model is created when there are no models with trace error smaller than the plasticity. The mechanism starts with only one model and then incrementally creates new partial models as they become necessary. Pseudo-code for RL-CD is presented in algorithm 1.

Algorithm 1 RL-CD algorithm

Let m_{cur} be the currently active partial model.
Let $\mathcal{M}$ be the set of all available models.
1: $m_{cur} \leftarrow newmodel()$
2: $\mathcal{M} \leftarrow \{m_{cur}\}$
3: $s \leftarrow s_0$, where s_0 is any starting state
4: **loop**
5: Let a be the action chosen, considering model m_{cur}
6: Observe next state s' and reward r
7: Update E_m, for all m, according to equation 6
8: $m_{cur} \leftarrow \arg\min_m (E_m)$
9: **if** $E_{m_{cur}} > \lambda$ **then**
10: $m_{cur} \leftarrow newmodel()$
11: $\mathcal{M} \leftarrow \mathcal{M} \cup \{m_{cur}\}$
12: **end if**
13: Update $\hat{T}_{m_{cur}}$ and $\hat{R}_{m_{cur}}$ (equations 2 and 3)
14: $\mathcal{N}_m(s,a) \leftarrow \min(\mathcal{N}_m(s,a)+1, M)$
15: $s \leftarrow s'$
16: **end loop**

The *newmodel*() routine is used to create a new partial model and initializes all estimates and variables to zero, except T_m, initialized with equally probable transitions. The values of parameters M, ρ, Ω and λ must be tunned according to the problem. Formal analytical investigation of these parameters is being developed but detailed discussions are not presented due to lack of space.

4. EMPIRICAL RESULTS

We now present some experiments which evaluate how well our system performs in non-stationary environments. The scenario used consists of a non-stationary toroidal grid of 15x15 cells. The agent is a cat whose goal is to catch a moving ball, which starts in a random column and row and can take one of following behaviors: **1)** moves to the left; **2)** moves to the right; **4)** moves down; or **3)** moves up.

The first experiment performed aims at testing the relative efficiency of classic model-free and model-based algorithms: Q-Learning (QL) and Prioritized Sweeping with Finite Memory (PS-M)[1]. The agent is trained for one of the ball behaviors at a time, until the algorithm converges to the optimal policy. After that, we change the environment dynamics by modifying the ball behavior and measure how fast each algorithm converges. It can be seen in figure 1 that, as the context changes, the time needed to recalculate the optimal policies, both in the model-free and the model-based approaches, is much superior to the convergence time of RL-CD. Even though PS-M performs better than Q-Learning, it still has to reestimate $\hat{T}$ after every context change. Our method, on the other hand, takes only a few steps until realizing that the context has changed. After that, it automatically selects the appropriate partial model for the new dynamics.

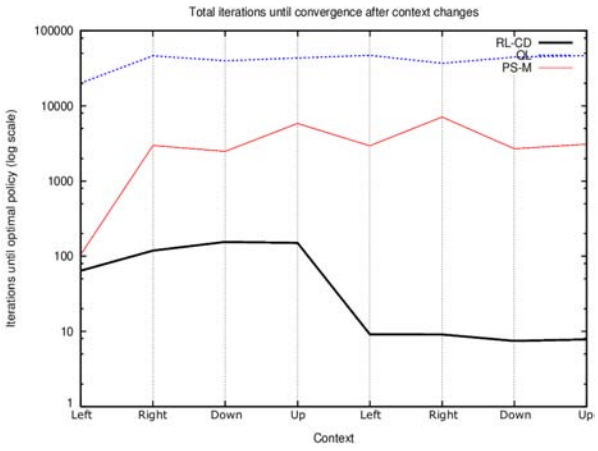


Figure 1: A comparison of convergence times for Q-Learning, PS-M and our approach.

It is also important to measure the average time the cat takes to catch the ball *while* the policy is being learned in a context. In order to implement this experiment, we runned the learning algorithms and changed the ball behavior 8 times. For each one of these, we performed 5 batches of 100 episodes, each episode corresponding to the cat being placed in a random place and running after the ball. We measured the average time of a batch as the average number of iterations needed to catch the ball during all episodes. The results are shown in figure 2.

In figure 2, vertical stripes are colored with different levels of gray, each indicating a different context. Each time the ball behavior changes, the average steps per episode grows (peaks in the graph), indicating that the algorithms take some time to relearn how to behave. During the first 4 contexts, our method is actually very similar to PS-M. However, as soon as the contexts begin to repeat, RL-CD is capable of acting optimally all the time, while the other algorithms present periods of relearning and suboptimal acting.

[1]We verified that standard PS requires exponential time to converge in non-stationary environments, and thus we designed and used **PS-M**, a modified PS with truncated memory instead of maximum likelihood estimation.

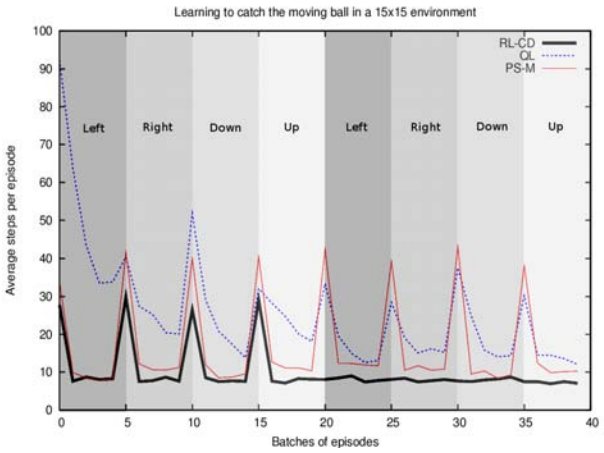


Figure 2: A comparison of performance for Q-Learning, PS-M and our approach.

From the empirical results discussed in this session we can imply that: **1)** our approach converges to the optimal policy much faster than classic model-free and model-based methods in non-stationary environments; and **2)** in our approach the readapting time is very small, which implies that the period of time with suboptimal acting is short.

5. CONCLUSIONS AND FUTURE WORK

In this paper we have introduced and formalized a simple method, called RL-CD, for solving reinforcement learning problems in non-stationary environments. We presented empirical results which show that both the classic model-free and model-based approaches have worse convergence times.

Formal analytical analysis of the RL-CD parameters is being developed, as well as a more detailed study regarding the trade-off between memory requirements and model quality in highly non-stationary environments. We are also studying the possibility of using context detection as a method for dimensionality reduction, since partial models might summarize the impact of a subset of hidden sensors which cause non-stationarity due to partial observations.

6. REFERENCES

[1] S. P. M. Choi, D.-Y. Yeung, and N. L. Zhang. Hidden-mode markov decision processes for nonstationary sequential decision making. In *Sequence Learning - Paradigms, Algorithms, and Applications*, pages 264–287, London, UK, 2001. Springer-Verlag.

[2] K. Doya, K. Samejima, K. ichi Katagiri, and M. Kawato. Multiple model-based reinforcement learning. *Neural Computation*, 14(6):1347–1369, 2002.

[3] L. P. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

Software agents that learn through observation (Short paper)

Jaçanã Machado
ADETTI / ISCTE
We, the Body and the Mind Research Lab
+351 21 782 64 84
Jacana.Machado@we-b.mind.org

Luís Botelho
ADETTI / ISCTE
We, the Body and the Mind Research Lab
+351 21 782 64 84
Luis.Botelho@we-b-mind.org

ABSTRACT
In this paper, we present an architecture for software agents that enables them to learn vocabulary through the observation of each other bodies and actions. Besides sensors, effectors, and action control, the architecture provides the equivalent of a body with visual appearance. The agent soft visual appearance is designed to be seen by other software agents, not by people. The paper describes the agent software body with visual appearance, the learning mechanism, the demonstration scenario and presents some results showing that the agent software body allows agents to learn vocabulary through observation and to ground the meaning of the symbols they learn. The paper emphasizes the important role of social interaction in learning processes.

Categories and Subject Descriptors
I.2.6 [**Artificial Intelligence**]: Learning – *language acquisition.*

General Terms
Algorithms, Experimentation, Theory

Keywords
Learning vocabulary through observation, Embodiment, Software body, Software visual appearance, software agents

1. INTRODUCTION
One of the most remarkable features of living bodies is their visual appearance, which enables them to communicate and to learn through mutual observation.

Having a body enables agents to benefit from the advantages embodiment accomplishes for living beings [1]. This work proposes a component based agent architecture with software visual appearance that enables agents to learn simple vocabulary (names of body parts and actions) through observation. This is a short version of the paper *"Software agents that Learn through observation"*[1]

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005…$5.00.

[1] iscte.pt/~luis/papers/AAMAS2006Learning.pdf

Our work is centred on the problem of learning simple vocabulary (nouns and verbs) through observation. We have implemented some learning experiments that provided some evidence supporting the validity of our proposal. In our learning experiments there are only two agents: the Observed Agent, and the Observer agent. The Observer Agent learns the words spoken by the Observed Agent and grounds their meanings in its own body and actions using a vocabulary map. The Observed Agent shows parts of its body (e.g., hand) and executes simple actions (e.g., it shakes its hand) while it says words that represent nouns (e.g., "hand") or words that represent verbs (e.g., "shake") naming the shown body parts or executed actions. Nouns are related to parts of the body, verbs are related to actions of the body [3]. This process requires that agents have body that can be observed by other software agents.

The reminder of the paper is organized as follows. Section 2 describes related work. Section 3 describes the demonstration scenario, and the results that have been achieved in our learning experiments. Section 4 describes our proposal. Section 4.1 describes the component-based architecture which sensors, effectors, action control, Learning Mechanism, social skills, and soft visual appearance. Finally, in section 5, we present conclusions and describe future directions for further research.

2. RELATED WORK
Artificial Intelligence research was founded under the influence of classical cognitive theory and the issue of symbolic representation: the mind is a symbol system and cognition is symbol manipulation. Generation of complex behaviour through symbol manipulation was successfully demonstrated by classical artificial intelligence [4].

Modern cognitive science has reviewed old concepts of pure mental processes and started considering the importance of the body in cognitive processes. For modern AI researchers and philosophers, there cannot be intelligence without a body. Modern artificial intelligence researchers, like Tom Ziemke[1], Brooks[2], Kushmeric[5], Varela[7], and Mataric[6] have been studying the role of the body on the generation of intelligent behaviour in natural and artificial systems. Principles of software embodiment architectures have been proposed by Kushmeric [5], and Botelho and Figueiredo [1]. Kushmeric has developed a computational framework for analyzing embodiment and identified several ways embodiment may reduce the complexity of the methods used in the agent's tasks. Botelho and Figueiredo have proposed concrete architectural principles for embodied software agents [1], a class of software agent capable of exhibiting some of the beneficial roles embodiment accomplishes in living beings, such as facilitating learning processes and social interactions.

3. DEMONSTRATION SCENARIO

In our demonstration scenario, the observed and observer agents are anthropomorphic. Their bodies are composed of the following components: mouth (i.e., textual language effector), ear (i.e., textual language sensor), eye (static and dynamic visual sensor), hand, action control, learning mechanism, action capturing mechanism, vocabulary map, and graphic user interface. Even though they could have different bodies, we used two agents with identical bodies. The Observed Agent teaches two concepts: *hand* and *shake hand*. It shows its hand and says the word "hand", and it shakes its hand and says the phrase "shake hand". In the described experiments, "saying" and "hearing" are accomplished through the exchange of textual messages.

The proposed approach can be used for agents of different nationalities. Therefore it can be used in cases in which a French speaking agent needs to learn English or some other language. When learning the word "hand" and its meaning, the Observer Agent identifies two body parts as possible candidates to match the Observed Agent part that is named by the word "hand": *pied* (French word for foot) and *main* (French word for hand). Given the ambiguity, the Observer Agent asks the Observed Agent for help. The Observed Agent knows that *hand* is part of *arm* and *foot* is part of the *leg* therefore it can help the Observer Agent to choose the correct part. After disambiguating the meaning of the learnt word, the Observer Agent imitates the observed action: it shows its hand, generates the textual expression "*I have a hand*" and sends it to the Observed Agent. The Observed Agent recognizes the success of the learning experiment and sends a positive textual message to the Observer Agent. Finally, since the Observer Agent knows the learning has been successful, it creates an entry in its vocabulary map to the noun "hand" and its grounding: the source of information is the Observed Agent; the observer agent class for "hand" is called *hand*; and its own class for hand is called *main*.

In the second experiment, the Observed Agent shakes its hand and says the phrase "shake hand". The Observer Agent identifies its body parts *pied* and *main* as possible candidates to perform the action named by the phrase "shake hand". The Observed Agent knows that *hand* is part of *arm* and *foot* is part of *leg* therefore it can help the Observer Agent to choose the right part. Then, the Observer Agent performs the actions that can be done with the class *main*. This process ends when it determines that the action *agiter* (French word for "to shake") performed by its *main* matches the observed action (through the visual appearance of the Observed Agent). Therefore the Observer Agent shakes its hand, and generates the expression "To shake the hand" and sends it to the Observed Agent. This one recognizes that the learning process was successful hence it sends a positive textual message to the Observer Agent. Finally, the Observer Agent creates a new entry for the verb "to shake" and its grounding in its vocabulary map: the source of information is the Observed Agent; the observed agent class of the body part involved in the observed action is *hand*, its own class of the corresponding body part is *main*; the observed agent method that implements the action is called *shake*; and its own method that implements the observed action is called *agiter.*

These two learning experiments show that the proposed architecture can support simple forms of vocabulary learning through observation. The architecture supports both noun learning and verb learning. The experiments also emphasize the role of social interaction for disambiguating the result of the learning process. The experiments also show that the architecture enables the agent to ground the learnt symbols in its own body and actions. Although the used algorithms exhibit some limitations (e.g., they do not work for learning actions performed by several parts of the body), the general processes are independent of the agents specific make up. The same algorithms could have been applied to agents with FTP and SMTP sensors and effectors instead of hands and feet.

4. LEARNING SIMPLE VOCABULARY THROUGH OBSERVATION

The agent architecture supporting the described learning experiments includes the mechanisms required to create the software-equivalent to the agent visual appearance and the mechanisms required for learning through observation. The agent visual appearance can be seen (consulted) by its owner and by other agents.

4.1 Agent Architecture and Software Visual Appearance

The proposed agent architecture is a component-based architecture with several components providing the following functions: sensors, effectors, vocabulary map, control mechanism, learning mechanism, and software visual appearance (static composition and dynamic behavior).

Sensors support visual and textual language perceptions. Visual sensors are used to capture visual appearance while the textual language sensor is used to capture textual messages. The visual sensor captures dynamic and static information of the observed body. The learning mechanism is the algorithm used by the Observer Agent to learn through observation. The vocabulary map represents the words known to the agent together with their meanings. It grounds the meaning of each word through a relationship between known word and its type (verb / noun), the source of information from where the word was acquired (in our case, this is the identification of the Observed Agent or the special symbol *built-in*), the classes of the component named by the acquired words or the classes of the component that executed the action named by the acquired words (in the Observed Agent's body and in the Observer Agent's body), and the signatures of the methods of the agent program encoding the action named by the stored word (on the side of the observed agent and on the side of the observer agent). All these attributes are mandatory, except method signatures which are mandatory only when the type of word is verb.

The agent software visual appearance consists of static information regarding the composition of the agent body and dynamic information about the actions the agent executes. Software visual appearance (both static and dynamic information) is made publicly available to other agents by the agent software visual appearance component. The agent software visual appearance is a component that captures information regarding its static body composition, and information about its dynamic body actions. Static visual appearance includes information regarding the composition, properties and roles of the agent visible components. The information regarding the agent's components

and their roles is used by Observer Agents to create a static mental image of the parts integrating Observed Agents. This mental image is interpreted as the static visual appearance of the Observed Agent. Dynamic elements of the body are used by the Observer Agent to create a dynamic mental image of the actions of the Observed Agent. Dynamic visual image relates to what the agent is doing at each instant of time.

"Seeing another agent" means consulting its software visual appearance and performing the required inferences. The dynamic information maintained by the agent visual appearance is captured by a crosscutting mechanism called the Action Capturing Mechanism. The Action Capturing Mechanism captures all relevant elemental operations performed by the agent involving the visible parts of its body, including get and set operations (i.e., get and set the value of a visible attribute of any of the visible agent components), entry and exit operations (i.e., invoking and exiting a method that performs a visible action). Composite actions are made up of sequences of elemental actions therefore they can also be seen through the agent software visual appearance.

4.2 The Learning Mechanism

Learning is facilitated through textual interactions among the two agents. The Observed Agent initiates the interaction by showing a part or an action of its body, and saying a word naming the shown part or action. When the Observed Agent shows its body parts and actions, the relevant part/action is marked, in its soft visual appearance. Therefore the Observer Agent knows exactly what is being shown, and the Observed Agent does not need to have salience building mechanisms. In real life scenarios, learning vocabulary through observation would require that the Observed Agent would be capable of building salience on the parts and actions it shows because, when the Observer Agent looks at the Observed Agent body, it sees the whole body, not only the relevant parts and actions intended to be shown. We overcame this salience building problem through a simplification consisting of marking the salient parts of the visual appearance. The visual appearance does not contain the information about the type of word (noun or verb). There is no visual difference between showing a body part or perform an action: both are actions. The difference is given by the textual message said by the Observed Agent.

The Observer Agent triggers its learning mechanism when it hears and sees the Observed Agent showing its body parts or executed actions while saying its names. The Learning mechanism identifies the part of the body/action shown by the observed agent associating it to the heard word. It compares the observed part/action with its own body parts/action in order to select those that more closely mirror the observations. When the learning agent discovers the part or the action of its body matching the observations, it grounds the meaning of the learnt words in its own body and actions. When the Observer Agent learns a word, it generates sentences using the learnt word, using pre-coded sentence schemata and the vocabulary already mastered by the agent. The Observed Agent provides feedback regarding the accuracy of the learning results. When learning is perceived to be accurate, the Observed Agent generates a positive textual reaction. When learning is perceived to be inaccurate, the Observed Agent generates a negative textual reaction. This feedback allows the Observer Agent to know whether its learning was correct or incorrect.

5. CONCLUSION AND FUTURE WORK

The strongest contribution of this paper is to present a concrete software agent architecture exhibiting some of the features required to have a body with software visual appearance that can be seen by other software agents (not by people) through which it is possible to improve several cognitive tasks, in particular "learning simple vocabulary through observation". The proposed approach can be used to overcome some criticisms put forth against classical AI. Namely, through the proposed approach, it is possible for an agent to ground the meanings of acquired vocabulary. That is, in addition to being understood by third parties (agent designers and observers), learnt symbols acquire meaning to the agent itself. The proposed architecture and algorithms allow the agent to ground the meanings of the acquired vocabulary on the parts and actions of its body. This allows agents to create first person meanings for the symbols they process. The defined and implemented demonstration scenario shows that the proposed architecture and the used learning algorithms are general enough to support learning through observation in diverse application domains. The demonstration also shows that social interaction is an important condition for learning through observation. This is most apparent when the learning process generates ambiguity. The proposed learning process can also be used in different situations such as those in which an agent must learn a different communication language.

A direction for future work is to build the mechanisms necessary to increase the salience of the shown body parts and actions. We will explore the role of emotion as the basis for those processes.

6. REFERENCES

[1] Botelho; L. and Figueiredo, P. 2004. "What Your Body and Your Living Room Tell My Agent". Proceedings of the AAMAS 2004 Workshop "Balanced Perception and Action in Embodied Conversational Agents"

[2] Brooks, R. 1991. *Intelligence without Representation*, Artificial Intelligence 47(1991) 139-159 Elsevier

[3] Cangelosi, A., Riga, T., Giolito, G., and Morocco, D. Language Emergence and Grounding in Sensorimotor Agents and Robots. *First International on Emergence and Evolution of Linguistic Communication,* May 31 – Jun 1 2004, Kanazawa Japan.

[4] Harnad, S. 1990. The Symbol Grounding Problem. Physica D 42: 335-346.

[5] Kushmerick, N. 1997. "Software agents and their bodies". Journal of Minds & Machines, 7(2):227-47

[6] Mataric, M. Studying the Role of Embodiment in Cognition. In Cybernetics and Systems, Special issue on Epistemological Aspects of Embodied AI, 28(6):457-470.

[7] Varela, F., Thompson, E. & Rosch, E. (1991) *The Embodied Mind - Cognitive Science and Human Experience*.Cambridge, MA: MIT Press.

[8] Ziemke, T. 1999. "Rethinking grounding". In Riegler, Peschl & von Stein (eds.) Understanding Representation in the Cognitive Sciences (pp. 177-190). Plenum Press

Task and resource allocation

Resource Selection Games with Unknown Number of Players

Itai Ashlagi Dov Monderer Moshe Tennenholtz
Technion–Israel Institute of Technology
Faculty of Industrial Engineering and Management
Haifa 32000, Israel
ashlagii@tx.technion.ac.il
{dov,moshet}@ie.technion.ac.il

ABSTRACT

In the context of pre-Bayesian games we analyze resource selection games with unknown number of players. We prove the existence and uniqueness of a symmetric safety-level equilibrium in such games and show that in a game with strictly increasing linear cost functions every player benefits from the common ignorance about the number of players. In order to perform the analysis we define safety-level equilibrium for pre-Bayesian games, and prove that it exists in a compact-continuous-concave setup; in particular it exists in a finite setup.

Categories and Subject Descriptors

K.4 [**Social and Behavioral Sciences**]: Economics; I.2 [**Artificial Intelligence**]: Distributed Artificial Intelligence —*Multiagent systems*

General Terms

Economics, Theory

Keywords

Pre-Bayesian games, Safety-level equilibrium

1. INTRODUCTION

In a *resource selection system*, Γ, there is a set of m resources, $j \in \{1, \cdots, m\}$. Each resource j is associated with a cost function $w_j : \{1, 2, \cdots, \cdots\} \to \Re$, where $w_j(k)$ is the cost for every user of resource j if there are k users. Together with a set of n players a resource selection system defines a game in strategic form– *a resource selection game*, $\Gamma(n)$. The action set[1] of every player i in $\Gamma(n)$ is the set of resources M, and the cost of i depends, via the resource-cost functions, on the resource she chooses and on the number of other players who choose this resource. Thus, resource selection games are special types of congestion games [23, 18]. A resource selection game is also referred to as a simple congestion game.[2] In many situations the assumption that every player knows the number of players is not reasonable. The goal of the current paper is to analyze resource selection games with unknown number of players. One approach for analyzing such situations is the Bayesian approach, where it is assumed that the distribution of the random set of players is commonly known. In this approach one is interested in Bayesian equilibrium. [3] The goal of this paper is to analyze resource selection games in the non-Bayesian setup in which the players do not have probabilistic information about the number of the other players in the game. We will use what we call pre-Bayesian games. A pre-Bayesian game is defined to be a Bayesian game without the state probability.[4]

In a classical pre-Bayesian game there is a fixed set of players $I = \{1, 2, \cdots, n\}$, each of them is endowed with a set of actions, $x_i \in X_i$. There is a set of states $\omega \in \Omega$. The payoff of player i, $u_i(\omega, x_1, x_2, \cdots, x_n)$ depends on the realized state, on her choice of action, x_i, and on the choices of all other players. However, the realized state is not known to the players. Every player receives a state-correlated signal, $t_i = \tilde{t}_i(\omega)$ on which she conditions her action. A pre-Bayesian game becomes a Bayesian game when a commonly known probability measure on the set of states is added to the system. The above description is not general enough for our purpose because the set of players is fixed. We generalize the model by allowing the set of active players as well as the action sets to depend on the state.

In pre-Bayesian games one can deal with ex-post equilibrium, which generalize the notion of dominant strategy

[1]When dealing with games in strategic form the choice set of a player is referred to as an action set or as a strategy set. We use "actions" rather than "strategies" because we keep the notion of strategy to describe the choice set of a player in "bigger" games.

[2]Simple congestion games and their generalization to player-specific games were discussed in [22, 16, 8, 27, 11]. Simple congestion games were also discussed in the price of anarchy literature, e.g., [12, 2]

[3]See [15, 14, 6] for such analysis in the context of auctions, and [19, 20, 21] for such analysis in the context of elections.

[4]Surprisingly, until recently such games did not have a name in the literature. Pre-Bayesian games have been also called *games in informational form* and *games without probabilistic information* [9, 7], *games with incomplete information with strict type uncertainty* [10], and *distribution-free games with incomplete information* [1].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

equilibrium. Such equilibria rarely exist, and consequently only very special type of pre-Bayesian games have been discussed in Game Theory or in Computer Science. Hyafil and Boutilier (2004)[10] suggested minimax regret equilibrium as a solution to general pre-Bayesian games. In such equilibrium, every agent, after receiving her informational signal is using the minimax regret criterion in the decision making problem obtained by fixing the behavior of all other agents. They proved that such an equilibrium exists in every finite pre-Bayesian game. In this paper we define another type of equilibrium concept, safety level equilibrium. In such an equilibrium, every player, after receiving her informational signal use the maximin payoff (or minimax cost) criteria in the decision making problem obtained by fixing the behavior of all other agents. We use a more general definition of pre-Bayesian games and prove that a safety level equilibrium always exists. Independently, [1] also define safety level equilibrium. However, they restricted attention to a very particular information structure in which all players receive the same signal. All of the above mentioned existence results rely on Kakutani fixed point theorem, which is a standard tool in game theory for proving equilibria existence results.[5]

As we said, our main goal in this paper is to analyze resource selection games with unknown players, and we do it by analyzing safety-level equilibria in such games. However, for the sack of completeness we also define and prove existence of a third type of equilibrium, which is consistent with a large part of single agent setups of incomplete information in CS. We define competitive ratio equilibrium and prove that it exists.[6]

Once we have the right tool we proceed to analyze resource selection games with unknown number of players. We focus on a model with increasing resource cost functions. In order to derive results for the case in which the number of players is unknown we prove several results, some of them are interesting for themselves, about classical resource selection games with known number of players. In particular we prove that every resource selection game possesses a unique mixed-action symmetric equilibrium. When the number of players is k, the unique symmetric equilibrium mixed-action of every player is denoted by p^k. That is, $p^k = (p_1^k, p_2^k, \cdots, p_m^k)$, where p_j^k is the probability that a player chooses resource j.

In our pre-Bayesian model all active players know a common bound, n, on the number of active players, but the players do not know the true number of players, say k, $k \leq n$. Hence, the only signal a player receives is an "activity" signal. A state in this pre-Bayesian game is the set of active players. We prove that a resource selection game with unknown number of players has a unique symmetric safety level equilibrium. In this equilibrium, every active player is using the unique symmetric-equilibrium mixed-action, p^n in the game in which the number of players is commonly known and equals n.

Hence, when the number of players is k, every player uses p^k when the number of players is commonly known, and every player is using p^n when the number of players is unknown. Surprisingly, the lack of knowledge makes each of the players better off in the linear system, in which the resource cost functions are linear! That is, we show that in the linear model, when there are k players, and each of them is using p^n, the cost of each player is at most the cost he obtains in the unique symmetric equilibrium, p^k. Under very modest assumptions every player is strictly better off.

The above results are applicable to a mechanism design setup in which the organizer knows the number of active players, and the players do not know this number. If the goal of the organizer is to maximize revenue then he is better off revealing his private information.[7] If his goal is to maximize social surplus then he should not reveal the information. In order to estimate the gain of the players resulting from their ignorance we investigate in the last section the function $c^k(p^n)$ for $n \geq k$, where $c^k(p^n)$ is the cost for a player in a k-player setup when all players play the mixed-action associated with the symmetric equilibrium of the corresponding n-player setup.

2. BACKGROUND

We remark that games can be analyzed with either payoff functions or cost functions. Most of the general theory has been developed with payoff functions. However, congestion games have been mainly analyzed using cost functions. Translating results proved in one of the setups to the other setup is obvious in most cases.[8] We follow the tradition of previous literature in the sense that the general theory is discussed with payoff functions, while resource selection games are discussed with cost functions.

A game in strategic form is a tuple, $\Gamma = (I, (X_i)_{i\in I}, (u_i)_{i\in I})$. I is a nonempty set of players, X_i is a nonempty set of actions for player i, and $u_i : X \to \Re$ is the payoff function of i, where $X = \times_{i\in I} X_i$ is the set of action profiles. Hence $u_i(x)$ is the payoff of player i when the profile of actions $x \in X$ is played. Γ is a *finite* game if I and X are finite sets.

Let $x \in X$ denote a profile of actions. For each $i \in I$ we let $x_{-i} = (x_j)_{j\in I\setminus\{i\}}$ denote the actions played by everyone but i. Thus $x = (x_i, x_{-i})$. An action profile $x \in X$ is in *equilibrium* if $u_i(x_i, x_{-i}) \geq u_i(y_i, x_{-i})$ for every player $i \in I$ and for every $y_i \in X_i$.

A *permutation* of the set of players is a one-to-one function from I onto I. For every permutation π and for every action profile $x \in X$ we denote by πx the permutation of x by π. That is, $(\pi x)_i = x_{\pi(i)}$ for every player i. Γ is a *symmetric game* if $X_i = X_s$ for all $i, s \in I$, and

$$u_i(\pi x) = u_{\pi(i)}(x)$$

for every player i, for every action profile x, and for every permutation π. A *symmetric action profile* is an action profile x such that $x_i = x_s$ for every $i, s \in I$. x is *a symmetric equilibrium* if it is both an equilibrium profile and a symmetric action profile .

For any finite set C, $\Delta(C)$ denotes the set of probability distributions over C. Let $\Gamma = (I, (X_i)_{i\in I}, (u_i)_{i\in I})$ be a fi-

[5]We note that an equilibrium concept for specific types of pre-Bayesian games were already defined in [26, 25, 24] in the context of work on artificial social systems, and in [13], where another equilibrium concept is defined for particular pre-Bayesian auctions. Both groups of authors used versions of safety-level equilibrium.

[6]The competitive ratio approach is relatively common in computer science in the context of single agent decision problems (see e.g., [3]. A recent paper of [5] applies the notion of competitive equilibrium to a particular problem of online algorithms.

[7]This is indirectly related to the Linkage Principle, in auction theory [17].

[8]An exception is the price of anarchy theory.

nite game in strategic form. Every $p_i \in \Delta(X_i)$ is called a *mixed action* for i. $p_i(x_i)$ is the probability that player i plays action x_i. Every vector $p \in \Delta = \times_{i\in I}\Delta(X_i)$ defines a probability distribution over X; The probability, $p(x)$ of $x \in X$ is $\prod_{i\in I} p_i(x_i)$.

Let u_i^m be the expected payoff function defined on Δ by u_i. That is, $u_i^m(p) = E_p(u_i)$.

The game $\Gamma^m = (I, (\Delta(X_i))_{i\in I}, (u_i^m)_{i\in I})$ is called the *mixed extension* of Γ. A mixed action for player i in the game Γ is an action for player i in the game Γ^m.

A *mixed action equilibrium* in a finite game Γ is defined to be an equilibrium in Γ^m. It is well-known (see e.g, [4]) that every finite symmetric game in strategic form possesses a symmetric mixed action equilibrium.

3. RESOURCE SELECTION GAMES WITH KNOWN NUMBER OF PLAYERS

In a *resource selection system*, $\Gamma = (M, (w_j)_{j=1}^m)$ there is a set of resources, $M = \{1, \cdots, m\}$, $m \geq 1$. Every resource j is associated with a cost function $w_j : \{1, 2, \cdots\} \to \Re$. $w_j(k)$ is the cost for every user of resource j if there are k users. Together with the set of players $I_n = \{1, \cdots, n\}$ a resource selection system defines a game in strategic form– *a resource selection game* $\Gamma(I)$. In this paper only the number of players will be relevant (and not the "names" of the players) and therefore we will denote a resource selection game by $\Gamma(n)$. The action set of every player i in $\Gamma(n)$ is the set of resources M, and the cost of i depends on the resource she chooses and on the number of other players that choose this resource via the resource-cost functions. That is, $X_i^n = M$ for every $1 \leq i \leq n$, and $c_i^n(x) = w_{x_i}(\sigma_{x_i}(x))$, where for every resource j and for every action profile $x \in \times_{i=1}^n X_i^n = M^n$, $\sigma_j(x)$ is the number of all players s for which $x_s = j$. Obviously every resource selection game is a finite symmetric game.

Let $p \in \Delta(M)$ be a mixed action of an arbitrary player. That is, $p = (p_1, \cdots, p_m)$, where p_j is the probability that a player who uses the mixed action p will select resource j. We denote the support of p by $supp(p)$. That is $supp(p) = \{j \in M | p_j > 0\}$. Denote by $c^n(p, j)$ the expected cost of a player that chooses resource j when each of the other $n-1$ players in $\Gamma(n)$ is using p. Let $c^n(p)$ be the expected cost of every player when each of the n players in $\Gamma(n)$ is choosing p.

For every $n \geq 1$, and for every $0 \leq \alpha \leq 1$. Let $Y_\alpha^n \sim Bin(n, \alpha)$ be a binomial random variable. That is, $f_\alpha^n(k) = P(Y_\alpha^n = k) = \binom{n}{k}\alpha^k(1-\alpha)^{n-k}$ for every $0 \leq k \leq n$. Let $F_\alpha^n(k) = P(Y_\alpha^n \leq k)$ be the distribution function of Y_α^n. Obviously

$$c^n(p, j) = E(w_j(1 + Y_{p_j}^{n-1})), \quad (1)$$

where E stands for the expectation operator. That is,

$$c^n(p, j) = \sum_{s=0}^{n-1} w_j(s+1) f_{p_j}^{n-1}(s). \quad (2)$$

Let $(q, \cdots, q) \in \Delta(M)^n$ be a symmetric mixed-action equilibrium profile in $\Gamma(n)$. We will refer to q as a *symmetric-equilibrium action.*

THEOREM 1. *Every resource selection game with at least two players ($n \geq 2$) with increasing* [9] *resource cost functions possesses a unique symmetric mixed-action equilibrium.*

[9]That is, $w_j(k) < w_j(k+1)$ for all j and k.

In order to prove Theorem 1 we need some preparations. The proof of the following lemma is given in the full version.

LEMMA 1. *Let $n \geq 1$. $F_\alpha^n(k)$ is a strictly decreasing function of α for every $0 \leq k \leq n-1$.*

LEMMA 2. *Let $\Gamma(n)$, $n \geq 2$ be a resource selection game. Let $q, p \in \Delta(M)$ be mixed actions, and let $j \in M$ be a resource such that w_j is increasing in $\{1, 2, \cdots, n\}$. If $p_j > q_j$ then $c^n(p, j) > c^n(q, j)$.*

PROOF. We have to show that $c^n(p, j)$ is increasing in p_j. By manipulating (2)

$$c^n(p, j) =$$

$$\sum_{k=1}^{n-1}\left((w_j(k) - w_j(k+1))\sum_{l=0}^{k-1} f_{p_j}^{n-1}(l)\right) +$$

$$w_j(n)\sum_{l=0}^{n-1} f_{p_j}^{n-1}(l) =$$

$$w_j(n) - [\sum_{k=1}^{n-1}(w_j(k+1) - w_j(k))F_{p_j}^{n-1}(k-1)],$$

where the last equality follows from the fact that $\sum_{k=0}^{n-1} f_{p_j}^{n-1}(k) = 1$.

By lemma 1, $F_{p_j}^{n-1}(k)$ is strictly decreasing in p_j for every $k = 0, ..., n-2$.In addition, w_j is strictly increasing, and therefore $c^n(p, j)$ is strictly increasing in p_j. □

Proof of Theorem 1

Suppose in negation that there is more than one mixed-action symmetric equilibrium in $\Gamma(n)$. Let q and p be two symmetric equilibrium actions with $p \neq q$. Since $p \neq q$ there exists $j \in M$ with $q_j \neq p_j$. W.l.o.g $q_j > p_j$. Therefore there exist a resource $r \in M$ such that $r \neq j$ and $q_r < p_r$. We get a contradiction from the following sequence of inequalities: $c^n(q, j) > c^n(p, j) \geq c^n(p, r) > c^n(q, r) \geq c^n(q, j)$, where the strict inequalities hold hold by Lemma 2 and the other inequalities hold because q and p are equilibrium actions. □

For every $n \geq 1$ we will denote the unique symmetric equilibrium in $\Gamma(n)$ by p^n, and we denote by $c^n = c^n(p^n)$ the equilibrium cost of a player in $\Gamma(n)$.

We say that a resource cost function w_j is *convex* if it can be extended to a convex function on $[1, \infty)$.

The following lemma will be useful later.

LEMMA 3. *Let $\Gamma = (m, (w_j)_{j=1}^m)$ be a resource selection system, with increasing and convex cost functions. There exists an integer $N \geq 2$, $N = N(\Gamma)$ such that for every $n \geq N$, the unique symmetric-equilibrium action in the game $\Gamma(n)$, $p^n \in \Delta(M)$ has a full support. That is, $p_r^n > 0$ for every $1 \leq r \leq m$.*

PROOF. Recall that p^n is the unique symmetric-equilibrium action in $\Gamma(n)$, and that $c^n = c^n(p^n)$ is the symmetric-equilibrium cost of every player. As p^n is in equilibrium, $c^n(p^n, j) = c^n$ for every $j \in supp(p^n)$. For every resource j we denote by w_j the convex extension of w_j to $[0, \infty)$. As w_j is convex,

$$c^n(p^n, j) = E(w_j(1 + Y_{p_j^n}^{n-1})) \geq w_j(1 + E(Y_{p_j^n}^{n-1})) =$$

$$w_j(1 + p_j^n(n-1)),$$

where the first equality follows from (1), the inequality follows from the convexity of w_j, and the last equality follows from the well-known fact that

$$E(Y_\alpha^n) = \alpha n. \quad (3)$$

Obviously, there exists $j \in supp(p)$ for which $p_j^n \geq \frac{1}{m}$. For this resource j

$$c^n = c^n(p^n, j) \geq w_j(1 + \frac{1}{m}(n-1)) \geq$$

$$\min_{r=1}^{m} w_r(1 + \frac{1}{m}(n-1)).$$

Since w_j is increasing and convex, $\lim_{n\to\infty} w_j(n) = \infty$ for every resource j. Therefore $\lim_{n\to\infty} c^n = \infty$. Hence, there exists N such that for every $n \geq N$ $c^n > \max_{j=1}^m w_j(1)$. We claim that for every $n \geq N$, $p_r^n > 0$ for every $1 \leq r \leq m$. Indeed, if $p_r^n = 0$ for some r then because $c^n > w_r(1)$, a player will decreases her cost by deviating from p^n to r (assuming every other player is using p^n). This contradicts p^n being a symmetric-equilibrium action. □

4. EQUILIBRIUM IN PRE-BAYESIAN GAMES

In this section we define a general model of pre-Bayesian games, and present and prove existence results for safety-level, and competitive-ratio equilibria in such games. We also give the definition of minimax-regret equilibrium (existence proved by Hyafil and Boutilier [10]).

A *pre-Bayesian game* is a tuple $H = (\Omega, I, (Z_i)_{i\in I}, J, (X_i)_{i\in I}, (u_i(\omega, *)_{\omega\in\Omega, i\in I}, (T_i)_{i\in I})$, where,

- Ω is a set of *states*.
- I is a finite set whose elements are called *potential players*.
- $J : \Omega \to 2^I \setminus \{\emptyset\}$ is the function determining the set of active players. That is, i is *active* at ω if $i \in J(w)$.

  Let Ω_i be the subset of states at which i is active. That is, $\Omega_i = \{\omega \in \Omega | i \in J(w)\}$.
- Z_i is the set of potential actions of i.

  Only a subset of Z_i will be available to i at a given state.
- For every potential player i, $X_i : \Omega_i \to 2^{Z_i} \setminus \{\emptyset\}$ is the variable determining the set of actions that are available for i at ω.

  Let $X(\omega) = \times_{i\in J(\omega)} X_i(w)$ be the set of action profiles that can be generated at ω.
- For every potential player i, and for every $\omega \in \Omega_i$, $u_i(\omega, *) : X(\omega) \to \Re$ is the *payoff function* of i at the state ω.
- T_i is the set of possible signals that i may receive.

  We will sometimes follow the traditional approach in economics and refer to $t_i \in T_i$ as a *type* of i.
- $\tilde{t}_i : \Omega_i \to T_i$ is the *signaling function of* i.

  That is, whenever i is active at ω she receives the signal $\tilde{t}_i(\omega)$.

  Without loss of generality we assume that every type is possible. That is, $\tilde{t}_i(\Omega_i) = T_i$.

  Let $\Omega_i(t_i)$ be the set of all states ω that generate the type t_i. That is, $\Omega_i(t_i) = \{\omega \in \Omega_i | \tilde{t}_i(\omega) = t_i\}$.
- $\tilde{t}_i(\omega) = \tilde{t}_i(\omega')$ implies $X_i(\omega) = X_i(\omega')$.

  That is, players know their set of actions; In other words, X_i is a constant function over $\Omega_i(t_i)$ for every $t_i \in T_i$.

  This assumption allows us to define the set of available actions for i when his type is t_i. Indeed we define $X_i(t_i) = X_i(\omega)$ for any arbitrary $\omega \in \Omega_i(t_i)$.

The pre-Bayesian game proceeds as follows. Nature chooses $\omega \in \Omega$, and the game in strategic form $\Gamma(\omega) = (J(w), (X_i(\omega))_{i\in J(\omega)}, (u_i(\omega, *))_{i\in J(\omega)})$ is played. Every active player in the game, $i \in J(\omega)$ receives the signal $\tilde{t}_i(\omega)$, which reveals part of the information about the game. The minimal information is the set of available actions at ω. Thus, every $i \in J(\omega)$ chooses $x_i \in X_i(\omega)$ and his payoff is $u_i(\omega, x)$, where $x = (x_i)_{i\in J(\omega)}$.

A *strategy* for a potential player i in the pre-Bayesian game H is a function $s_i : T_i \to Z_i$ that satisfies: $s_i(t_i) \in X_i(t_i)$ for every $t_i \in T_i$. Hence, i follows the strategy s_i if whenever he is active and receives the signal t_i he chooses the action $s_i(t_i)$. The set of strategies of i is denoted by $\Sigma_i = \Sigma_i(H)$, and the set of strategy profiles is denoted by $\Sigma = \Sigma(H)$, that is $\Sigma = \times_{i\in I}\Sigma_i$.

Let H be a pre-Bayesian game. We say that H is *finite* if the set of states, and the sets of potential actions are finite.

We say that H is a *compact-continuous pre-Bayesian game* if the set of states are finite, Z_i is a compact subset of some Euclidean space[10], $X_i(t_i)$ is a compact subset of Z_i for every type t_i, and $u_i(\omega, *)$ is continuous on $X(\omega)$ for every $\omega \in \Omega_i$.

A compact continuous pre-Bayesian game is *(quasi) concave* if Z_i is a convex set, $X_i(t_i)$ is a convex subset of Z_i for every type t_i, and $u_i(\omega, *, x_{-i})$ is (quasi) concave on $X_i(\omega)$ for every $\omega \in \Omega_i$ and every $x_{-i} \in X_{-i}(\omega)$.

Let $H = (\Omega, I, (Z_i)_{i\in I}, J, (X_i)_{i\in I}, (u_i(\omega, *)_{\omega\in\Omega, i\in I}, (T_i)_{i\in I})$ be a finite pre-Bayesian game. We are about to define the mixed extension, H^m of H by allowing every active player at $\omega \in \Omega$ to choose a mixed action $\mu_i \in \Delta(X_i(\omega))$. However we need the following definition.

For every $\omega \in \Omega_i$ let $\Delta_i(\omega)$ be the set of all probability distributions $\mu_i \in \Delta(Z_i)$ that vanishes outside $X_i(\omega)$. Obviously, $\Delta_i(\omega)$ can be identified with $\Delta(X_i(\omega))$.

The pre-Bayesian game $H^m = (\Omega, I, (\Delta(Z_i))_{i\in I}, J, (\Delta_i)_{i\in I}, (u_i^m(\omega, *)_{\omega\in\Omega, i\in I}, (T_i)_{i\in I})$ is called the *mixed extension* of H, where $u_i^m(\omega, *)$ is the expected-payoff function defined by u_i. Obviously, H^m is a concave pre-Bayesian game. Every strategy of i in H^m is called a *mixed-strategy for i in H*.

4.1 Safety-level Equilibrium

Let H be a compact-continuous pre-Bayesian game. For every profile, $s \in \Sigma$ of strategies in H, and for every state ω we denote by $s[\omega]$ the profile of actions chosen in $X(w)$ when

[10]Or of some linear topological space.

every player i uses s_i. That is, $s[\omega] = (s_j(\tilde{t}_j(\omega))_{j\in J(\omega)}$. As usual, $s_{-i}[\omega]$ is obtained from $s[\omega]$ by removing the action chosen by i.

Let $i \in I$, $t_i \in T_i$, $x_i \in X_i(t_i)$, and let $s_{-i} = (s_j)_{j\in I\setminus i}$ be a profile of strategies of all players but i. The worst case cost of i is

$$W_i(t_i, s_{-i}, x_i) = \min_{\omega\in\Omega_i(t_i)} u_i(\omega, x_i, s_{-i}[\omega]).$$

Obviously $W_i(t_i, s_{-i}, *)$ is continuous on $X_i(t_i)$. We say that $x_i^* \in X_i(t_i)$ is *optimal* for type t_i given s_{-i} if the maximal value of $W_i(t_i, s_{-i}, x_i)$ over $x_i \in X_i(t_i)$ is attained at x_i^*. A strategy of player i, s_i, is a *safety-level best-response to* s_{-i} if for every type t_i, $s_i(t_i)$ is optimal for t_i given s_{-i}. A strategy profile $s = (s_i)_{i\in I}$ is called a *safety-level equilibrium* if for every i, s_i is a safety-level best-response to s_{-i}.

Hence, s is a safety-level equilibrium if and only if for every ω, and for every player i, which is active at ω, $s_i(\tilde{t}_i(\omega))$ is optimal for $\tilde{t}_i(\omega)$ given s_{-i}.

A safety-level equilibrium in a pre-Bayesian game with exactly one state is simply a Nash equilibrium in this game. We next show that safety-level equilibria exist in every quasi concave game.

THEOREM 2. *Every quasi concave pre-Bayesian game possesses a safety-level equilibrium.*

PROOF. Note that for every i Σ_i is a closed convex and compact subset of $Z_i^{T_i}$. Therefore Σ is a closed convex and compact set because it is a cartesian products of such sets. For every $s \in \Sigma$ and for every $i \in I$ let $B_i(s) \subseteq \Sigma_i$ be the set of all $d_i \in \Sigma_i$, which are best response to s_{-i}. Let $B(s) = \times_{i\in I} B_i(s) \subseteq \Sigma$. It is standard to check that the correspondence $s \to B(s)$ satisfies the conditions of Kakutani's fixed point theorem. That is, it is upper hemicontinuous, and $B(s)$ is a nonempty compact convex subset of Σ for every $s \in \Sigma$. Therefore there exists a fixed point s^*, that is, $s_i^* \in B_i(s^*)$ for every $i \in I$. Obviously such a fixed point is a safety-level equilibrium. □

Hence, if H is a finite game, H^m possesses a safety-level equilibrium. Every such an equilibrium is called a *mixed-strategy safety-level equilibrium* in H.

4.2 Minimax-regret Equilibrium

Minimax-regret equilibrium was defined by Hyafil and Boutilier in [10]. We use their definition for our general model of pre-Bayesian games.

Let H be a compact-continuous pre-Bayesian game. Let $i \in I$, $t_i \in T_i$ and let $s_{-i} = (s_j)_{j\in I\setminus i}$ be a profile of strategies of all players but i. Let $\omega \in \Omega_i(t_i)$. The *regret* of $x_i \in X_i(t_i)$ at w is defined as:

$$R(x_i, w, t_i, s_{-i}) =$$

$$\max_{z_i\in X_i(t_i)} [u_i(w, z_i, s_{-i}[\omega]) - u_i(w, x_i, s_{-i}[\omega]]].$$

The *maximal regret* of $x_i \in X_i(t_i)$ over all $\omega \in \Omega_i(t_i)$ is denoted by $MR(x_i, t_i, s_{-i})$. That is:

$$MR(x_i, t_i, s_{-i}) = \max_{\omega\in\Omega_i(t_i)} R(x_i, w, t_i, s_{-i}).$$

We say that $y_i \in X_i(t_i)$ is a minimax regret strategy at t_i given s_{-i} if the minimal value of $MR(x_i, t_i, s_{-i})$ over $x_i \in X_i(t_i)$ is attained at y_i. A strategy s_i is a minimax regret best response to s_{-i} if for every type t_i, $s_i(t_i)$ is a minimax regret strategy at t_i given s_{-i}. s is a *minimax regret equilibrium* if for every player i s_i is a minimax regret best response to s_{-i}. In [10] the authors show:

THEOREM 3 (HYAFIL AND BOUTILIER). *Every quasi concave pre-Bayesian game possesses a minimax-regret equilibrium.*

4.3 Competitive Ratio Equilibrium

Competitive ratio equilibrium resembles the minimax-regret equilibrium. They differ only in the definition of regret.

Let H be a compact-continuous pre-Bayesian game, where all payoff functions are positive. Let $s_{-i} = (s_j)_{j\in I\setminus i}$ be a profile of strategies of all players but i, let t_i be an active type of i, and let $\omega \in \Omega(t_i)$. Replace the definition of $R(x_i, w, t_i, s_{-i})$ in the previous section with the following:

$$\hat{R}(x_i, w, t_i, b_{-i}) = \max_{z_i\in X_i(t_i)} \frac{u_i(w, z_i, s_{-i}[\omega])}{u_i(w, x_i, s_{-i}[\omega])}.$$

Note that $\hat{R}$ is well defined since all payoff functions are positive. A strategy profile s is a competitive-ratio equilibrium if for every player i s_i is a minimax-regret best response to s_{-i} with respect to the regret function $\hat{R}$.

THEOREM 4. *Every quasi concave game in pre-Bayesian game with positive payoffs functions possesses a competitive ratio equilibrium.*

The proof follows from Theorem 3 by applying the logarithmic function to the payoff functions.

5. RESOURCE SELECTION GAMES WITH UNKNOWN NUMBER OF PLAYERS

Consider a fixed resource selection system, Γ with the set of resources $M = \{1, \cdots, m\}$, $m \geq 1$, and resource cost functions $(w_j)_{j=1}^m$.

We proceed to describe our model of resource selection games with unknown number of players. Let $I = \{1, 2, \cdots, n\}$, $n \geq 1$. be the set of potential players. The set of states, Ω is the set $2^I \setminus \{\emptyset\}$ of all nonempty subsets of I. The set of active players at state K is K itself. That is $J(K) = K$ for every $K \in \Omega$. The set of actions of player i is the set of resources. That is, $Z_i = M$ for every player i. The set of types of $i \in I$ is a singleton, $T_i = \{active\}$. That is, an active player knows that he is active. The non-active players do not receive any signal. For every player $i \in I$ and every $\omega \in \Omega_i$ $X_i(\omega) = M$. For every $i \in I$ and for every $K \in \Omega$ such that $i \in K$ the cost function of i $c_i(K, *) : Z \to \Re$ is defined by $c_i(K, z) = w_{z_i}(\sigma_{z_i}^K(z))$, where $\sigma_{z_i}^K(z)$ is the number of all players $l \in K$ for which $z_l = z_i$. The above pre-Bayesian game is finite. We denote its mixed extension by $H_\Gamma(n)$, and we referred to $H_\Gamma(n)$ as a *resource selection game with unknown number of players*. A strategy of player i in $H_\Gamma(n)$ can be described by a mixed action $q[i] \in \Delta(M)$. That is, when receiving the signal $\{active\}$, i uses $q[i]$. Since we deal with costs and not with payoffs we use minimax rather than maximin in the definition of safety-level equilibrium. let $\mu = (q[1], \cdots, q[n])$ be a strategy profile in $H_\Gamma(n)$, and let i be a player. Let $\mu[-i] = (q[l])_{l\in I\setminus\{i\}}$ be the profile of strategies of the other players. Observe that Ω_i is the set of all nonempty subsets of I that contain i. If all cost functions are non-decreasing, and if i believes that all other players

are using the profile $q[-i]$, then it is obvious that the worst case scenario for i is obtained in the state I. Thus we have:

LEMMA 4. *Let Γ be a resource selection system in which the resource cost functions are non-decreasing. Let $\mu \in \Delta(M)^n$. μ is a safety-level equilibrium in $H_\Gamma(n)$ if and only if μ is a mixed-action equilibrium in $\Gamma(n)$.*

PROOF. Assume μ is a mixed-action equilibrium in $\Gamma(n)$. Let i be an active player. By the comment we made before the statement of the lemma,

$$\min_{p[i]\in\Delta(M)} \max_{K\in\Omega_i} c_i(K, p[i], \mu[-i]) =$$

$$\min_{p[i]\in\Delta(M)} c_i(I, p[i], \mu[-i]).$$

Because μ is a mixed-action equilibrium in $\Gamma(n)$, the min in the right hand-side of the above formula is attained at $q[i]$. Therefore μ is a safety-level equilibrium in $H_\Gamma(n)$. An analogous argument proves the if part of the lemma. □

THEOREM 5. *Let Γ be a resource selection system in which the resource cost functions are non-decreasing. $H_\Gamma(n)$ has a unique symmetric safety-level equilibrium. In this symmetric safety-level equilibrium every player is using the strategy p^n, where p^n is the unique symmetric-equilibrium action in $\Gamma(n)$.*

PROOF. The proof follows directly from Theorem 1 and Lemma 4. □

By Theorem 5, each of the players in $H_\Gamma(n)$ is using the strategy p^n, where p^n is the unique symmetric-equilibrium mixed action in $\Gamma(n)$. However, the cost of each active player in $H_\Gamma(n)$ is not $c^n = c^n(p^n)$, it depends on the true state. If the true state is K, that is K is the set of active players, and $|K| = k$, the cost of each active player i is $c^k(p^n)$. It is worthy to compare this cost with the cost $c^k = c^k(p^k)$ that every player in K would have paid had the players in K known the state. We make these comparison in linear models. We say that a resource selection system is *linear* if for every resource j there exists a constant d_j such that $w_j(k) = w_j(1)+(k-1)d_j$ for every $k \geq 1$. For every number of players, n, the associated resource selection game, $\Gamma(n)$, as well as the associated resource selection game with unknown player set, $H_\Gamma(n)$ will be called *linear* too. Note that in a linear system, w_j is increasing if and only if $d_j > 0$, and w_j is non-decreasing if and only if $d_j \geq 0$.

THEOREM 6. *Let Γ be a linear resource selection system with increasing resource cost functions. For every $k \geq 2$ let p^k be the unique symmetric equilibrium in $\Gamma(k)$. There exist an integer $N = N(\Gamma)$, $N \geq 2$ such that for all $n > k \geq N$:*

1. *$c^k(p^k) \geq c^k(p^n)$.*
2. *All inequalities above are strict if and only if there exists $j_1, j_2 \in M$ such that $w_{j_2}(1) \neq w_{j_1}(1)$*

PROOF. 1. Let $j \in M$. By lemma 3 there exist an integer N such that for every $n \geq N$ the unique symmetric equilibrium of $\Gamma(n)$ has full support. By (1) and (3), $c^k = c^k(p^k) = w_j(1) + (k-1)d_j p_j^k$ for every $k \geq N$. Let n, k be integers such that $n > k \geq N$. We begin by showing that $c^k(p^k) \geq c^k(p^n)$. Let $c^k = c^k(p)$.

$$c^k(p^n) = \sum_{j=1}^m p_j^n c^k(p^n, j) = \sum_{j=1}^m p_j^n [c^n - (n-k)d_j p_j^n] =$$

$$c^n - (n-k)\sum_{j=1}^m d_j (p_j^n)^2. \quad (1)$$

It remains to show that

$$c^n - c^k \leq (n-k)\sum_{j=1}^m d_j (p_j^n)^2 \quad (2)$$

We have

$$p_j^n = \frac{c^n - w_j(1)}{d_j(n-1)}, \quad p_j^k = \frac{c^k - w_j(1)}{d_j(k-1)}.$$

Let $A = \sum_{j=1}^m \frac{1}{d_j}$ and $B = \sum_{j=1}^m \frac{w_j(1)}{d_j}$. Since

$$\sum_{j=1}^m p_j^k = \sum_{j=1}^m p_j^n = 1$$

we have that $c^n = \frac{(n-1)+B}{A}$ and $c^k = \frac{(k-1)+B}{A}$. Hence

$$c^n - c^k = \frac{n-k}{A}.$$

Together with (2) it remains to show that

$$A\sum_{j=1}^m d_j (p_j^n)^2 \geq 1.$$

Let $L = \sum_{j=1}^m \frac{(w_j(1))^2}{d_j}$. We have

$$A\sum_{j=1}^m d_j (p_j^n)^2 = A\sum_{j=1}^m \frac{(c^n - w_j(1))^2}{d_j(n-1)^2} =$$

$$\frac{A}{(n-1)^2}[(c^n)^2 A - 2c^n B + L] =$$

$$\frac{1}{(n-1)^2}[((n-1)+B)^2 - 2((n-1)+B)B + LA] =$$

$$\frac{1}{(n-1)^2}[(n-1)^2 - B^2 + LA] = 1 - \frac{B^2 - LA}{(n-1)^2}. \quad (3)$$

It remains to show that $LA - B^2 \geq 0$. This is immediate since for every couple of resources $j, r \in M$ such that $j \neq r$, $(w_j(1))^2 + (w_r(1))^2 \geq 2w_j(1)w_r(1)$.

2. Observe that equality holds if and only if $w_j(1) = w_r(1)$ for every $j, r \in M$.

□

Theorem 6 is applicable to a mechanism design setup in which the organizer knows the number of active players, and the players do not know this number. If the goal of the organizer is to maximize revenue then he is better off revealing his private information. If his goal is to maximize social surplus, then he should not reveal that information. In order to estimate the gain of the players resulting from their ignorance we analyze the function $c^k(p^n)$. The proof of the following theorem is given in the full version.

THEOREM 7. *Let Γ be a linear resource selection system with increasing resource cost functions.There exist N such that for every $n \geq N$ the following assertions hold:*

1. $p_j^n = \frac{n-1+B-w_j(1)A}{Ad_j(n-1)}$, *where* $A = \sum_{j=1}^{m} \frac{1}{d_j}$, *and* $B = \sum_{j=1}^{m} \frac{w_j(1)}{d_j}$.

2. *The minimal social cost in* $\Gamma(k)$ *attained with symmetric mixed-action profiles is attained at* p^{2k-1}. *Consequently,* $c^k(p^n)$ *is minimized at* $n = 2k - 1$.

6. CONCLUSIONS

This paper tackles two fundamental issues in the study of multi-agent systems:

1. Providing solution concepts for multi-agent interactions with incomplete information, where exact probabilistic information is unavailable.

2. Incorporating uncertainty about the number of participants into the context of resource selection systems, one of the most fundamental settings in multi-agent systems.

Our results turned out to be both general and illuminating. On one hand, we provided general existence results for safety-level/competitive ratio equilibria. On the other hand, we were able to apply them to the context of resource selection games. Most importantly, our study led to highly surprising results, showing the positive effect of ignorance about the number of participants on the agents' costs and the system surplus.

Together, our work provides a foundational rigorous study of congestion games with incomplete information, while presenting general tools for the analysis of multi-agent systems where exact probabilistic information is not available.

7. REFERENCES

[1] M. Aghassi and D. Bertsimas. Robust Game Theory. 2004.

[2] I. Ashlagi, D. Monderer, and M. Tenneholtz. On the value of correlation. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 34–41, Corvallis,Oregon, 2005. AUAI Press.

[3] A. Borodin and R. El-Yaniv. *On-Line Computation and Competitive Analysis*. Cambridge University Press, 1998.

[4] S.F. Cheng, D. M. Reeves, Y. Vorobeychik, and M.P. Wellman. Notes on Equilibria in Symmetric Games. Technical Report, University of Michigan, 2004.

[5] R. Engelberg and J. Naor. Equilibria in Online Games. Unpublished manuscript, 2005.

[6] R. M. Harstad, J.H Kagel, and D. Levin. Equilibrium bid functions for auctions with an uncertain number of bidders. *Economics Letters*, 33:35–40, 1990.

[7] R. Holzman, N. Kfir-Dahav, D. Monderer, and M. Tennenholtz. Bundling equilibrium in combinatorial auctions. *Games and Economic Behavior*, 47:104–123, 2004.

[8] R. Holzman and N. Law-yone (Lev-tov). Strong equilibrium in congestion games. *Games and Economic Behavior*, 21:85–101, 1997.

[9] R. Holzman and D. Monderer. Characterization of Ex Post Equilibrium in the VCG Combinatorial Auctions. *Games and Economic Behavior*, 47:87–103, 2004.

[10] N. Hyafil and C. Boutilier. Regret minimizing equilibria and mechanisms for games with strict type uncertainty. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pages 268–277, Arlington, Virginia, 2004. AUAI Press.

[11] H. Konishi, M. Le Breton, and S. Weber. Equilibrium in a Model with Partial Rivalry. *Journal of Economic Theory*, 72:225–237, 1997.

[12] E. Koutsoupias and C. Papadimitriou. Worst-Case Equilibria. In *In Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413, 1999.

[13] D. Levin and E. Ozdenoren. Auctions with Uncertain Number of Bidders. To appear in 'Journal of Economic Literature', 2004.

[14] S. Matthews. Comparing auctions for risk averse buyers: a buyers point of view. *Econometrica*, 55:633–646, 1987.

[15] R. P. McAfee and J.McMillan. Auctions with a stochastic number of bidders. *Journal of Economic Theory*, 43:1–19, 1987.

[16] I. Milchtaich. Congestion Games with Player-Specific Payoff Functions. *Games and Economic Behavior*, 13:111–124, 1996.

[17] P. Milgrom and R. Weber. A Theory of Auctions and Competitive Bidding. *Econometrica*, 50:1089–1122, 1982.

[18] D. Monderer and L.S. Shapley. Potential Games. *Games and Economic Behavior*, 14:124–143, 1996.

[19] R. B. Myerson. Population uncertainty and Poisson games. *International Journal of Game Theory*, 27:375–39, 1998a.

[20] R. B. Myerson. Extended Poisson games and the Condorcet jury theorem. *Games and Economic Behavior*, 25:111–131, 1998b.

[21] R. B. Myerson. Large Poisson games. *Journal of Economic Theory*, 94:7–45, 2000.

[22] T. Quint and M. Shubic. A Model of Migration. working paper. Cowles Foundation, Yale University, 1994.

[23] R.W. Rosenthal. A Class of Games Possessing Pure-Strategy Nash Equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[24] Y. Shoham and M. Tennenholtz. Artificial Social Systems. *Computers and Artificial Intelligence*, 14:533–562, 1995.

[25] Y. Shoham and M. Tennenholtz. On Social Laws for Artificial Agent Societies: Off-Line Design. *Artificial Intelligence*, 73:231–252, 1995.

[26] Moshe Tennenholtz. *Efficient Representation and Reasoning in Multi-Agent Systems*. PhD Thesis, Weizmann Institute of Science, Israel, 1991.

[27] M. Voorneveld, P. Borm, F. van Megen, S. Tijs, and G. Facchini. Congestion Games and potentials reconsidered. *International Game Theory Review*, 1:283–299, 1999.

Adaptive Distributed Resource Allocation and Diagnostics Using Cooperative Information-Sharing Strategies

Partha S. Dutta, Nicholas R. Jennings, and Luc Moreau
School of Electronics & Computer Science
University of Southampton
Highfield, Southampton SO 17 1BJ, U.K.
{psd,nrj,l.Moreau}@ecs.soton.ac.uk

ABSTRACT

A major challenge in efficiently solving distributed resource allocation problems is to cope with the dynamic state changes that characterise such systems. An effective solution to this problem should be able to detect state changes and determine why they occur (diagnosing the cause) in order to adapt to the prevailing situation. Now, since agents typically have localised views and communication constraints that prohibit global instantaneous synchronisation, we argue that cooperative information-sharing can provide them with the necessary adaptiveness and diagnostics ability. To this end, we develop a novel information-sharing algorithm for resource allocation tasks by building upon the most effective algorithm currently available in this domain. Then, using empirical analyses on a resource allocation application with dynamic state changes, network call routing with network failures, we show that, compared to the benchmark, our new algorithm achieves up to a 20% increase in call throughput, up to 3.5 times faster throughput recovery after failures, and provides a novel mechanism for distributed failure diagnosis without false positives and false negatives.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Experimentation, Theory

Keywords

Resource allocation, state diagnosis, information-sharing

1. INTRODUCTION

Resource allocation is a core functionality of many distributed systems [5, 8]. In such systems, multiple agents typically allocate resources under a decentralised regime using their individual localised views of the system, in order to complete their tasks. To do so, they use estimates of system states to ensure their individual actions are coordinated [2]. Now, to do this effectively in complex environments (where states change dynamically), such estimation necessarily involves detecting where, when, and why changes occur. Thus, to maintain these estimates and to perform such diagnostics, the agents need to share information. In particular, they need to communicate their individual views of the system to one another. Such information can then be used to generate robust local estimates and allow the agents to adapt these when states change. To this end, we develop and empirically evaluate a novel information-sharing algorithm that allows distributed agents to adapt dynamically to state changes and to perform distributed state diagnostics.

Given its importance, several strands of work on cooperative MAS have focused on analysing how communication can generate better solutions to distributed coordination [1, 6, 7]. However, in these, several limitations exist. For example, in [7], the authors develop a theoretical model for optimising information-sharing with the assumption that the agents need to synchronise their views periodically. In a typical distributed application, however, the agents should be able to act in a decentralised fashion without requiring to synchronise. The work in [6] considers how communication can improve the expected utility of an individual agent. We argue, however, that in cooperative systems, communication should be targeted towards improving overall system performance. Finally, the partial global planning algorithm in [1] allows the agents to communicate partial plans, representing their individual views of the system, to ensure consistent global solutions. Nevertheless, this work uses a set of pre-determined coordination mechanisms which can be inadequate for the agents to adapt to all possible environment dynamics that cannot be predicted a priori.

In our previous work, we addressed these limitations by developing an information-sharing protocol (the *post task-completion* (PTC) protocol) for distributed resource allocation [3]. Specifically, we showed that such cooperative information-sharing can generate highly effective distributed resource allocation. Our algorithm outperformed a range of state-of-the-art algorithms (including Littman's Q-routing, Stone's TPOT-RL, and a broadcast protocol) that have previously been shown to perform highly effective distributed task processing in dynamic systems. In [3], we chose an exemplar resource allocation application, that of call routing in resource-constrained mesh networks, to verify the performance advantage of our solution in a real application and demonstrated that our solution is more generally applicable to all such problems using a formal analysis. However, PTC did not produce the adaptiveness required in highly dynamic resource allocation problems by ensuring effective state diagnostics. Thus, in this paper, we extend PTC (to e-PTC) to remove these limitations. This necessarily implies an improvement over the benchmarks used in [3].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

In more detail, e-PTC allows the agents to share information both when they successfully complete tasks and fail to do so. In contrast, PTC only allows agents communicate state information after they successfully complete tasks. In so doing, they are able to communicate their local states and build state estimates when the system is functioning under "steady state". However, unpredictable incidents such as process failures, network node saturation, and hardware crashes, can lead to subsequent failures in task processing. To adapt under these conditions, which are common in many distributed systems [10], the agents need to maintain estimates of the dynamic state changes and diagnose states for efficient recovery. Accordingly, we argue that such capabilities can be incorporated by allowing information-sharing after task failures. Indeed, by developing e-PTC, this paper extends the current state-of-the-art in the following significant ways: (1) It provides a simple but effective means of providing highly adaptive behaviour among distributed agents in dynamic environments. (2) It establishes, using empirical analyses, the advantage of e-PTC in providing effective resource allocation in a real application: call routing in mesh networks. Specifically, e-PTC generates up to 20% higher call throughput than PTC when network failures occur and allows up to 3.5 times faster recovery of throughput after failures. (3) A novel state-diagnosis algorithm is developed, based on e-PTC, that allows distributed agents with localised views to diagnose network failures correctly (no false positives) and only when failures occur (no false negatives).

In the rest of this paper, section 2 briefly describes the distributed resource allocation task in the context of our application domain and reviews how information-sharing aids state estimation. Then, section 3 describes e-PTC and how it generates effective adaptiveness in highly dynamic systems. The experimental setup is described in section 4 and the results in section 5. Section 6 develops the state-diagnosis algorithm using e-PTC and presents empirical results on its performance. Section 7 concludes.

2. INFORMATION-SHARING IN RESOURCE ALLOCATION TASKS

The task of decentralised routing in mesh networks requires individual agents to allocate node bandwidth (representing the resources in the system) and forward a call to a neighbouring node (the next hop along the call route) such that the call reaches the destination when a set of end-to-end bandwidth is allocated. To forward a call, an agent chooses that neighbour for which it estimates that the call would be placed via the most efficient overall route. Thus, each agent maintains an estimate of the bandwidth availability along paths to any other node. This reflects its partial view of the overall network. It attempts to keep these estimates up-to-date as much as possible by using the information exchanged between agents. The more up-to-date the estimates with respect to the actual network states, the higher the quality of routing and, thus, the better the network performance.

More specifically, each node i in a network of A nodes, maintains a routing table RT_i, where each element $RT_i(d,n)$ ($d \in A$ is the destination node, and $n \in A$ is a neighbour of i) represents i's estimate of the best end-to-end bandwidth availability in going from n to d. Node i chooses a neighbour j using a Boltzmann distribution over its neighbours based on the RT_i values: $e^{\frac{-RT_i(d,j)}{\tau}} / \sum_n e^{\frac{-RT_i(d,n)}{\tau}}$, where τ is the "temperature" parameter used to set the skewness of the distribution. When a call forward request reaches the destination, the latter sends "upstream" along the call route an `ack` message indicating a successful call connection. Otherwise, if a call forwarding fails before reaching the destination, the node at which it fails sends a `drop` message "upstream". As each node i transmits the `ack` or `drop`, it appends its state value s_i to the message. In this application, a node's state is the bandwidth available at that time expressed as a fraction of the maximum available bandwidth.[1] Therefore, a node (say, k) receiving an `ack` or `drop` from the immediate "downstream" node (say, l) gets the set of state values that were appended by all "downstream" nodes on this call route. These state values are then used by k to update its RT as: $RT_k(d,l) \leftarrow (1-\alpha)RT_k(d,l) + \alpha\Gamma(s_l, ..., s_d)$.[2] Here, $\Gamma()$ is an aggregation function on a set of node state values, and α is a discount factor ($0 < \alpha < 1$) that puts different weights on previous and current information. In particular, we choose Γ as the *minimum* of the set of node states since the node with the minimum available bandwidth determines the maximum number of calls that can be accommodated along a given route. Figure 1 presents a schematic of the above description.

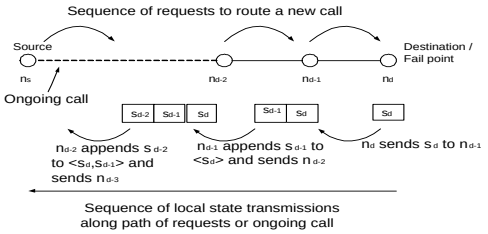


Figure 1: Call setup and information exchange

In this system, state information is exchanged between nodes *after every call connection and failure*. This is the definition of e-PTC (see [2] for more details of the design of e-PTC on this application). Note, the original PTC allowed information-sharing only after a call connection. Thus, e-PTC expands the set of events that PTC uses for sharing information.

3. ADAPTIVENESS IN HIGHLY DYNAMIC SYSTEMS

The e-PTC algorithm dictates that agents should share information after both successful task completions and task failures. Sharing information after task completion provides e-PTC with the same advantage as that of PTC: effective resource allocation when the system states are steady. However, when unforeseen events such as network failures cause task processing to fail, e-PTC allows the agents to share state information and thus allows for the estimation of the state changes caused by such events and diagnosis of the events. These capabilities cannot be achieved using PTC alone. Thus, the e-PTC algorithm, together with the estimate-generation mechanism discussed in section 2, provide effective adaptive behaviour in dynamic systems which is a significant improvement over the current state-of-the-art. In the rest of this section, we first describe the model we adopt to define network failures that cause dynamic state changes and, then, describe the design of e-PTC in the context of call routing with failures.

3.1 Model of Dynamic State Changes

Network failures cause dynamic state changes in our application. In the failure model, there are the following components:

[1] As nodes route new calls or existing calls terminate, available bandwidth (hence, node state) varies dynamically.

[2] This mechanism of computing a discounted running average is used to learn estimates in environments where the dynamics are not known a priori [11].

Failure Characterisation: Only **stop** failures are considered because these are the most commonly studied in distributed systems [9] and, hence, form a good testbed for measuring the performance of e-PTC. We assume that a failed node (1) cannot run any processes, (2) cannot communicate with other nodes, and (3) once failed, continues to remain so. Thus, after a failure, all ongoing calls on a node will be lost and it will not be able to detect or acknowledge new call forward requests. A failure would cause both bandwidth availability (by removing existing calls) and the topology to change, thereby, changing the network state.

Detection Characterisation: After a node fails, one of its neighbours actually detects that it has failed after the latter makes a forwarding request. The requesting node detects the failure status since it does not receive any response to its request. It then generates `drop` messages to inform its neighbouring nodes to deallocate the reserved bandwidth for the calls that were ongoing along this affected route.

3.2 Adaptiveness and Diagnostics with e-PTC

As identified in section 3.1, a failure causes network states to change dynamically. The e-PTC algorithm, by distributing state information after failures, allows the agents to assess these state changes. This, in turn, lets the agents adapt their task processing to best respond to state changes and to diagnose where in the system the failure has occurred.

In more detail, figure 2 shows the schematic of a node failure and the subsequent transmission of state information by its neighbour. In this figure, node f is shown to have failed. One of its neighbours, i, detects that f has failed after it attempts to forward a call request to f and does not get a response. Thus, it deallocates its bandwidth for all calls that were being routed through both i and f, and that for the new call it was trying to set up. Also, it informs each j (a neighbour of i that is also on the path of calls via i and f) by sending a `drop` message to de-allocate bandwidth. Each such j continues to propagate the `drop` message until the terminal node on that path is reached. Note that, a similar process is undertaken by each neighbour of f whenever they detect that f has failed. In the above process, after receiving a `drop` message, a node updates its estimate (see section 2 for the estimate-update mechanism) about the affected network path from the node state values appended to the message by each agent transmitting it. In this manner, the nodes learn that the bandwidth availability along the affected path has changed. Such information, therefore, allows them to adapt their routing behaviour to best respond to the state dynamics.

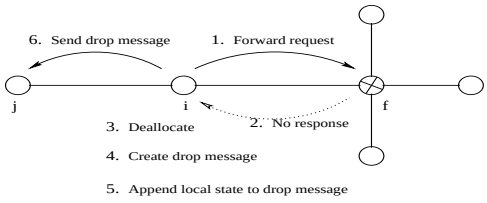


Figure 2: Network failure detection schematic

Importantly, using this information, a node can diagnose if a certain node in the network has failed. Note, when i transmits state information with a `drop` message, it assigns a value of zero for f's state. Alternatively, when a node has no available bandwidth, it rejects a forwarding request and, therefore, the call attempt fails. Similar to a node failure, the state value transmitted with a `drop` message in this case is also zero. Therefore, for correct failure diagnosis, an agent should be able to distinguish between the information (zero) received from a node that has failed and that is saturated (also zero). The key to do this is to observe a *sequence* of state values. Note, to indicate a failure, a separate message (e.g., `fail`) could be used instead of a zero state. However, the state values are used to update the routing tables. Hence, a separate message to explicitly indicate failure would not be useful for state estimation. In fact, as shown in section 6, the zero state values can be used for diagnosing failures in addition to state estimation. In section 6, we develop an algorithm that computes the *number of consecutive state values* an agent needs to monitor from another node to achieve correct diagnosis. It is shown that this algorithm is both (i) without false positives — a failure is diagnosed only if one has occurred, and (ii) without false negatives — once a failure occurs, a positive diagnosis is guaranteed.

4. EXPERIMENTAL SETUP

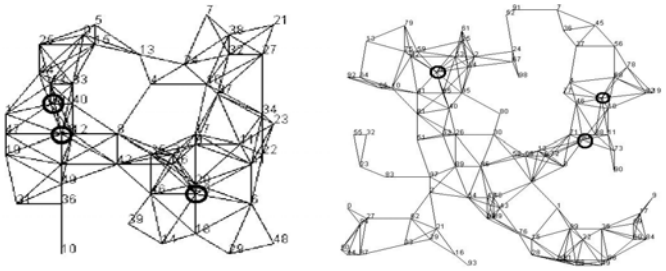
This section first outlines the simulation of a limited-bandwidth mesh network application used for call routing with failures. This system effectively emulates distributed resource allocation tasks under highly dynamic environments (see [2] for additional justification). Then, we go on to present the criteria used to evaluate the performance of e-PTC.

4.1 The Simulation Environment

Each node has a fixed maximum bandwidth (C units). This captures the limited bandwidth of mesh networks. Each call takes one bandwidth unit. The probability (p_c) of a call originating at any node is termed the "load". Call destinations are selected uniformly. A node reserves one bandwidth unit for a call after forwarding it to a neighbour. The reservation is removed if the call fails to connect or terminates after the duration t_l, for which the call lasts, measured since it is routed to its destination. Before forwarding, an agent inserts its id to the call so that a call has a sequence of ids used to transmit state information after a connection or failure. If connected, an acknowledgement (`ack`) message is transmitted from the destination up to the source. In case of a failure, the failed node's neighbour transmits a `drop` message along the call path and bandwidth reserved for this call is de-allocated. Nodes share state information by appending state values to an `ack` or `drop` message: only `ack` is used by PTC, while e-PTC uses both. Numerically, a node state is the bandwidth units available expressed as a fraction of the maximum state estimate (C). Thus, if it has x units available, its state is $\frac{x}{C}$. The routing tables are initialised to 1.0, the maximum availability. One simulation run consists of 2×10^6 simulated time steps (time is measured on a global clock) where in one step any agent can send one message to one of its neighbours. Experiments are run for a sufficient number of runs to ensure statistical significance at the 95% confidence level. The following parameter values are chosen: a network with a total of $N = 50$ nodes (figure 3(a)), and $N = 100$ nodes (figure 3(b)), $C = 10$, $\alpha = 0.03$, $\tau = 0.1$, $p_c = \{0.1, 0.2, 0.4, 0.6, 0.8\}$ (to test the effect of different loads), and $t_l = 50$ for the 50-node network and $t_l = 100$ for the 100-node network. Experiments have been conducted on several other topologies and similar trends are observed in the results [2]. Here, therefore, we report results from these sample topologies. Also, the same general observations hold for other values of α and τ.

4.2 Performance Evaluation Criteria

The following criteria are used to compare the performance of e-PTC against that of PTC to determine its adaptiveness in dynamic resource allocation tasks.



(a) 50-node topology (b) 100-node topology

Figure 3: Network topologies

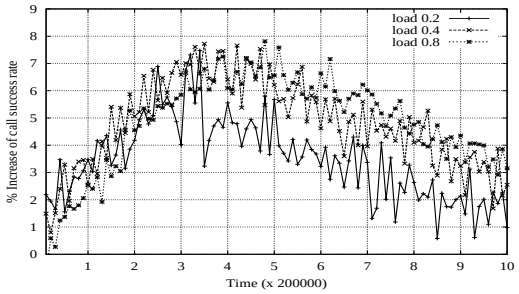
Call Success Rate (CSR): The advantage of having a highly adaptive solution in dynamic systems is to have a high rate of successful task processing. In the call routing application, this is measured in terms of the call success rate. Specifically, if, in a given time interval, K calls originate out of which k are successfully connected, the CSR within that period is: k/K. The higher the CSR in a given system, the better the performance of the information-sharing algorithm by maintaining high-quality routing tables.

Recovery of Call Success Rate: Good adaptiveness in dynamic systems should allow quick recovery of performance after state changes occur. Now, in our application, after a failure, the CSR will necessarily drop since a set of network paths become unavailable. However, as the agents adapt their estimates and find alternative routes, the CSR should recover from the initial drop. The efficiencies of e-PTC and PTC are compared by measuring how fast the CSR recovers following failures. Thus, we compare the CSR (denoted as r_{fail}) when a node fails in the simulation (termed the "test" case) to the CSR (r_{miss}) when the corresponding node is removed from the topology from the start of the simulation (termed the "baseline" case) keeping all other parameters identical. In the baseline, the agents learn the routing tables based on the topology that results after failure in the test simulation. Now, if all agents had global information, then, after failure, r_{fail} could recover and become equal to r_{miss}. But, since the agents do not have this, r_{fail} is unlikely to be able to recover to exactly r_{miss}. Nevertheless, the closer and faster it can get to r_{miss}, the more efficient the system.

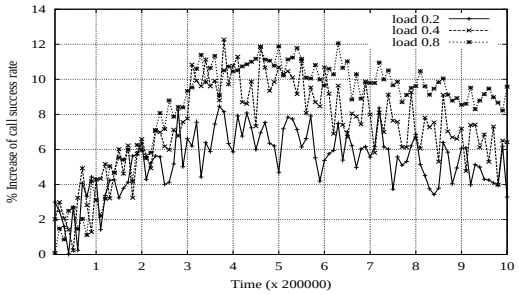
Message Size (MS): Any advantage of e-PTC over PTC in CSR or CSR recovery is due to sharing more information. Although one message is transmitted for a call success (`ack`) and one for a call failure (`drop`) in both e-PTC and PTC, these may not be of the same size. In both cases, the message size is determined by the number of state values appended. Thus, the average size of k messages is $\sum_{i=1}^{k}(l_i)/k$, where l_i is the size of the ith message. To assess if e-PTC has an overhead over PTC, we compare the average size of messages generated in both.

5. RESULTS AND ANALYSIS

In this section, first, the CSR and MS of e-PTC and PTC are compared in the absence of failures to test if e-PTC has any inherent advantage (a higher CSR) or overhead (higher MS). The second set of experiments then measure their performances when dynamic state changes occur due to failures. Since it has already been shown that PTC performs better than a range of benchmark algorithms widely used for adaptive distributed resource allocation tasks [3], any improvement in e-PTC over PTC necessarily indicates an improvement over the current state-of-the-art.



(a) % CSR increase in 50-node topology

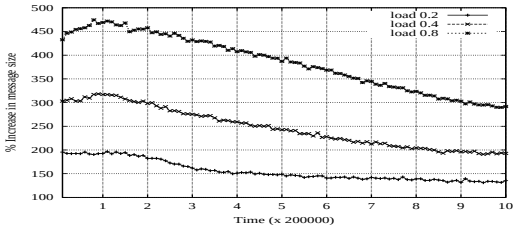


(b) % CSR increase in 100-node topology

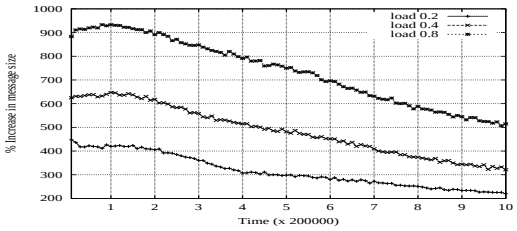
Figure 4: CSR increase in e-PTC over PTC without failures

5.1 No State Change Dynamics (No Failures)

The time-variation of the network call success rates for e-PTC and PTC are compared by calculating the percentage deviation of the e-PTC CSR from that of PTC: $(\texttt{CSR(ePTC)} - \texttt{CSR(PTC)})/\texttt{CSR(PTC)}$. These values are plotted for different network load values in figure 4. Now, the motivation for designing e-PTC is to generate effective adaptation of the system in response to dynamic state changes. However, the results in figure 4 show that *even without such dynamic state changes, e-PTC has an advantage over PTC in terms of generating higher CSR*: in figure 4(b), at load 0.8, e-PTC achieves a peak 11% increase in CSR compared to PTC. When no failures occur in the network, the source of state-change dynamics is due to network saturation when some nodes are unable to transmit further calls. Now, this obviously happens more at higher loads. Thus, the call success rates of both PTC and e-PTC decrease with increasing load. Note, in e-PTC, nodes share information after call failures. Thus, with increasing load and, hence, more call failures due to network saturation, e-PTC allows more information-sharing and ensures that the routing table estimates remain more up-to-date with the actual bandwidth availability. This allows e-PTC to adapt to the dynamics caused due to saturation better than PTC. The CSR of e-PTC does not decrease as much as that of PTC with increasing load; in figure 4, we observe the relative improvement of e-PTC CSR over PTC CSR to increase with increasing load.



(a) % MS increase in 50-node topology



(b) % MS increase in 100-node topology

Figure 5: MS increase in e-PTC over PTC without failures

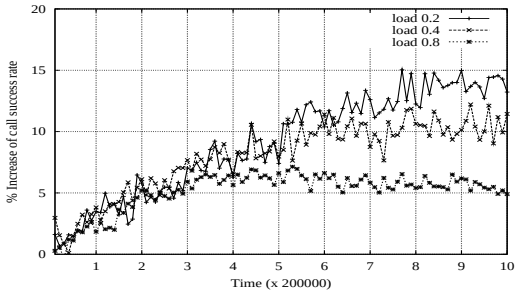
The average message size of e-PTC and PTC are compared in an analogous manner in figure 5. These results show that e-PTC incurs more message overhead than PTC and it increases with load. So, in figure 5(a), the e-PTC MS is about 2.3 times of PTC MS at load 0.2, and it is about 3.9 times at load 0.8. Now, e-PTC communicates state information when calls fail to keep the routing tables updated. However, this causes the MS (of the `drop` messages) to increase. This effect becomes more pronounced at higher loads when more calls fail (as explained before). Therefore, e-PTC attains a higher CSR at a cost of larger MS compared to PTC.

5.2 With Dynamic State Changes (Failures)

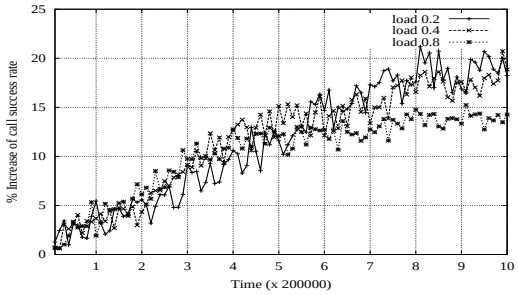
State-change dynamics are incorporated in our simulator by way of network failures. We allow one node to fail at a given time and different failures to occur at equidistant points in an experiment. More arbitrary failure patters (e.g., several failures occurring at close intervals) have been used in our ongoing work that show similar broad trends. The nodes that fail are pre-selected. However, different combinations of nodes are selected to fail in different experiments. Note, the failures are initiated by the simulator and no agent knows when and where one would occur. The results reported here use failures of (i) 3 nodes with the highest edge connectivity (encircled in figure 3), and (ii) 3 randomly chosen nodes with average edge connectivity. The motivation for choosing (i) is that the nodes with the most connectivity are critical points and heavily used for routing. Thus, their failures cause a major disruption in the network state which provides a "hard" test for the advantage, if any, that e-PTC has over PTC. However, (ii) provides a more typical scenario. We report results using (i) since the results from (ii) show identical trends.

5.2.1 Call Success Rate.

The percentage increase of call success rate of e-PTC over PTC with the three most-connected nodes failing are shown in figure 6. As explained in section 5.1, the CSRs of both e-PTC and PTC decrease with increasing load. There is an additional drop due to node



(a) % CSR increase in 50-node topology



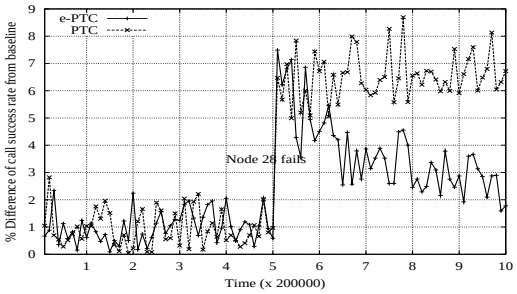
(b) % CSR increase in 100-node topology

Figure 6: CSR increase in e-PTC over PTC with 3 most-connected nodes failing

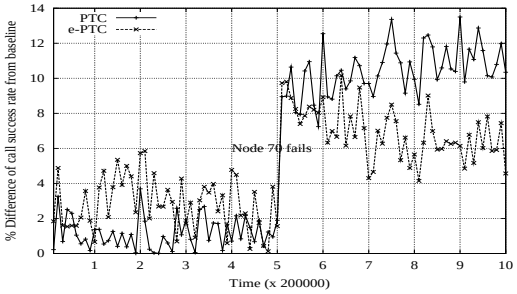
failure since it results in less bandwidth being available. However, *the CSR for e-PTC is always higher than that for PTC, confirming the effectiveness of e-PTC over PTC*. The difference in their performances is, however, more pronounced at lighter loads because at high loads the effects of both saturation and failures offset the advantage achieved by e-PTC over PTC (unlike section 5.1 where a higher relative improvement was observed at higher loads). Thus, in figure 6(b), after all 3 nodes fail, the CSR of e-PTC is about 20% higher than PTC at load 0.2 and about 14% at load 0.8. Moreover, *at a given load, the relative improvement in CSR of e-PTC over PTC increases with the number of failures*. Hence, although the CSRs decrease with failures, *the rate of deterioration in e-PTC is lower than in PTC*. These results indicate that our e-PTC algorithm generates better adaptiveness in response to state changes caused by failures. This is the reason for the networks to achieve higher call success rates with e-PTC compared to PTC in such dynamic conditions.

5.2.2 Recovery Rate.

To evaluate the efficiency with which e-PTC and PTC allow the network call success rate to recover after failures occur, we compare the CSRs of the e-PTC(test) and PTC(test) strategies against that of the baseline strategy (see section 4.2 for the definitions of these). Specifically, the percentage difference between the CSRs of e-PTC(test) and the baseline, and that between the CSRs of PTC(test) and the baseline are computed as: $|\ CSR_{base} - CSR_{test}\ | / CSR_{base}$. Figure 7 shows these results, generated when the highest-connected node in the corresponding networks fail halfway through the simulation and a load of 0.1 is used. Similar trends have been observed with all other load values. Now, before the failure, e-



(a) CSR recovery in 50-node topology



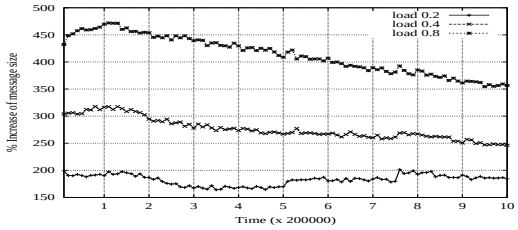
(b) CSR recovery in 100-node topology

Figure 7: Recovery of CSR after failure

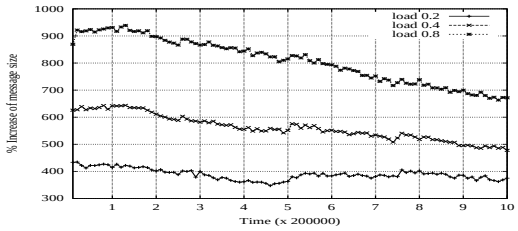
PTC(test) has the highest CSR (similar to section 5.1). However, after the failure in the test cases, the CSRs of both e-PTC(test) and PTC(test) drop. However, in figure 7, the CSR of e-PTC(test) recovers at a faster rate towards the baseline than PTC(test). The percentage deviation of the CSRs in both sets of graphs of figure 7 are observed to decrease faster for e-PTC than PTC. For example, in figure 7(a), at the end of a simulation run, e-PTC(test) CSR reaches within about 2.0% of the baseline CSR which is 3.5 times (350%) better than that of PTC(test) which reaches within 7% of the baseline. These results, therefore, confirm that e-PTC ensures a *superior efficiency in the recovery of system performance after state-changes occur.* Thus, better adaptiveness to dynamic environments is indeed achieved by e-PTC than PTC.

5.2.3 Message Size.

Sections 5.2.1 and 5.2.2 have confirmed that using e-PTC, distributed agents with localised views can achieve significantly better adaptiveness when states change dynamically. This advantage of e-PTC is generated by sharing information about the state changes, a feature lacking in PTC. Thus, the message overhead of e-PTC exceeds that of PTC. Specifically, figure 8 shows the percentage increase in message size of e-PTC compared with that of PTC, computed in a way analogous to the results in figure 5. However, the results in figure 8 are generated when the three highest-connected nodes in the corresponding topologies fail. These results show that MS *of e-PTC is significantly higher than PTC.* Also, they show that the higher the load, the greater is the increase in the MS of e-PTC relative to PTC: in figure 8(a), at the end of a simulation run, e-PTC MS is about 2.8 times that of PTC at load 0.2, whereas it is about 4.6 times at load 0.8. As mentioned earlier, e-PTC incurs the extra overhead in message size since it uses `drop` messages



(a) % MS increase in 50-node topology



(b) % MS increase in 100-node topology

Figure 8: MS increase with 3 most-connected nodes failing

to communicate the state-change information. Such information-sharing, while allowing the routing tables to be more up-to-date compared to PTC, leads to larger messages in e-PTC. As the number of call failures increase with load and network failures, so does the number of state-value transmissions in `drop` messages. One way of reducing the message overhead of e-PTC is to selectively communicate information. We are developing such a mechanism in our ongoing work, where agents use their communication histories to judge how useful a certain information would be to a recipient. Significant savings in message overhead are observed while maintaining the advantages of e-PTC.

6. STATE DIAGNOSIS USING E-PTC

The results of section 5 indicate how e-PTC ensures that agents *adapt* effectively to dynamic state changes caused by failures. In this section, we elaborate on the additional benefit of e-PTC in *diagnosing* such state changes. In particular, we develop an algorithm, based on the shared information generated by e-PTC, that distributed agents can use to detect if a certain node has failed. Such a capability would then be useful for automating distributed diagnosis and recovery.

Here, failure diagnosis means that a node raises an alert when it interprets, from the state values it receives along `drop` messages, a fail condition of another node. The detecting node is termed the "monitoring" node and the one interpreted is the "monitored" node. Note, any node can take either of these roles because information is distributed between all nodes in the network. This is because calls can originate from and terminate at all nodes and, thus, e-PTC would distribute information along call paths between all node pairs. Our aim in this context is to ensure that failure diagnosis using e-PTC is both truly indicative of a failure (no false positives), and that all failures will be detected (no false negatives). To ensure the first property, the monitoring node needs to observe a series of state values of the monitored node (explained shortly). The second property guarantees a detection.

To explain the diagnosis process, a node n_a is considered to do the monitoring of another node n_b (the monitored node). Let T be the average interval at which n_a receives the state values of n_b, distributed by e-PTC. The state values of n_b can be transmitted along `ack` and/or `drop` messages either when n_b is saturated or unsaturated, or when the latter has failed. Now, the state values of a node that is saturated and that is failed are both zeros (section 3.2). However, the difference is that a saturated node becomes unsaturated when bandwidth becomes available, whereas a failed node continues to remain so. Therefore, after receiving the first zero state value of a saturated (not failed) n_b, if n_a continues to observe the state values, then it will receive a non-zero state value after a *certain number of observations*. This will happen when n_b becomes unsaturated. Against this background, note that if n_a has the information about the average durations for which n_b remains alternately saturated and unsaturated, then it can compute the number of such observations using its knowledge of T. We assume that n_b remains alternately saturated for an average period of $\hat{t}_s$ and unsaturated for $\hat{t}_{us}$. To summarise, when n_a receives a zero state value of n_b, it assumes n_b is saturated. Then, using the knowledge of T, $\hat{t}_s$, and $\hat{t}_{us}$, n_a computes *the minimum number of observations* (termed as m) *after which it would receive a non-zero state value* of n_b. If the actual number of zero-state values of n_b received exceeds the computed value, then n_a interprets a failure by rejecting its previous assumption of a saturated n_b. The algorithm to compute m is detailed in the following section. Now, out of T, $\hat{t}_s$, and $\hat{t}_{us}$, n_a can estimate online the average interval (T) at which it receives information from n_b. However, it cannot estimate online $\hat{t}_s$ and $\hat{t}_{us}$ of n_b since it does not have information about the overall network load. But, n_a can acquire these estimates from node traffic statistics that are usually available for most networks [4].

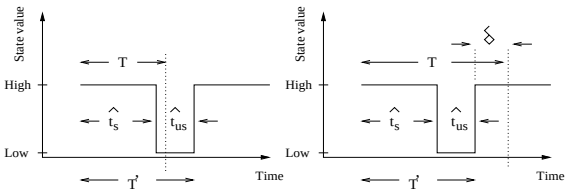
6.1 The Failure Diagnosis Algorithm

After receiving the first zero state of n_b, n_a invokes the algorithm assuming the state value to be from a $\hat{t}_s$ interval of n_b (i.e., from a saturated n_b; it may of course be from a failed n_b). Now, knowing that it would receive state values from n_b after every T units and that n_b would remain alternately saturated for $\hat{t}_s$ and unsaturated for $\hat{t}_{us}$, n_a determines whether the next reception should be from a $\hat{t}_s$ or a $\hat{t}_{us}$ interval of n_b. This is done by comparing T against the sum of $\hat{t}_s$ and $\hat{t}_{us}$. This process is repeated until the next projected reception is from a $\hat{t}_{us}$ interval; the total number of receptions before this happens is equal to m. In more detail, figure 9 shows the algorithm. The algorithm's inputs are T, $\hat{t}_s$, and $\hat{t}_{us}$ and the output is m, the minimum number of zero state values n_a would observe from n_b. In figure 9, T' is the sum of $\hat{t}_s$ and $\hat{t}_{us}$. The algorithm identifies the various possible relationships between T and $\hat{t}_s$ and $\hat{t}_{us}$ to decide how to compute m.

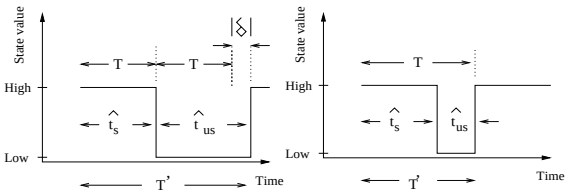
Input: T, $\hat{t}_s$, $\hat{t}_{us}$; Output: m.

0. $T' \leftarrow \hat{t}_s + \hat{t}_{us}$; $m \leftarrow 1$;
1. if($T < T'$ and $T > \hat{t}_s$)
2. $m \leftarrow 2$; //2nd msg in unsaturation
3. else if ($T > T'$) //next msg in a subsequent T' zone
4. $\delta \leftarrow T \% T'$; //modulus
5. while($(\delta - \hat{t}_s) < 0$)
6. $m \leftarrow m + 1$;
7. $\delta \leftarrow (T - (T' - \delta)) \% T'$; //offset from start of next interval
8. else if($T \leq \hat{t}_s$) //next msg in a saturation zone
9. $\delta \leftarrow 0$;
10. do
11. $m \leftarrow m + \frac{\hat{t}_s - \delta}{T} + 1$; //integer division
12. $\delta \leftarrow T(\frac{\hat{t}_s - \delta}{T} + 1) - (T' - \delta)$; //offset from end of current interval
13. while($\delta > 0$);
14else
15. $m \leftarrow \infty$; //to observe infinite zero-state messages; diagnosis inconclusive

Figure 9: Failure diagnosis algorithm



(a) Condition: $T < T'$ and $T > \hat{t}_s$ (b) Condition: $T > T'$



(c) Condition: $T \leq \hat{t}_{us}$ (d) Condition: $T = T'$

Figure 10: Relationships between T, $\hat{t}_s$, and $\hat{t}_{us}$ of figure 9

First, if $T < T'$ and $T > \hat{t}_s$ (line 1), the next state value of n_b (after an interval of T since the first reception) would be from its $\hat{t}_{us}$ interval and, hence, $m = 2$. Figure 10(a) shows this condition in schematic. Here, the saturated and unsaturated states of n_b are shown as a pulse wave: the "high" value represents the $\hat{t}_s$ interval and "low" the $\hat{t}_{us}$ interval. Along the x-axis is time and the y-axis represents state value. Thus, from the start of the first $\hat{t}_s$ interval (corresponding to the first zero state value n_a received from n_b), the end of a period equal to T (which is when n_a would receive the next message from n_b) falls within the $\hat{t}_{us}$ interval; which is when n_a will receive a non-zero state value. Thus, n_a would receive a non-zero state value on the message after the first zero it received from n_b. Hence, $m = 2$.

Second, if $T > T'$ (line 3), first, it is computed by how much (δ) the next reception is within the next T' of n_b (line 4). Then, this offset is recomputed (line 7) for successive observations (the loop at line 5) until it is greater than $\hat{t}_s$ of the next T' (line 5), which indicates a non-zero state. Until this happens, in each iteration one more zero state value needs to be observed by n_a (line 6). Figure 10(b) shows the schematic.

Third, if $T \leq \hat{t}_s$ (line 8), a similar calculation is done as above. Specifically, in line 12, $T(\frac{\hat{t}_s - \delta}{T} + 1)$ gives the total duration of the consecutive observations made within an offset-adjusted $\hat{t}_s$ interval, and $(T' - \delta)$ gives the effective (offset-adjusted) T' in which the above calculation is done. Thus, the new offset from the end of the current T' is the difference of the above two values (line 12). The process is repeated until the next observation is in the $\hat{t}_{us}$ of the current T' which is when the offset is less than zero (line 13) and indicates a non-zero state. As an example, consider $T = \hat{t}_s$. Then, δ at the end of iteration 1 is (line 12): $\delta = T(\frac{\hat{t}_s - 0}{T} + 1) - (T' - 0)$, $= 2T - T'$. If $2T < T'$, then $\delta < 0$ (line 13 violated) and, thus, the third message after the first zero state that n_a received will be from a $\hat{t}_{us}$ interval of n_b; so, in this example, $m = 3$. Figure 10(c) shows this example in schematic. If $2T > T'$, then line 13 holds. So, in the next iteration, an offset-adjusted $\hat{t}_s$ and T' are used (where the offset is the δ of the previous iteration) to re-compute a new δ.

Lastly, when $T = T'$ (line 14), the next state from n_b always co-

incides with a $\hat{t}_s$ interval (see the schematic of figure 10(d)). Therefore, n_a receives an endless sequence of zero states from n_b and, hence, cannot distinguish between a saturated and a failed n_b.

The above algorithm can be invoked when n_a receives a zero state of a *saturated* n_b. However, assuming n_a has the correct estimates of T, $\hat{t}_s$, and $\hat{t}_{us}$, it will eventually receive a non-zero state of n_b on the mth instance since the first zero. This is when n_b has become unsaturated. So, n_a would not signal a failure which is the correct diagnosis. Hence, the diagnosis has no false positive. Alternatively, if n_b had indeed failed when n_a received the first zero, it would continue to be failed indefinitely. Therefore, n_a would eventually receive more than m zero state values from n_b and then signal a failure — the correct diagnosis (no false negatives).

6.2 Empirical Evaluation

The algorithm of figure 9 is evaluated empirically using the same experimental setup as in section 5. In a given network, one node (n_a) is selected as the monitoring node and then, in turn, all other nodes are selected to fail (n_b), each in a differect simulation with all parameters the same as before. Selecting one node to fail does not limit our conclusions about our diagnosis algorithm which is theoretically capable of detecting arbitrary number of failures. However, multiple failures can cause a practical difficulty of state information not reaching the monitoring node due to a disconnected network. Now, for every other node, n_a estimates T and the simulator estimates $\hat{t}_s$ and $\hat{t}_{us}$, which are fed to n_a using which, it can invoke the algorithm of figure 9 to compute m as described before. In these experiments, node 8 in figure 3(a) and node 13 in figure 3(b) are used as the monitoring nodes. With a different node (the monitored node) failing in each experiment, the average values of m and T over the monitored nodes that are at a given distance from n_a, are computed. Hence, a summary assessment of how many consecutive observations n_a requires to diagnose failure at a given distance is obtained.

To this end, tables 1 and 2 show the average estimates of m, T, and the delay between the time a node fails and n_a actually making a diagnosis for figure 3(a) and 3(b), respectively. *In each experiment, n_a has diagnosed only after the chosen n_b has failed. Thus, the properties of no false positives and no false negatives of the diagnosis algorithm are corroborated by these empirical results.* Thus, not only does e-PTC yield highly effective adaptive response in distributed dynamic environments, but also using the information exchanged by e-PTC, accurate state diagnosis can be achieved.

Note the diagnosis delay increases with increasing distance of the failures because of the increasing delay (T) in receiving information from distant nodes. The value of T is determined by the topology, load, call pattern, and the information-sharing protocol used. By reducing T, such as by distributing information more frequently and along multiple paths, the detection delay can be reduced; this is part of our future work. The delay is also due to using a single monitoring node. By choosing multiple such nodes, the maximum distance of any node and at least one monitoring node can be restricted such that the delay is within tolerable limits. Nevertheless, we verified the theoretical properties of the diagnosis algorithm using a single monitoring node which actually provides a "harder" test than using multiple ones.

Table 1: Diagnosis performance on the topology of figure 3(a)

Min dist of target	T		m		Detection delay
	avg	stdev	avg	stdev	
1	420	50.7	2.8	1.2	756
2	1559	562.4	2.9	1.4	2962.1
3	7402	5339.3	2.3	1.2	9622.6
4	28616	15895.0	2	0	28616

Table 2: Diagnosis performance on the topology of figure 3(b)

Min dist of target	T		m		Detection delay
	avg	stdev	avg	stdev	
1	372	11.2	2.2	0.3	446.4
2	4148	491.1	2.0	0.1	4148.0
3	10807	658.9	2.0	0.2	10807.0
4	20265	1870.5	2.1	0.4	22291.5
5	33336	4668.6	2.5	1.3	50004.0
6	44569	15631.4	2.0	0.0	44569.0
7	63416	8510.4	2.1	0.6	69757.6
8	132917	15006.6	1.85	0.3	112979.4
9	464743	13766.6	2.0	0.1	464743.0
10	425047	21690.3	2.0	0.1	425047.0

7. CONCLUSIONS AND FUTURE WORK

In this paper, we develop a new information-sharing algorithm, e-PTC, for distributed resource allocation problems that can generate highly effective adaptive behaviour in dynamic environments and achieve distributed diagnosis of state changes. Empirical evaluation of e-PTC in a simulated call routing application confirms its advantages over the previous state-of-the-art in this area. Specifically, with dynamic state changes caused by network failures, e-PTC achieves up to 20% higher call throughput, up to 3.5 times faster recovery of throughput after failures, and provides a novel algorithm for diagnosing failures without false positives and false negatives. These improvements are earned by e-PTC at the cost of sending more messages than PTC. However, to address this, we plan to investigate ways of limiting the message overhead of e-PTC by using selective communication in which the agents can use a notion of information redundancy to decide when to communicate.

8. REFERENCES

[1] E. H. Durfee and V. R. Lesser. Partial Global Planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1167–1183, 1991.

[2] P. S. Dutta. *Adaptive Distributed Resource Allocation Using Cooperative Information-Sharing*. PhD thesis, University of Southampton, 2005.

[3] P. S. Dutta, N. R. Jennings, and L. Moreau. Cooperative information sharing to improve distributed learning in multi-agent systems. *JAIR*, 24:407–463, 2005.

[4] Floodnet: Pervasive computing in the environment, 2004. http://envisense.org/floodnet/floodnet.htm.

[5] I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaugmann, 2003.

[6] P. J. Gmytrasiewicz and E. H. Durfee. Rational communication in multi-agent environments. *Autonomous Agents and Multi-Agent Systems*, 2001.

[7] C. V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of AAMAS 2003*, pages 137–144, 2003.

[8] M. Lemaitre, G. Verfaillie, H. Fargier, J. Lang, N. Bataille, and J. M. Lachiver. Equitable allocation of earth observing satellites resources. In *Proc. of the 5th ONERA-DLR Aerosapce Symposium*, 2003.

[9] N. Lynch. *Distributed Algorithms*. Morgan Kauffman, 1996.

[10] V. Tamarapalli and S. Srinivasan. Survivability in WDM networks under multiple link failures. In *Proceeding of Optical Communications Systems and Networks*, 2004.

[11] C. J. C. H. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8:279–292, 1992.

Forming Efficient Agent Groups for Completing Complex Tasks

Efrat Manisterski [1], Esther David [2]
[1] Computer Science Department
Bar-Ilan University
Ramat-Gan 52900, Israel
{maniste,sarit}@cs.biu.ac.il

Sarit Kraus [1] and Nicholas R. Jennings [2]
[2] Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK
{ed,nrj}@ecs.soton.ac.uk

ABSTRACT

In this paper we produce complexity and impossibility results and develop algorithms for a task allocation problem that needs to be solved by a group of autonomous agents working together. In particular, each task is assumed to be composed of several subtasks and involves an associated predetermined and known overall payment (set by the task's owner) for its completion. However, the division of this payment among the corresponding contributors is not predefined. Now to accomplish a particular task, all its subtasks need to be allocated to agents with the necessary capabilities and the agents' corresponding costs need to fall within the preset overall task payment. For this scenario, we first provide a cooperative agent system designer with a practical solution that achieves an efficient allocation. However, this solution is not applicable for non-cooperative settings. Consequently, we go on to provide a detailed analysis where we prove that certain design goals cannot be achieved if the agents are self interested. Specifically, we prove that for the general case, no protocol achieving the efficient solution can exist that is individually rational and budget balanced. We show that although efficient protocols may exist in some settings, these will inevitably be setting-specific.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence —*Multiagent systems*

General Terms

Algorithms, Design, Economics

Keywords

Complex Task Allocation, Efficient Allocation

1. INTRODUCTION

Many task allocation problems require a number of autonomous agents to form groups in order to execute complex activities they can't perform individually [16]. In this work, we specifically consider the case in which tasks: (1) are composed of several subtasks and (2) have predetermined and known overall payments, set by the tasks' owners (i.e., a fixed price contract) [6]. To accomplish a particular task in this case, all its relevant subtasks need to be completed by agents with the necessary capabilities, and the agents' costs need to fall within the preset task payment. Therefore, in this scenario, allocated agents are likely to receive payments which are higher than their reported costs, to meet the requirement that the overall task payment need to be fully paid out.

This task allocation problem is common in a wide range of contexts, including both cooperative and non-cooperative systems. By means of illustration, for the former case consider a large organization composed of several departments, each with its own set of capabilities. Such organizations may receive offers to carry out multiple projects with known payments. In order to accomplish the specific goals of each project, several departments need to work together. In such cases, the role of the organization manager is to allocate the projects to the departments and to decide on the division of the complete available payments to the contributing departments, in order to maximize the organization's efficiency. Here it is assumed the departments cooperate by revealing their actual costs and are prepared to execute any given subtask within their capabilities. Similar scenarios also occur in Grid resource allocation problems where a controller faces demands to allocate an appropriate set of computational resources for the completion of a set of tasks within the fixed payments constraints of the tasks' owners. However, in contrast to the first example, here the controller neither owns the computation resources that may be deployed nor does it have full information about them. Therefore, this must be a non-cooperative designation. Each such resource has to explicitly declare which activities it is willing to perform and its corresponding costs. Only with this information can the controller make the allocation.

Given the ubiquity of this problem, it is surprising that it has received no significant attention to date. Specifically, the main work in this area is due to Kraus et al. [7, 6] who consider a non-cooperative agent system for executing such tasks. In particular they provide a protocol that achieves 80% efficiency, but which does not scale to a large number of subtasks and agents. In addition, they benchmark their protocol's efficiency against a solution achieved by a heuristic and not against the the most efficient solution (which makes their results even more fragile). To rectify these shortcomings, in this paper we first provide a cooperative agent system designer with a practical solution that achieves the efficient allocation. We do so by developing

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

an algorithm which is exponential in the number of tasks, but polynomial in the number of agents and the number of subtasks. Since in many real world situations, the number of tasks is relatively small and can be bounded by a small constant number, the algorithm's complexity can be considered polynomial in the size of the input. We also prove that no polynomial algorithm exists when the number of tasks is unbounded (unless P=NP). This is therefore the best result that can be achieved.

Unfortunately, this solution is not applicable for non-cooperative settings, where more difficulties are raised (e.g., uncertainties about agent's cost, self interested nature). Consequently, in the second part of the paper we analyze what can and cannot be achieved when we consider self interested agents. Specifically, for the general problem we prove that no individually rational and budget balanced protocol can exist which achieves an efficient solution. However, there are some non-cooperative settings for which an efficient protocol does exist. The drawback is that such an efficient protocol for one such setting is likely to be inefficient for another. This result can then be used by the designers of agent systems, who are faced with this task allocation scenario, to direct them to develop appropriate protocols and strategies for particular settings of the problem, and to ensure they know what can and cannot be achieved. Moreover, the performance of any proposed protocol for the non-cooperative setting can in the future be benchmarked against our solution for the cooperative case. This will give more accurate results than benchmarking against heuristics as has been done in previous work.

The rest of the paper is organized as follows. Section 2 introduces our problem definition. In section 3 a cooperative setting for the problem is considered for which we develop an algorithm resulting in an efficient solution. Section 4 focuses on the non-cooperative setting where we show what can and cannot be achieved. In section 5 related work is described and, finally, we conclude in section 6.

2. PROBLEM FORMALIZATION

The general description of our task allocation problem assumes there are M candidate tasks $\Im = \{T_1, ..., T_M\}$, where each T_i is composed of several subtasks $\left\{ST_{i_1}, ..., ST_{i_{k_i}}\right\}$. The overall payment of a task, $p(T_i)$, is predefined and known to all and no inter-task payments transfer is allowed. Moreover, by definition, the task payments need to be fully paid out for their completion. However, the partitioning of the task payment among its subtasks is not predetermined, but will be set as part of the task allocation process. Here a task is considered to have been completed if all its subtasks have been executed.

For the completion of these tasks, a set of agents, $A = \{A_1, ..., A_N\}$, is given. Each of these agents has capabilities to perform multiple subtasks (where some overlapping among different agents' capabilities is assumed). Here, we assume these capabilities to be known to all participating agents and the controller. However, as the subtasks should be carried out simultaneously, we assume that each agent is capable of performing only one subtask at a time.[1] Note that given the dependency on the resource constraints and the reward structures, not all of the tasks may get executed.

In this paper we consider both cooperative and non-cooperative settings. For both cases we assume a central controller is responsible for the task allocation. In the cooperative case we assume the controller has full information about the agents' costs (as the capabilities are known to all) either because of the system's characteristics or because the agents are assumed to report their costs truthfully (since they have no incentive to manipulate this information). In addition, here an agent's cooperation also signals its agreement to complete any assigned subtask which it is capable of performing. Now in the non-cooperative setting, the agents' capabilities are still known to all including the controller, but in this case the controller can only assign subtasks to those agents that indicated their willingness to perform them. Therefore, an agent may strategize by choosing which of the subtasks it is willing to undertake and by manipulating their corresponding declared costs. Note that in doing so agents may use the information they have about the other agents' capabilities and their cost distributions as we define below.

Formally, a knowledge capability function (φ) that is known to all agents and the controller is assumed, where $\varphi(A_j, ST_{i_l})$ equals one if agent A_j is capable of performing subtask ST_{i_l} and otherwise zero. Using this function, we summarize each agent's capabilities by $\Psi_j = \{ST_{i_l} | \varphi(A_j, ST_{i_l}) = 1\}$. Conversely, for any subtask ST_{i_l} there is a set of agents capable of completing it, $\Omega_{i_l} = \{A_j | \varphi(A_j, ST_{i_l}) = 1\}$. Moreover, each agent A_j has a private cost for performing a given subtask ST_{i_l}, $c^j_{i_l}$, taken from a common distribution F_{i_l} in the range of $\left[\underline{\theta}_{i_l}, \overline{\theta}_{i_l}\right]$.

An *allocation* Φ is defined as a match between some of the agents and some of the subtasks, $\Phi = \left\{(A_{i_1}, ST_{i_{1_{l_1}}}), ..., (A_{i_n}, ST_{i_{n_{l_n}}})\right\}$. Here Φ_T indicates the set of tasks allocated by Φ ($\Phi_T \subseteq \Phi$) and an allocation Φ is *feasible*, if and only if it satisfies three conditions:

C1 Each agent in Φ is allocated to at most one subtask.

C2 Any task is either fully allocated (one agent per subtask) or it is unallocated.

C3 For each task $T_i \in \Phi_T$, the total cost for the agents to perform its constituent subtasks does not exceed the task's total payment $p(T_i)$.

As discussed in the introduction, we seek to find an efficient allocation. To this end, we define the **value of an allocation** Φ, $V(\Phi)$, to be the difference between the total payments of the allocated tasks and the total cost for the assigned agents to perform the corresponding subtasks, $V(\Phi) = \sum_{T \in \Phi_T} p(T) - \sum_{(A_j, ST_{i_l}) \in \Phi} c^j_{i_l}$. The feasible allocation with the highest value is considered to be the ***efficient allocation*** Φ_{eff}. Consequently, the efficiency of any proposed allocation Φ is measured relative to Φ_{eff} as: $efficiency(\Phi) = \frac{V(\Phi)}{V(\Phi_{eff})}$.

Next we present a formal description of our problem using an integer program. The variables y_i and $x^j_{i_l}$ are decision variables where $y_i = 1$ if task T_i is chosen to be allocated and otherwise zero, and $x^j_{i_l} = 1$ if agent A_j is chosen to perform subtask ST_{i_l} and otherwise zero:

$$\Phi_{eff} = \max_{y_i, x^j_{i_l}} \left(\sum_{i=1}^{M} (y_i \cdot p(T_i)) - \sum_{j=1}^{N} \sum_{ST_{i_l} \in \Psi_j} (x^j_{i_l} \cdot c^j_{i_l})\right) \quad (1)$$

[1]This is for simplicity of exposition and our results can be easily extended to the case where an agent is capable of performing any constant number of subtasks at a time.

such that:

$$\sum_{ST_{i_l} \in \Psi_j} (x_{i_l}^j) \leq 1, \forall j \in 1, ..., N \quad (2)$$

$$\sum_{j \in \Omega_{i_l}} (x_{i_l}^j) = y_i, \forall i \in 1, ..., M, \text{ and } l \in 1, ..., k_i \quad (3)$$

$$x_{i_l}^j, y_i \in \{0, 1\} \forall i \in 1, ..., M, \forall l \in 1, ..., k_i, \forall j \in \Omega_{i_l} \quad (4)$$

where equations 2 and 3 correspond to C1 and C2, respectively. Note that condition C3 is implicitly achieved by the definition of Φ_{eff}.[2]

3. THE COOPERATIVE SETTING

In this section, we develop a practical algorithm that can be used by the system's controller to achieve an efficient solution in a cooperative setting. Specifically, we analyze the problem's complexity, then develop an algorithm to find an efficient solution, and finally explore the algorithm's properties.

3.1 Problem Complexity

The complexity of our task allocation problem is given by the following lemma:

LEMMA 1. *Given the task allocation problem defined in the integer program (1-4), the computational complexity of finding an efficient solution is NP-hard. Moreover, it is hard to approximate within a factor of* $M^{1-\epsilon}$.

PROOF. We show this using a reduction from the winner determination problem in combinatorial auctions. In a winner determination problem, there is a set of n items and a set of M bids $B_j(S_j, p_j)$, where S_j may be any bundle of items and p_j is the price offered for S_j. The goal of winner determination is to find the subset of bids that maximizes the revenue with no conflicts (in the sense that the same item would not be included in more than one bid). Given an instance of the winner determination problem we present the items as the agents in our problem (we define a one to one correspondence, so that any item i is associated with an agent A_i). Each bid $B_j(S_j, p_j)$ is viewed as a task T_j for which its payment is p_j. T_j is composed of $|S_j|$ subtasks, where the agents that are associated with the items in S_j are the only agents that are capable of performing the subtasks in T_j. We assume that an agent can perform any of the subtasks of the task it is associated with. In addition, we assume that the cost of the agents performing any subtask is zero. Given this mapping, it is easy to see that there is a solution to the winner determination problem in combinatorial auctions that will achieve a certain profit P for the auctioneer, if and only if an allocation to our task allocation problem exists in which its value is P. Building on this, [14] proves that no polynomial time algorithm can guarantee a bound $k \leq M^{1-\epsilon}$ for any $\epsilon > 0$ for the winner determination problem and thus based on reduction preserving approximability, our task allocation problem is also hard to approximate within a factor of $M^{1-\epsilon}$. □

Having shown our task allocation problem is NP hard, we now go on to present an algorithm which is exponential solely in the number of tasks, but polynomial in the numbers of subtasks and agents. This is a significant improvement over the brute-force algorithm which exhaustively goes through all possible assignments and searches for the minimum cost one. Such an algorithm is also exponential in the number of subtasks.

[2]Assume, by way of contradiction, that an Φ_{eff} exists that doesn't satisfy C3 (i.e., some tasks exist for which the agent's costs exceed the tasks' payments). Omitting these tasks from the allocation results in a higher value allocation, which is a contradiction.

Agents	ST_{1_1}	ST_{1_2}	ST_{2_1}	ST_{2_2}
A_1	5	-	5	-
A_2	-	5	-	5
A_3	30	-	10	-
A_4	-	30	-	10

Table 1: Costs and capabilities for example 1

3.2 Finding an Efficient Allocation

Our solution is based on the algorithm for finding the minimum weighted perfect matching in a bipartite graph [1]. Before describing the algorithm, we first provide an interpretation of our problem in terms of a bipartite graph $G = (V, E)$. Each of the agents and subtasks are represented by a vertex in V, such that a capability of an agent to perform some subtask is indicated by an edge (in E) between the corresponding vertices. A graph G is considered a **bipartite** graph if its vertices, V, can be partitioned into two groups, V_1 and V_2, such that no edge in E exists between members of the same group. In our case, this partition can be viewed as the agents and the subtasks groups. Given a bipartite graph G, a **perfect matching** $M \subset E$ is defined to be a subset of edges that cover all vertices and for which no common vertex exists for any pair of edges in M. In terms of our problem, a perfect matching is an assignment of agents to subtasks such that all the agents and the subtasks in V are allocated. By representing an agent's cost to complete some subtask as a weight of the edge connecting the two corresponding vertices, we obtain a weighted graph. Thus a **minimum weighted** perfect matching is a perfect matching, such that the sum of its edges' weights is the minimum (which in our case is an efficient solution if it satisfies the feasibility conditions). However, in our problem, some of the tasks may not be completed, as explained in section 2, thus in order to find the efficient solution one must go through all the subsets of tasks as we formally describe below.

Given a set of M candidate tasks $\Im = \{T_1, ..., T_M\}$, we define the power set of $\Im$ to be $\Lambda = \{\lambda | \lambda \subseteq \Im\}$). For each λ in Λ, we construct a bipartite graph $G_\lambda = (V_1, V_2, E)$, where V_1 represents all the subtasks, ST, associated with one of the tasks in λ, and V_2 corresponds to the agents in A (if the number of vertices in V_2 is larger than the number of vertices in V_1, the algorithm extends V_1 to the size of V_2 by adding $|V_2| - |V_1|$ vertices that are connected to all vertices in V_2 with a weight of 0). An edge between ST in V_1 and A_j in V_2 exists in E if A_j is capable of performing ST and the edge's weight is A_j's cost to perform ST. Given this construction, the problem of finding the ***minimum cost assignment*** that allocates all tasks in λ to agents in A is equivalent to the problem of finding the ***minimum weighted perfect matching*** M_λ in G_λ. An additional step of the feasibility test is required to validate the solution. The Algorithm is described in figure 1.

To illustrate the proposed algorithm consider the following example:

EXAMPLE 1. *Assume there are four agents* $\{A_1, A_2, A_3, A_4\}$ *and two tasks, each of which includes two subtasks* $T_1 = \{ST_{1_1}, ST_{1_2}\}$, $T_2 = \{ST_{2_1}, ST_{2_2}\}$, *where the task payments*

1. For each λ in Λ do:
   (a) Build a bipartite graph, $G_\lambda = (V_1, V_2, E)$.
   (b) Find the minimum weighted perfect matching M_λ in G_λ (if no perfect matching exists go to 1).
   (c) Check whether M_λ is a feasible allocation.
   (d) If M_λ is feasible, compute its allocation value $V(M_\lambda)$, otherwise go to 1.
2. Return the most efficient allocation, M_λ, which is a feasible allocation with the highest value $V(M_\lambda)$.

Figure 1: Algorithm for finding an efficient allocation

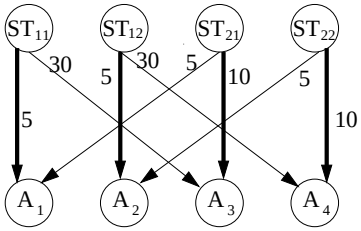


Figure 2: Graph for $\lambda_1 = \{T_1, T_2\}$

are $p(T_1) = 500$ and $p(T_2) = 12$. The agents' costs and capabilities are given in Table 1. The power set of the tasks includes three nonempty sets: $\lambda_1 = \{T_1, T_2\}$, $\lambda_2 = \{T_1\}$ and $\lambda_3 = \{T_2\}$. For each of them, the algorithm constructs the corresponding graph and computes the minimum cost allocation. Figure 2 shows the graph built for $\lambda_1 = \{T_1, T_2\}$.

Here the bold edges indicate the minimum cost allocation generated by the algorithm
$\Phi_{eff} = \{(A_1, ST_{1_1}), (A_2, ST_{1_2}), (A_3, ST_{2_1}), (A_4, ST_{2_2})\}$ for λ_1 whose value is 482. As this allocation is infeasible (since the total cost for T_2 is 20 whilst T_2's payment is 12) the algorithm ignores it and continues by considering λ_2. The efficient allocation of this example is to allocate only task T_1 to agents A_1 and A_2 for which the value is 490 (the highest possible).

According to the algorithm, given a certain subset of tasks, λ, first, a minimum weighted perfect matching is generated. Next its feasibility is verified[3]. Once it fails to satisfy the feasibility requirement, all other possible allocations of λ are ignored. It may seem possible that the efficient solution is not necessarily the minimum perfect matching for some λ, but, instead, is the minimum of all the *feasible weighted perfect matching*. For instance, in example 1, a feasible allocation exists where
$\Phi' = \{(A_1, ST_{2_1}), (A_2, ST_{2_2}), (A_3, ST_{1_1}), (A_4, ST_{1_2})\}$ that might have been the efficient allocation, but the algorithm ignores it. However, as we will prove in section 3.3.2, such a scenario cannot occur.

[3]Note that for each set of tasks $\lambda \in \Lambda$, the algorithm checks for the allocation that includes all tasks in λ, regardless of its feasibility, and computes the minimum allocation if such an allocation exists. However, if we change the algorithm to forbid infeasible allocations, the problem of checking for the *existence* of a *feasible* allocation that allocates all tasks in λ will become NP complete (a reduction can be made from the subset sum problem [2]).

3.3 The Algorithm's Properties

Here, we consider the time complexity and the correctness of our algorithm.

3.3.1 Time Complexity

The complexity of step 1 is exponential in the number of tasks in $\Im$, simply because it goes through all the $\Im$'s subsets. However, the complexity of the internal steps 1a-1d is polynomial, as the complexity of the matching problem is polynomial [1]. Therefore the total complexity of the algorithm is exponential in the number of tasks $|\Im|$ and polynomial in the numbers of subtasks and agents.

3.3.2 Correctness

Here, we prove that if the set of tasks allocated by an efficient allocation Φ is λ, then for G_λ the algorithm will return a feasible perfect minimum matching with the same value as Φ. This implies that the algorithm always returns an efficient allocation. Note that the algorithm can return an allocation different from Φ, but since it has the same value as Φ it is also considered an efficient allocation.

LEMMA 2. *Assume Φ_{eff} is an efficient allocation that allocates the set of tasks in λ_{eff}. Then the algorithm: (i) finds the minimum weighted perfect matching $M_{\lambda_{eff}}$ in graph $G_{\lambda_{eff}}$, (ii) $M_{\lambda_{eff}}$ is a feasible allocation, and (iii) $M_{\lambda_{eff}}$ has the same value as Φ_{eff}'s value.*

PROOF. Assume λ_{eff} is the set of tasks that are allocated by the efficient allocation Φ_{eff}. We begin by proving that a minimum weighted perfect matching $M_{\lambda_{eff}}$ exists for the corresponding graph $G_{\lambda_{eff}}$. As Φ_{eff} is feasible, by definition, it satisfies C1 and C2. That is, Φ_{eff} defines a perfect matching for $G_{\lambda_{eff}}$ which, in turn, means that a *minimum weighted* perfect matching exists. Therefore the algorithm returns $M_{\lambda_{eff}}$ (in the worst case it will be Φ_{eff}). Next, we prove that $M_{\lambda_{eff}}$ is a feasible allocation. Assume, by way of contradiction, that $M_{\lambda_{eff}}$ is an infeasible allocation. By definition $M_{\lambda_{eff}}$ satisfies C1 and C2. This implies that C3 is not satisfied. Let $\lambda_{infeasible}$ be the set of tasks in $M_{\lambda_{eff}}$ that do not satisfy C3 (i.e., $\lambda_{infeasible} = \left\{T_i | T_i \in \lambda_{eff} \text{ and } \sum_{(A_j, ST_{i_l}) \in M_{\lambda_{eff}}} c^j_{i_l} > p(T_i)\right\}$). Now let λ_{new} be the set of tasks that are allocated by $M_{\lambda_{eff}}$ that satisfy C3. That is, $\lambda_{new} = \lambda_{eff} - \lambda_{infeasible}$. As we assume $M_{\lambda_{eff}}$ is an infeasible allocation, $\lambda_{infeasible}$ must not be empty (therefore $\lambda_{eff} \neq \lambda_{new}$). Let $M_{\lambda_{new}} = \left\{(A_j, ST_{i_l}) | (A_j, ST_{i_l}) \in M_{\lambda_{eff}} \text{ and } T_i \in \lambda_{new}\right\}$. $M_{\lambda_{new}}$ is a perfect matching for the set λ_{new}. Consequently, $M_{\lambda_{new}}$ is a feasible allocation for λ_{new} and thus, from the feasibility definition, $M_{\lambda_{new}}$ is a feasible allocation for the entire set of tasks $\Im$. In conclusion, at this point, we have two feasible allocations for $\Im$, Φ_{eff} and $M_{\lambda_{new}}$. Next, we will calculate their efficiency values.

$V(\Phi_{eff}) = \sum_{T \in \lambda_{eff}} p(T) - \sum_{(A_j, ST_{l_i}) \in \Phi_{eff}} c^j_{i_l}$
$\leq$ [4] $\sum_{T \in \lambda_{eff}} P(T) - \sum_{(A_j, ST_{l_i}) \in M_{\lambda_{eff}}} c^j_{i_l} = V(M_{\lambda_{eff}})$
$= \sum_{T \in \lambda_{infeasible}} P(T) - \sum_{(A_j, ST_{i_l}) \in M_{\lambda_{eff}} - M_{\lambda_{new}}} c^j_{i_l} +$
$\sum_{T \in \lambda_{new}} P(T) - \sum_{(A_j, ST_{i_l}) \in M_{\lambda_{new}}} c^j_{i_l}$

[4]As the minimum perfect matching finds the lowest weight matching but not necessarily a feasible one, the efficiency value of $V(M_{\lambda_{eff}})$ is greater or equal to $V(\Phi_{eff})$.

$< \sum_{T \in \lambda_{new}} P(T) - \sum_{(A_j, ST_{i_l}) \in M_{\lambda_{new}}} c^j_{i_l} = V(M_{new})$.

To conclude, we prove that $V(\Phi_{eff}) < V(M_{new})$, which contradicts the fact that Φ_{eff} is the allocation with the maximum efficiency value. Therefore the efficiency value of $V(M_{\lambda_{eff}})$ is equal to $V(\Phi_{eff})$. Previously we proved that $M_{\lambda_{eff}}$ is a feasible allocation. That is, the value $V(M_{\lambda_{eff}})$ cannot be greater than $V(\Phi_{eff})$ since Φ_{eff} is *the* efficient allocation. On the other hand, $V(M_{\lambda_{eff}})$ cannot be less than $V(\Phi_{eff})$ as both allocate all tasks in λ_{eff} and $M_{\lambda_{eff}}$ is the minimum matching cost . □

Up to this point, we have dealt with a cooperative setting. Unfortunately, however, the algorithm developed for this setting cannot be simply applied to the non-cooperative case for the following reasons: (1) the controller in a non-cooperative setting cannot assign a subtask to an agent that has not indicated its willingness to take on the activity and (2) the agents in a non-cooperative case may manipulate their reported costs in order to increase their individual benefit. Moreover, while in a cooperative setting the division of a task's payment does not have any impact on the allocation outcome, in the non-cooperative case the payment division may well play an important role (as it directly affects the agents' behavior). Therefore, as part of any proposed protocol, a rule by which the task's payment is divided among the contributing agents needs to be designed. As a consequence, no guarantees about the proposed algorithm's efficiency exist when applied to the non-cooperative settings. Thus next we examine the non-cooperative setting in terms of desiderata, impossibility results, and potential solutions.

4. THE NON-COOPERATIVE SETTING

In a non-cooperative system, agents have the freedom to decide which subtasks they wish to carry out and which costs to report (since the controller has no information on the agents' actual costs). Thus the controller can only allocate the subtasks based on the agents' requests. Therefore, in order to achieve an efficient solution, the simple approach of asking agents to report their private information (as assumed by the algorithm for the cooperative case) needs to be replaced by a more sophisticated protocol. That protocol should still seek to maximize the overall efficiency, while taking into consideration that each individual agent may strategise (on the subset of its capabilities and their costs to report) in order to maximize its own utility.

In most task allocation scenarios, the strategy space for a buyer considers values that are lower than its actual valuation. Similarly, the strategy space for a seller considers values that are higher than its cost. In our case, the agents are selling their services to perform some subtasks. Given this, one might think that as part of our agents' strategies in the task allocation problem we need to consider only those costs that are higher than their actual costs. In contrast, however, we show that this is not the case in our setting and an agent designer needs to consider values that are both higher and lower than the actual agents' costs[5]. Consequently, given that in non-cooperative settings the agents may make strategy-based declarations, we would like to develop protocols for which dominant strategies (i.e., the best strategy for an agent regardless of the strategies chosen by other players), or at least Bayesian Nash equilibria [11] exist to free agents from the need to strategize.

In particular, we would like to find one general protocol for which agents will have equilibrium strategies (for the strongest concept possible) that achieves the ***efficient*** solution for all instances of our problem and, in addition, will satisfy the following desirable properties. First, as we have self interested agents, a primary desiderata of any proposed protocol is to be ***individually rational*** (i.e., the expected revenue for each agent from participating should be higher than or equal to the revenue achieved by not participating). This is essential to guarantee that no agent loses by participating in the allocation process. Second, the protocol should also be ***budget balanced*** to match the definition of our task allocation problem (i.e., any task's payment should be fully paid out to its contributing agents and no transfer is allowed between tasks). The motivation for such a requirement is the fact that the preset overall payment is known and that it has been allocated solely for the completion of the task. Third, we would like our protocol to be ***undiscriminating*** (i.e. it should not discriminate among the agents by utilizing the knowledge of the agents' capabilities. In this model the protocol is required to give the same expected payment to any two agents with the same declared capabilities and the same costs, regardless of the difference in their actual capabilities). Such a requirement is vital to ensure that the maximum number of agents are willing to engage in the pool of providers.

For the analysis that follows, we consider two classes of environment: one in which the agents may strategize about their costs and another in which they cannot. We term the latter a ***real cost environment*** and we consider it a means of factoring out any issues that follow from the information uncertainty, rather than the self-interested nature of the agents.

4.1 No Efficient, Individually Rational, and Budget Balanced Mechanism Exists

Here we prove that no protocol can exist that is efficient, individually rational and budget balanced, for which there is a dominant strategy or at least a Bayesian Nash equilibrium.

To prove this, we use a reduction to the bilateral trading problem. In the bilateral trading problem, a single buyer wants to buy a single item from a single seller. Both the seller and the buyer have private values for the item v_s and v_b, respectively. Both values are independently drawn from the same uniform distribution between 0 and 1. In such problems there are cases in which trading is possible ($v_s \leq v_b$), but due to the buyer's and seller's strategies, trading might not be achieved. Myerson and Satterthwaite prove that for this problem no efficient protocol exists which is individually rational and budget balanced even in a Bayesian Nash equilibrium [10].

LEMMA 3. *Given the general task allocation problem defined in the integer program (1-4) even for only one task $|\Im| = 1$, no efficient protocol exists that is individually ra-*

[5]As an illustration, consider the task allocation described in Example 1 with a protocol, that given the agents' declaration, calculates the efficient solution (using the algorithm presented in figure 1) and distributes the surplus of each task equally among its contributing agents. In this case, A_3 is incentivised to declare a cost lower than the real one for subtask ST_{1_1} (for which $p(T_1) = 500$); by doing this it increases its chance to win this subtask.

tional, and budget balanced, even in a Bayesian Nash equilibrium.

PROOF. Assume, by way of contradiction, that there is an efficient, budget balanced and individually rational protocol PR_{task} for our task allocation problem. We can then use PR_{task} in order to design a protocol $PR_{BiTrading}$ that solves the bilateral trading problem.

Given an instance of the bilateral trading problem, the protocol $PR_{BiTrading}$ is defined as follows:

- Both the seller and the buyer bid their values for the item b_s and b_b, to the controller, respectively.
- The controller builds the following instance I_{task} of our task allocation problem that considers two agents, $\{A_1, A_2\}$, and one task including two subtasks, $T_1 = \{ST_{1_1}, ST_{1_2}\}$ where $p(T_1) = 1$. The agents' costs for both subtasks are uniformly distributed over $[0, 1]$. Agent A_1 is capable of performing only subtask ST_{1_1} and its cost is b_s. Agent A_2 is capable of performing only subtask ST_{1_2} and its cost is $1 - b_b$.
- Run PR_{task} to solve the instance of our task allocation problem I_{task}.
- If T_1 is allocated and the payments p_1 and p_2 are made to A_1 and A_2 respectively, (since the fixed payment constraint is $p_1 = 1 - p_2$) the seller sells the item to the buyer and the buyer pays p_1 to the seller, otherwise no trade occurs.

Since we assume that PR_{task} is efficient, budget balanced and individually rational, $PR_{BiTrading}$ is as well. This is a contradiction to Myerson and Satterthwaite's work which proved that there is no efficient, individually rational and budget balanced protocol that exists for the bilateral trading problem which is in Bayesian Nash equilibrium [10]. □

Hence, as the property of being strategy proof (i.e. truth telling is a dominant strategy) is a strictly stronger concept than the Bayesian Nash equilibrium, from Lemma 3 we can conclude that no efficient, individually rational and budget balanced protocol which is strategy proof exists for our general problem.

In the next section, we consider real cost environments, removing the agents' ability to falsely report their costs for their own benefit and demonstrate that there is still no efficient protocol meeting our requirements.

4.2 No Efficient Protocol Exists for Real Cost Environments

There are some situations in which the real costs of the agents may be verified. One example is a scenario where service providers do not directly receive payment for the costs of their raw materials, but payments are given against original receipts. Notice that in this case, in addition to the cost reimbursement, the agents expect a supplementary payment from the surplus of the task overall payments, and this is their motivation for participating in the allocation.

In such cases an agent's strategy reduces to one dimension: its willingness to perform certain subtasks (it no longer incorporates the subtask cost dimension). Therefore, a truth telling agent in this case is one that declares all the subtasks it is able to perform. If in this scenario an agent is forced to declare all the subtasks it can perform, we are reduced to the problem which is solved in section 2. However, we wish to consider the case where the agent is free to choose the subtasks it declares.

Agents	ST_{1_1}	ST_{1_2}	ST_{2_1}	ST_{2_2}
A_1	1	1	-	-
A_2	1	1	-	-
A_3	1	1	2	-
A_4	-	-	-	3

Table 2: Agents' capabilities and costs of example 4.2

Against this background, we will now go on to prove that even for real cost environments, no protocol achieving the efficient solution can exist that is individually rational and budget balanced (except where the allocation problem consists of a single task).

LEMMA 4. *Given the general task allocation problem defined in the integer program (1-4) and assuming there are at least two tasks, no protocol achieving the efficient solution can exist for real cost environments that is individually rational and budget balanced, not even in a Bayesian Nash equilibrium.*[6]

PROOF. We demonstrate that there is no general protocol by exhibiting a particular setting for which no protocol can exist. Assume there are four agents $\{A_1, A_2, A_3, A_4\}$ and two tasks, each of which includes two subtasks $T_1 = \{ST_{1_1}, ST_{1_2}\}$, $T_2 = \{ST_{2_1}, ST_{2_2}\}$, where the task payments are $p(T_1) = 100$ and $p(T_2) = 8$. The costs of subtasks ST_{1_1} and ST_{1_2} are 1 for all agents, and the costs for ST_{2_1} and ST_{2_2} are uniformly taken from [1,4]. The specific agents' costs and capabilities are given in table 2.

By the revelation principle[11] it is sufficient to prove that no efficient protocol exists where an agent's best strategy is to agree to perform all the subtasks it is able to perform and declare its real costs assuming all other agents will do the same. If agent $A3$ declares its willingness to perform all the subtasks it is capable of performing $\{ST_{1_1}, ST_{1_2}, ST_{2_1}\}$, it will be assigned to ST_{2_1} and its utility will be less than 8 (as any payment can not exceed the task payment). However, if A_3 agrees to perform only $\{ST_{1_1}, ST_{1_2}\}$, it becomes similar to A_1 and A_2 [7]. Consequently, its expected utility is $\frac{98}{3}$ (as the expected payment for agents with identical costs and the same declared subtasks should be the same). Therefore, for every protocol, A_3 won't declare subtask ST_{2_1}, resulting

[6]Note that this lemma cannot be derived from lemma 3 since here we assume a real cost environment (whilst in lemma 3 the agents can lie about their costs). On the other hand, lemma 3 cannot also be derived from this lemma (since there we proved that no desired solution can be achieved for one task and, for real cost environments a desired solution does exist for one task).

[7]Note that since A_3 knows the other agents' capabilities and subtask's costs distribution it knows both A_1 and A_2's costs and capabilities (the costs of ST_{1_1} and ST_{1_2} are equal to 1 for all agents). By assuming that all other agents declare all their subtasks, A_3 knows that by agreeing to perform only subtasks $\{ST_{1_1}, ST_{1_2}\}$ and declaring the costs of 1, it agrees to perform the same subtasks and declare the same costs as A_1 and A_2. Therefore the protocol should yield A_3 the same expected payment.

in an inefficient allocation as only one of the tasks can be allocated. □

However, in contrast to our results in section 4.1, next we prove that in the case where there is only one task to allocate, an efficient protocol exists for real cost environments.

LEMMA 5. *Given a real cost environment, an efficient protocol exists for the task allocation problem defined in the integer program (1-4) where* ***there is only one task****.*

PROOF. Consider a protocol in which each agent is requested to declare the subtasks it is interested in and its costs for them. Based on this information, the controller calculates the efficient allocation using the perfect minimum weighted matching algorithm (section 3). According to this efficient allocation, the protocol pays the cost of each allocated agent and an extra equal share of the surplus from the task payment. Given this protocol and the fact that there is only one task to allocate, it is easy to see that a dominant strategy exists where each agent is motivated to declare all the subtasks it is able to perform with their real costs (as we assume the real cost environment). By assuming that all agents use this dominant strategy, we can deduce that this protocol inevitably derives the real efficient solution as it does in the full information case. □

So far, we have shown that no single efficient protocol exists for the general case that achieves the desirable properties. However, there are settings for which such efficient protocols exist. The next section explores these settings.

4.3 Only Setting-Specific Efficient Protocols Exist

In order to show that efficient solutions can be generated for specific settings, but the ensuing solutions are specific to these settings, we consider two reasonably common environments. Specifically, we consider a Different Capabilities Setting (DCS) in which the agents have different capabilities and all the tasks are associated with the same overall payment, and a Different Task Payment Setting (DTPS) in which all agents are capable of performing all the subtasks and the tasks are associated with different payments. Then, for each setting, we provide an efficient protocol, and prove that this protocol is not efficient for the other one. In both settings we consider that the number of subtasks is equal to the number of agents and that each task includes K subtasks (the same number for all the tasks). Finally, we assume the agents' cost for any subtask is equal to c.

Given these settings, we consider two protocols: (1) the sequential protocol ***SEQP*** and (2) the simultaneous protocol ***SIMP***. According to both protocols, the task's payments are divided equally among the contributing agents. Then as it is assumed that each agent's cost to perform any of the candidate subtasks is c, each agent has to declare which subtasks it wants to perform but not their costs.

In detail, Protocol **SIMP** allocates all tasks simultaneously. Therefore, each agent has to declare at once all the subtasks (of all the tasks) it wants to perform. Based on these declarations, the protocol finds the most efficient allocation (using the algorithm described in figure 1). In contrast, the **SEQP** protocol allocates the tasks sequentially according to a decreasing order of the tasks' payments (a random order is used to break ties). In each round the controller announces a task, T_i, and the agents declare which of its subtasks they wish to perform. Based on these declarations, in each round, the SEQP protocol finds the most efficient allocation for the considered task. Next we show that for each setting, only one of the protocols achieves an efficient solution.

We begin by considering the efficient protocol for the DCS setting. In this case all subtasks' payments are equal; thus any agent is indifferent to which subtask it is allocated. Accordingly, under the SIMP protocol, a rational agent will declare all the subtasks it can perform. Moreover, as the SIMP protocol searches for the overall efficient allocation, based on the reported capabilities, it achieves the same result as in the cooperative system case. However, using SEQP in this setting does not achieve an efficient solution. This is because the allocation is made separately for each task in a greedy manner (i.e. finding the most efficient allocation for a given task at a time while ignoring the rest of the tasks). For example, consider the case where two agents, A_1 and A_2, are capable of performing subtask ST of task T, while A_1 is the only agent capable of performing subtask $ST^{'}$ of task $T^{'}$. For this case, if the allocation is made simultaneously for all tasks, A_1 will be allocated to $ST^{'}$ and A_2 will be allocated to ST so that both tasks may be completed. However, for SEQP protocol, T can be allocated first (as a random order is used to break ties). Therefore in this case, A_1 may be chosen to perform ST and consequently, $T^{'}$ won't be allocated.

For the DTPS setting, SEQP achieves an efficient solution. This is because when it allocates a certain task, it is the currently most attractive one (as all tasks with higher payments have already been allocated). Thus, each agent is motivated to declare all subtasks of the announced tasks. As all agents are capable of performing any of the tasks, the concern that some particular agent will be needed for a later allocation does not exist. By contrast, in this case SIMP does not achieve an efficient solution. This is because there may be a case where some subtasks may not be declared by any agent. To understand why SIMP fails in this setting, consider the case of an agent that declares all the subtasks. This agent might be assigned to the lowest payment subtask, while the highest payment one is allocated to another agent who declared only the subtasks with the highest payments.

In conclusion, we have demonstrated that in the non-cooperative case, there is no general protocol which fulfils all our desiderata. Moreover we have proved that there exist settings for which protocols can be developed. However, each such settings requires individual treatment and a tailored protocol. Consequently, any further attempt to develop a protocol for this task allocation problem in the non-cooperative setting, will have to commence by clearly defining the setting in which it is to operate.

5. RELATED WORK

Task allocation and coalition formation in multi-agent environments have been widely studied in recent years. However, to date most of the works including classical solutions for coalition formation (such as Shaply value and Kernal [5, 4]) assume complete information about the agents is available and that they are self interested. Therefore these solutions are not applicable to our cooperative and non-cooperative settings, where, respectively, cooperative agents and incomplete information are assumed.

However, the coalition formation work by Li and Zhang does consider a cooperative setting for a task allocation problem that is similar to ours [8]. However, the authors assume that the agents may use all their resources simultaneously (i.e using one resource type doesn't prevent the agent from using its other available resource types). This contradicts our assumption that agents have finite capabilities and can perform only a constant number of subtasks simultaneously. Thus their results cannot be applied to our problem. Other works on coalition formation in a cooperative setting are [15, 3] in which the authors looked for the optimal coalition structure. However, in their works they considered environments in which only one value is associated with each coalition. This contrasts with our case, where a coalition may have different values for each task it can perform depending on the subtask allocation.

In our algorithm for the cooperative setting we use matching in order to find an efficient solution, similar to other works like [13] that use matching in order to solve various allocation and assignment problems. However none of these works have the same constraints as we have (e.g., constraints C1-C3 in section 2) . Therefore their solutions are inapplicable to our case. The only work that considers the same problem constraints as ours is [6, 7]. However, as explained in section 1, they use heuristics (e.g., hill climbing), while we find the exact efficient solution.

Another related research area is the cost sharing mechanism problem [9]. According to this work, incomplete information, and, typically, a fixed known cost per coalition (similar to our task's payment) is assumed. But, in contrast to our work, they consider a much simpler structure of the problem. Specifically, the allocator is responsible for forming only one coalition, while in our case the allocator faces the more complex decision as of which group of agents to assign a certain task to and how to allocate the subtasks to its coalition members.

Our problem also has similarities to a combinatorial market problem [12]. Each subtask can be treated as an item, each task as a bundle of complementary items, and each agent as a seller with substitutable valuations, since an agent can perform only one sub-task simultaneously (see section 2). However, in the combinatorial market the payment for each bundle is determined by the auction. By contrast, in our problem the payment for each task is predefined and known. In addition, in most work on combinatorial markets the surplus of the overall buyers' values and the overall sellers' values is divided among all participants. This is not appropriate in our model, where the task payment is divided only among the agents that perform the task.

6. CONCLUSIONS

In this paper we have considered a complex task allocation problem that needs to be solved by a group of autonomous agents working together. Given the ubiquity of this problem, it is surprising that to date it has not received significant attention. To rectify this omission, we consider both cooperative and non-cooperative settings of the problem. For the former setting we provide a practical algorithm for finding an efficient solution. The time complexity of this algorithm is polynomial in case the number of tasks is bounded by a small constant number (which is often the case in real world situations). In contrast, for the non-cooperative setting we prove that no general protocol achieving the efficient solution exists for our problem that satisfies our desired properties. However, we show that there exist some non-cooperative settings scenarios for which an efficient protocol does exist. Unfortunately, each such setting requires the design of a bespoke protocol, since the solution for one setting is likely to be inappropriate for another. In so doing, this study hopes to assist system designers confronting this problem to understand what can and cannot be achieved.

7. ACKNOWLEDGMENTS

This research was partially funded by the DIF-DTC project (8.6) on Agent-Based Control. This work was also supported in part by NSF No. IIS0208608 and ISF 1211. Kraus is also aliated with UMIACS. In memory of Matat Rosenfeld-Adler.

8. REFERENCES

[1] D. Avis and C. Lai. The probabilistic analysis of a heuristic for the assignment problem. *SIAM J. Comput.*, 17(4):732–741, 1988.

[2] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[3] V. Dang and N. Jennings. Generating coalition structures with finite bound from the optimal guarantees. In *AAMAS'04*, pages 564–571, 2004.

[4] J. Kahan and A. Rapoport. *Theories of coalition formation*. Lawrence Erlbaum Associates, 1984.

[5] S. Kraus and O. Shehory. Feasible formation of coalitions among autonomous agents in non-super additive environments. *Computational Intelligence*, 15(3):218–251, 1999.

[6] S. Kraus, O. Shehory, and G. Taase. Coalition formation with uncertain heterogeneous information. In *AAMAS03*, pages 1–8, 2003.

[7] S. Kraus, O. Shehory, and G. Taase. The advantages of compromising in coalition formation with incomplete information. In *AAMAS04*, pages 588–595, 2004.

[8] H. Lau and L. Zhang. Task allocation via multi-agent coalition formation: taxonomy, algorithms and complexity. In *ICTAI'03*, pages 346–350, 2003.

[9] H. Moulin and S. Shenker. Strategyproof sharing of submodular costs:budget balance versus efficiency. *Economic Theory*, 18(3):511–533, 2001.

[10] R. B. Myerson and M. A. Satterthwaite. Efficient mechanisms for bilateral trading. *Economic Theory*, 29(3):265–281, 1983.

[11] D. Parkes. Mechanism design chapter 2 in phd dissertation. 2001.

[12] A. Pekec and M. H. Rothkopf. Combinatorial auction design. *Management Science*, 49(11):1485–1503, 2003.

[13] M. Penn and M. Tennenholtz. Constrained multi-object auctions and b-matching. *Information Processing Letters*, 75(1–2):29–34, July 2000.

[14] T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.

[15] S. Sen and P. S. Dutta. Searching for optimal coalition structures. In *ICMAS00*, pages 287–292, 2000.

[16] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, 1998.

Gradient Field-Based Task Assignment in an AGV Transportation System

Danny Weyns
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven, Belgium
danny@cs.kuleuven.be

Nelis Boucké*
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven, Belgium
nelis@cs.kuleuven.be

Tom Holvoet
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven, Belgium
tom@cs.kuleuven.be

ABSTRACT

Assigning tasks to agents is complex, especially in highly dynamic environments. Typical protocol-based approaches for task assignment such as Contract Net have proven their value, however, they may not be flexible enough to cope with continuously changing circumstances. In this paper we study and validate the feasibility of a field-based approach for task assignment in a complex problem domain.

In particular, we apply the field-based approach for task assignment in an AGV transportation system. In this approach, transports emit fields into the environment that attract idle AGVs. To avoid multiple AGVs driving towards the same transport, AGVs emit repulsive fields. AGVs combine received fields and follow the gradient of the combined fields, that guide them towards pick locations of transports. The AGVs continuously reconsider the situation of the environment and task assignment is delayed until the load is picked, improves the flexibility of the system.

Extensive experiments indicate that the field-based approach outperforms the standard Contract Net approach on various performance measures, such as the average wait time of transports and throughput. Limitations of the field-based approach are an unequal distribution of wait times across different transports and a small increase of bandwidth occupation.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Design

1. INTRODUCTION

Multiagent systems (MASs) are generally considered as an approach to build complex software systems. Adopting an agent-oriented approach means decomposing a system into multiple, autonomous entities that can act and interact in a shared environment to achieve the system goals. A MAS provides a decentralized architectural solution to a software problem that aims for qualities such as robustness and flexibility.

The application that is central in this paper is an Automatic Guided Vehicle (AGV) transportation system. An AGV system is a fully automated industrial transport system that makes use of multiple AGVs that have to transport loads in an industrial environment, e.g. a warehouse. One way to control an AGV system is through a central server. This server maintains all relevant information, needed to decide which AGV has to do what and which route is to be taken. In a joint research project between AgentWise taskforce and Egemin, a producer of automated logistic systems, we investigate the feasibility of applying the MAS paradigm to an AGV transportation system [5, 21]. In this approach, the control is distributed over the different AGVs, who cooperate to achieve the goals of the system. One major challenge in an AGV transportation system is task assignment, i.e. the assignment of transports to AGVs. *Transports*, which consist of picking up a *load* on a location (i.e. the pick location) and transporting it to another location (the drop location), are generated by an external system. Transports are dynamically created when needed and the transport profile (i.e. the order and pace in which transports are created) is irregular and unpredictable. Due to the highly dynamic nature of transport creation, the assignment of transports to AGVs is complex. Egemin currently applies a policy that pre-stages the AGVs based on a fixed set of rules. This approach however, lacks flexibility to cope with the continuously changing circumstances in the environment.

Typical protocol-based approaches for task assignment such as Contract Net have proven their value, however, they may not be flexible enough to cope with continuous dynamics in the environment. The main goal of the work reported here is to examine the *feasibility* of using a *field-based approach* to achieve *task assignment* in a complex domain, in casu an AGV transportation system. We aim to investigate the advantages as well as the limitations of the approach in this real world setting. In the field-based task assignment for AGVs, transports emit fields that attract idle AGVs, while competing AGVs emit repulsive fields. Each idle AGV follows the gradient of the combined fields it receives, which guide the AGV towards the pick location of a transport. To validate the field-based approach, we implemented it in an AGV simulator [1]. In the simulations, we use data of a realistic AGV system, including a real-world map and transport profiles. We have performed various performance measures, such as the average time a transport has to wait before it is executed, the throughput of the system, and the number of messages that are transmitted. The test results are compared with a reference algorithm for task assignment that uses a Contract Net protocol [19]. In this paper, we give an overview of the field-based approach for task assignment, and we discuss the main

*Supported by Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

results obtained from this study. A detailed description of additional tests not included in this paper can be consulted in [17].

Overview. The remainder of this paper is structured as follows. In Section 2 we briefly introduce the AGV application. Section 3 discusses related work on task assignment and field-based techniques applied in MAS. In Section 4 we describe the approach of field-based task assignment. Section 5 discusses test results and evaluates the approach. Finally, we draw conclusions in Section 6.

2. THE AGV APPLICATION

An AGV system is a fully automated industrial transport system, that uses multiple AGVs to transport loads (raw materials, finished products, etc.) in an industrial environment such as a warehouse. In this section we give a brief overview of the AGV application. Subsequently the map of the environment, the transports, and the AGVs are described.

2.1 Map of the Environment

The AGVs are restricted to follow predefined paths in the environment, on which they navigate by using sensors and stationary beacons. These paths are defined in a *map* of the environment. A map is logical representation of the physical layout on the factory floor, it consists of a network of stations and connecting segments that are accessible by the AGVs. A station is a predefined point on the map, stations can hold special locations, such as a pick or drop location for transports, a parking place for AGVs, or a place where AGVs can reload their battery. A segment defines in which direction(s) AGVs can drive. Additional information may be associated to segments, e.g. speed limits, mutexes that indicate that only one AGV can enter a particulate zone at a time, etc. In section 5 we give an example of a realistic map that we have used for testing.

2.2 Transports

The goal of the AGV system is to let AGVs transport loads. A transport is created by an external system, typically a warehouse management system. A transport represents an order to move a load from a pick location to a certain drop location. Each transport has a priority from 1 to 10, with 10 being the most important. A transport's life-cycle starts when it enters the system. The transport is started when an AGV picks up the load and is terminated when that AGV drops the load at the drop location. The priority of a transport may increase over time, e.g. as a result of long waiting times.

2.3 AGVs

An AGV is a computer controlled, unmanned vehicle that is capable of transporting loads in an industrial environment. An AGV is capable of picking up loads, driving them around and dropping them. AGVs can communicate by means of a wireless LAN. An AGV navigates by storing an internal map of the environment and using internal and external sensors, in combination with stationary beacons. AGVs are equipped with low-level control software that handles the physical interaction with the environment such as staying on track, turning, or determining the current position.

To allow smooth and save movement, an AGV has to make routing decisions over a certain distance in advance. This distance, called the *look-ahead*, is measured according to the logical topology of the environment map. The *look-ahead buffer* associated with an AGV is an ordered set of stations and segments that holds the routing decisions, representing the path that the AGV will follow with certainty, unless something catastrophical (such as a crash) happens.

In this paper we focus on achieving *task assignment* by applying a field-based approach. During task assignment the AGV will use a simple A* algorithm [8] to *rout* towards a transport. Besides task assignment, other important concerns are collision avoidance and deadlock prevention, however, these concerns are not further discussed.

3. RELATED WORK

In this section, we discuss a number of approaches related to our field-based task assignment approach. First we discuss a number of general approaches for task assignment. Task assignment is an extensive domain of research. Here we mainly focus on well-known Contract Net-based mechanisms. Next, we discuss a number of related field-based techniques for MAS described in literature.

3.1 Task Assignment

Efficiently assigning tasks to individual entities in a decentralized system is a common, but certainly not a simple problem [7]. A classic way to achieve task assignment is through auction-based mechanisms, e.g. the widely known and extensively used *Contract Net* protocol (CNET [19]). CNET is based on the notion of call for bids on markets, where managers request for bids and bidders bid to perform the task. In a first step, the manager sends a description of the task to perform to all the bidders. In a second step, the bidders draw up a proposal based on the description of the task and send it to the manager. In a third step, after the manager has received a proposal from all bidders or after a deadline has expired, the manager evaluates the received proposals and assigns the best bidder the task.

In the original CNET protocol, a bidder that has placed a bid for a task (step 2), cannot place a bid for a new task before the first task has ended. In [10], an improvement to CNET is proposed that allows a bidder to place bids for multiple unassigned tasks. This is achieved by giving the bidder the option to refuse the task in step 3 of the original protocol. The manager will then award the contract to the second best bidder, who can again accept it or refuse it, etc.

The CNET protocol can also be extended to allow cascading task assignment, where the receiver of a task proposal starts one or more new CNET protocols to delegate the task or parts of the task to other agents [10]. For example, a system of holonic agents can be used to improve efficiency when dealing with complex tasks that require the collaboration of several agents [11]. A holonic agent consists of several agents but interacts with agents outside the holon as a single agent.

The CNET protocol suffers from low flexibility when dealing with tasks that can not immediately be started upon the allocation (delayed commencement of tasks [3]). Consider for example the CNET protocol applied to task assignment in an AGV system. When a new transport enters the system, a suitable AGV is selected through the CNET protocol and is assigned to the transport. Suppose that while the assigned AGV is driving towards the pick location of the transport, a new transport appears that is more suitable to the AGV (e.g. it is closer or has a higher priority). The AGV will not be able to execute the second transport, since it is already assigned to the original transport. Thus, the CNET protocol achieves an *immediate* and *permanent* assignment of tasks, making it impossible for the system to react flexibly and adaptively when dealing with tasks with delayed start in a highly dynamic environment. [2] provides more flexibility by assigning tasks in two phases (first phase temporarily assignment, next phase definitive assignment) but still does not provide full flexibility in case of delayed commencement.

Another interesting approach for task allocation is described in [6] that is based on the exchange of *tokens* among agents. The authors describe an example of robots that drive around and perceive objects in the environment. These objects have to transferred from one location to another. To move an object, two robots have to col-

laborate. Therefore, two roles are associated with the task: a helper role and a collector role. The coordination among agents is based on the exchange of tokens, which are associated with the roles of a task. Tokens can be exchanged between agents and whoever owns a token can participate in the task in the associated role.

In earlier work [3], we have described an approach for flexible and decentralized task allocation that deals with the problems mentioned above. This approach consists of a continuous *negotiation* protocol where the situation and the allocation of tasks is continuously *reconsidered* until the task is actually started. The approach presented in this paper builds upon the principles developed in that research. However, here we provide a simplified and uniform model that consistently uses fields for task assignment.

3.2 Field-Based Techniques in MASs

Techniques based on *fields* have proven to be valuable in various applications that involve movement of agents in a metric space [7, 15, 12, 16, 13, 14]. In this approach, objects in the environment produce fields, which are propagated in the environment in a certain range. At each position in the metric space these fields have a certain value, forming *potential fields*. These values are typically proportional to the distance from the source of the field. Agents perceive the fields and combine them in a certain way. The behavior of an agent then simply consists of following the combined potential field downhill, that is, agents follow the direction of the *gradient* of the field. As an example, suppose an agent has to move to a goal while avoiding obstacles in the environment. The obstacles emit repulsive fields, while the goal emits an attractive field. The agent combines the fields by adding the values of the attractive and repulsive field. The combined field can be visualized as a landscape, with the goal representing a valley and the obstacles mountains. This combined field is then followed downhill, which can be viewed as a ball rolling down the slopes of the landscape towards the goal. The field-based approach readily extends to task allocation, where each task emits an attractive field and agents follow the strongest field. Competition among agents can be handled by letting the agents emit repulsive fields.

Field-based techniques have also been applied in systems that are inspired by biological phenomena and physics. For example, birds in a flock all fly in the same direction and avoid collision by keeping a fixed distance between each other. An example application for this idea are guards who have to patrol a museum, keeping a certain distance between each other to cover more ground [15]. Shehory et al. [18] describes a physics oriented model for cooperative goal-satisfaction with simple agents in a large-scale cooperative MAS. Different kinds of gradient field approaches have also been used in the context of RoboCup, a recent example is [4].

[12] discusses a variation on the field-based approach where agents construct a field in their direct neighborhood to achieve routing and deadlock avoidance in a simplified AGV systems. Another variation is described in [16], where the MMASS (Multilayered Multi Agent Situated System) model for multiagent coordination is used. In this model, the environment is represented as a multi-layered graph in which the agents can spread abstract fields. In the standard field-based approach, agents combine perceived fields and are constantly guided by the fields, while in MMASS fields are in principle considered independent from each other and are exploited only to trigger one shot reactions.

Pros versus Cons. The field-based approach is an elegant and effective way to tackle problems in a metric environment. The agents can be kept simple, relying on the environment to achieve the system objective. Since no rigorous protocols are used, robustness and openness (agents can easily be added or removed from the system) are achieved. The approach also benefits from adaptivity: the landscape of fields is readily adjusted to changes in the environment. Also, the fields can be interpreted and combined in different ways by different agents (e.g. a location may be a goal for one agent, while another agent needs to avoid it).

However, the field-based approach also suffers from some limitations. Firstly, the approach is difficult to apply when the situation cannot be represented as a metric space. Secondly, local minima can arise, agents can become stuck at a local minimum, leading to sub-optimal solutions. Another problem is that the approach is not easily to combine with cognitive decision making mechanisms. Finally, a disciplined methodology is lacking to predict the overall evolution of the system. With the field-based approach, the solution emerges from the interaction among the agents. Typically, the development of such systems requires intensive parameter tuning.

Maintaining Fields. An important issue when using the field-based approach is the distribution and maintenance of fields within the environment. A possible way to maintain fields is a storage infrastructure to store the different values of fields in different (digitized) positions in the environment. When dealing with a software environment, this requirement is automatically achieved. For example, the field-based approach has been used successfully in games such as in "Quake 3 Arena" [13] to guide the behavior of non-human characters.

When dealing with a physical environment, a distributed storage capability could be physically supported by the environment. For example in [15], the field-based approach is applied in guiding tourists in a museum. The tourists are provided with a software agent running on some wireless handheld device, giving suggestions on how and where to move. A computer network with a topology that mimics the topology of the museum plan is embedded in the museum walls. The hosts in the network are associated to each room and are capable of communicating with each other and the mobile devices in their proximity. Fields can be injected at a host, after which they will be diffused hop-by-hop across the network, modulating the values of the fields as necessary. The latter is achieved by supporting middleware, such a TOTA (Tuples On The Air) [14].

If no environmental storage infrastructure is available, fields can still be realized by broadcasting information messages, which contain the position of the source of the field and its initial strength. Agents receive these messages and calculate the value of the field on their position. In this paper we have applied this approach since a typical AGV system provides no distributed storage infrastructure external to the mobile AGV machines.

4. FIELD-BASED TASK ASSIGNMENT

In this section we explain the approach for field-based task assignment in AGV transportation systems. Similar to the approach in [3], task assignment is *delayed* until the AGV actually reaches the pick location of the transport, allowing the system to react flexibly and adaptively to new transports that enter the system. We start with a high-level explanation of the approach, followed by an overview of the software architecture.

4.1 High-Level Overview

The basic idea of field-based task assignment is to let each idle AGVs follow the gradient of a field that guides it toward a load that has to be transported. There are two types of fields in the system: (1) transports emit fields into the environment that attract idle AGVs; (2) to avoid multiple AGVs driving towards the same transport, AGVs emit repulsive fields. AGVs combine received fields and follow the gradient of the combined fields, that guide them towards pick locations of transports. The AGVs continuously reconsider the situation of the environment and task assignment is delayed

until the load is picked, which benefits the flexibility of the system. To explain the field-based approach for taks assignment, we use the scenario depicted in Fig. 1.

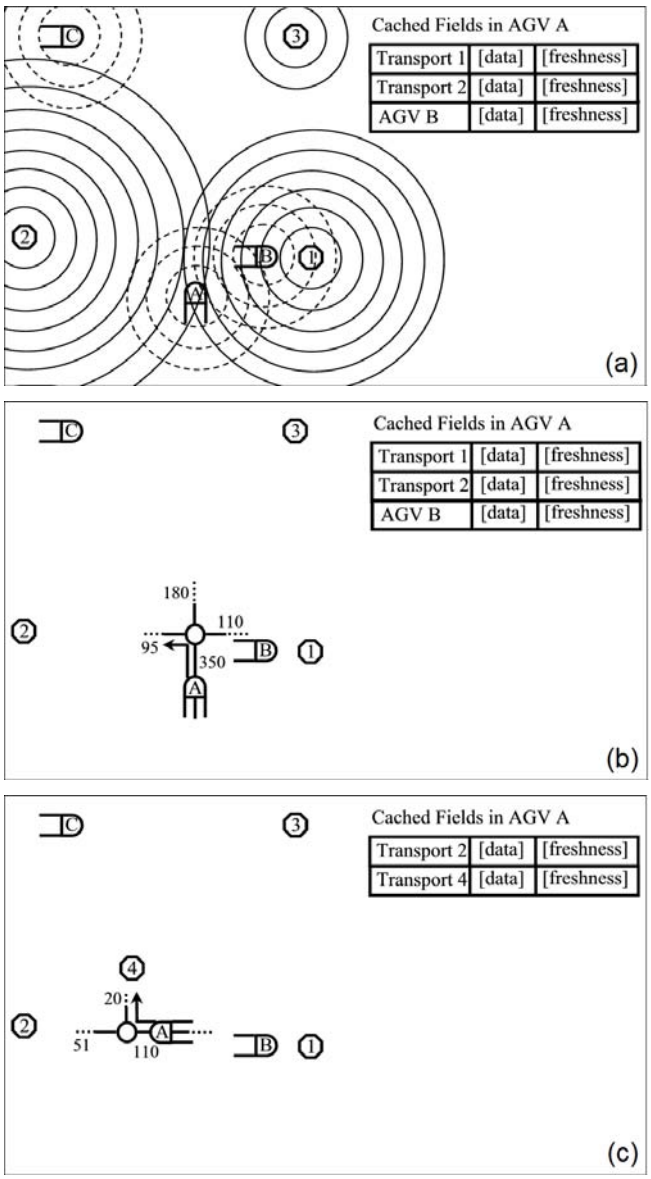


Figure 1: ***Example scenario of the developed approach. The full circles represent transport fields, the dotted circles AGV fields. For clarity, we have not drawn the fields in (b) and (c).***

- **AGV agents and transport agents.** Task assignment is achieved by the interaction between two kinds of agents: *AGV agents* and *transport agents*. The AGV agents correspond to the physical vehicles and the transport agents represent transports. Whenever a new transport enters the system, a corresponding transport agent is created. The lifetime of the transport agents ends when the associated load is delivered. *Conceptually* a transport agent resides at the pick location of the associated transport. On which physical computer system(s) the transport agents reside is not important for the discussion in this paper, the only important issue is that the agents operate independently and therefore can be distributed across several computer systems.

- **Agents emit fields.** Both AGV and transport agents emit fields, called *AGV fields* and *transport fields* respectively, see Fig. 1-a. Fields have a certain range and contain information about the source agent. AGV fields have a fixed range, while the range of transport fields is variable. Fields are refreshed at regular times, according to a predefined refresh rate.

- **AGV agents store received fields.** When an AGV agent perceives fields, it stores the data contained in these fields in a *field-cache*. The field-case consists of a number of cache-entries. Each cache entry contains the source of the received field (an ID), the most recent data contained in that field and a *freshness*. The freshness is a measure of how up-to-date the cached data is. Fig. 1 shows the field-cache of AGV A through the scenario.

- **AGV agents construct calculated-fields to decide their movement.** An AGV agent constructs a *calculated-field* to decide in which direction to drive from a station. A calculated-field is a combination of the received fields, which are stored in the field-cache. The lower the freshness of a cache-entry, the lower the influence of the associated field on the calculated-field.

  The calculated-field is constructed locally and temporarily from the last selected target station (this is the last station that is added to the look ahead buffer, we further clarify this below) and contains values for each outgoing segment. An AGV agent follows the calculated-field in the direction of the smallest value. This can be considered as following the calculated-field downhill, in the direction of the *gradient*.

  Transport fields have an *attractive* influence on the calculated-field, which results in AGVs driving towards unassigned transports. However, it is undesirable that many AGVs drive to the same transport, since they will obstruct each other and travel superfluous distance. To remedy this, AGV fields have a *repulsive* influence.

  In Fig. 1-b, AGV A constructs a calculated-field on a station. Although transport 1 is closer, the calculated-field will guide AGV A towards transport 2. This is the result of the repulsive effect of AGV B. Notice that it would have been ineffective for AGV A to drive towards transport 1, since AGV B is closer and likely will reach the transport first.

- **Task assignment occurs at pick up.** Task assignment is delayed until an AGV actually reaches the pick location and picks up the load. This results in a greater flexibility with respect to task assignment, in comparison for example to the CNET protocol, which assigns a transport as soon as it enters the system. By delaying task assignment, the field-based approach can cope with unforseen situation like transports which suddenly popup, as illustrated in Fig. 1-c. While AGV A is driving towards transport 2, a new transport (transport 4) appears close to AGV A. Since no transport has been assigned to AGV A yet, it can drive towards the closer transport 4.

4.2 Software Architecture

The software architecture of the AGV system is shown in Fig. 2. This architecture is based on a reference model for situated MAS that is described in [20, 22, 23]. Successively, we explain the different agent components. The architecture of the environment is not further discussed here.

- **Field-cache**: This module of the AGV agent stores the information of fields received from other AGV agents and transport agents in cache-entries. A freshness is associated with each cache-entry, which is a measurement of how up-to-date the entry is.

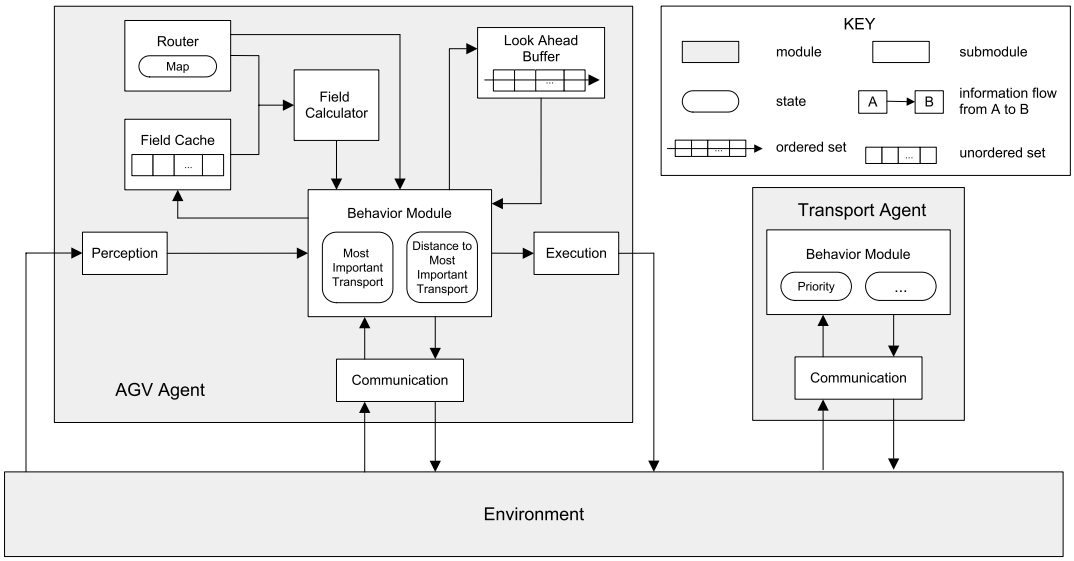


Figure 2: ***The architecture of the AGV transportation system.***

- **Router**: This module contains a global map of the stations and segments. The router is used by the AGV agent to calculate paths and distances from one station to another. We used a *static* router that uses the A* algorithm [8]. However, the approach discussed in this paper is fully compatible with a *dynamic* router that would take dynamic runtime information into account such as traffic distribution or blocked segments.

- **Look-ahead buffer**: As explained in section 2, an AGV has to make routing decisions over a certain distance in advance to allow save and smooth movement. This distance, called the *look-ahead*, is not a straight line distance, but is measured discretely, in terms of stations and segments of the environment map. The look-ahead buffer of an AGV agent is an ordered array that holds these routing decisions. It represents a path that the AGV will follow with certainty, unless something catastrophical (such as a crash) happens.

- **Field calculator**: This module constructs a calculated-field locally and temporarily from the last selected target station (i.e. the last station added to the look-ahead buffer) by combining the received fields, which are stored in the field-cache. The higher the freshness of a cache-entry, the more influence the field associated with the cache-entry will have on the construction of the calculated-field. Thus, although still used, less importance is given to outdated information. The field calculator makes use of the router to calculate the values of the calculated-field on different positions. The gradient of the calculated-field is used as driving direction on the target station.

- **Behaviour module**: The behavior module continuously reconsiders the dynamic conditions in the environment and selects appropriate actions to achieve the agent's goals. The behaviour module of an AGV agent also maintains the *current most important transport* and the *distance to the most important transport*. These values are used to achieve the repulsive influence of AGV agents, as described above. The behaviour module of a transport agent maintains, among other data, the current *priority* of the transport and the *field range*. To avoid starvation of the transport, the priority grows over time. The field-range of the transport agent is a function of time and the number of interested AGV agents. The following two high-level descriptions summarize the behavior of the agents during task assignment:

```
{Behavior procedure of the AGV agent}
while idle
 do repeat with constant frequency {
  1. Do communication
   1.1 Send out field
   1.2 Accept received fields and enter
       them in the field-cache
  2. Fill the look-ahead buffer (if needed)
  3. Perform an action
}

{Behavior procedure of the transport agent}
while not assigned
 do repeat with constant frequency {
  1. Do communication
   1.1 Send out field
   1.2 Accept received messages
  2. Calculate priority and field-range
}
```

5. TEST RESULTS

This section discusses the main test results obtained from applying the field-based task assignment approach. For a detailed discussion of the AGV simulator used in the tests we refer to [1, 17]. We start this section with explaining the test setting and subsequently we elaborate on the test results.

5.1 Setting

Map. All tests were performed on the map of a real layout that has been implemented by Egemin at EuroBaltic, see Fig. 3. The size of the physical layout is 134 m x 134 m. The map has 56 pick and 50 drop locations. The six locations at the lower side of the map are only pick locations, all other locations are pick and drop locations.

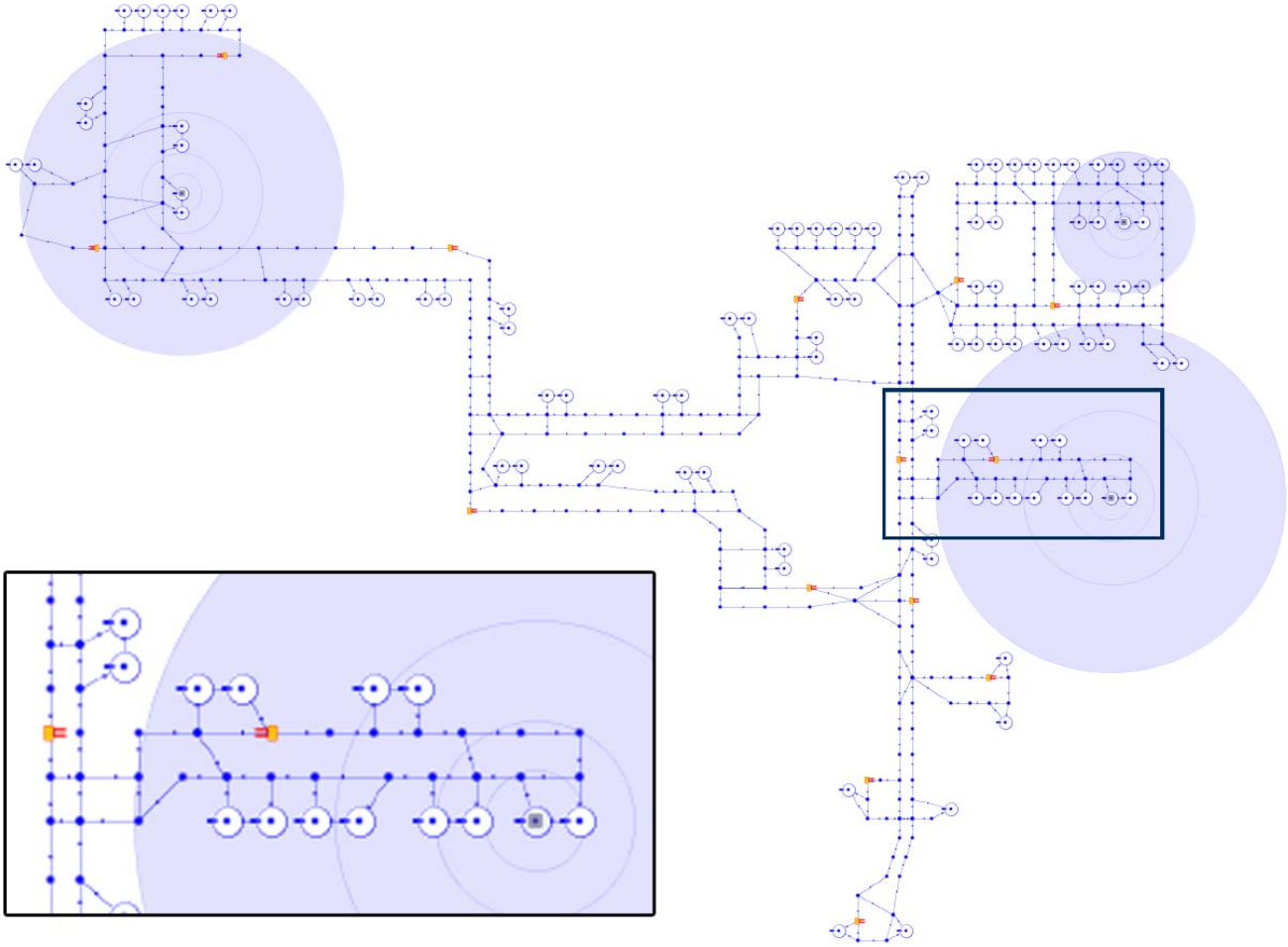


Figure 3: ***Map used in the tests.***

To allow running tests with the current implementation of the AGV simulator, we made a number of small modifications to the map, e.g., all curved segments were changed to straight lines.

Transport profile and AGVs. We used a standard transport test profile that is been used by Egemin for testing purposes. This profile generates 140 transports with a random pick location and a random drop location per simulation run, corresponding to the number of transports that are generated in 1 hour real time.

In the simulation, we used 14 AGVs just as in the real application. The average speed of driving AGVs is 0.7 m/s, while pick and drop actions take an average amount of time of 5 s.

Infrastructure. The AGV simulator and the multiagent system used for testing the field-based approach for task assignment are written in Java. An explanation of the working of the AGV simulator, the source code, as well as the tests results are available on the web [1]. The simulator uses a framework for time management [9] to make sure the simulation results are independent of performance characteristics of the execution platform, working load of the processor, or amount of memory.

To allow running the huge amount of simulations, we used a cluster of 40 machines: P4 2Ghz, 512MB RAM, Debian Stable 3.0. Testing one set of parameter values took approximately 30 hours of simulation time.

Metrics. Every simulation was run for 50000 timesteps, corresponding to approximately 1 hour real time, i.e. one time step represents 20 ms in real time. All test results displayed in the paper are average values over 20 simulation runs. Performance is measured in terms of throughput and reaction time. Additionally, the amount of communication needed is measured.

Reference algorithm: CNET. We used CNET as a reference algorithm to compare the field-based approach. With CNET each transport that enters the system is assigned as soon as possible to the most suitable AGV (i.e., an idle AGV for which the cost to reach the pick location is minimal). When transports can not be assigned immediately, they enter a waiting status. All waiting transports are ordered by priority, and this priority determines the order in which transports are assigned. The priority of transports grows over time, the same way as in the field-based approach.

5.2 Results

A number of tests have been performed to measure the performance of the field-based approach, both for tuning the parameters of the field-based approach and for comparison with CNET under varying circumstances and parameters. Due to space limitation we limit the discussion to the main test results. For a detailed discussion of the remaining test results we refer to [17].

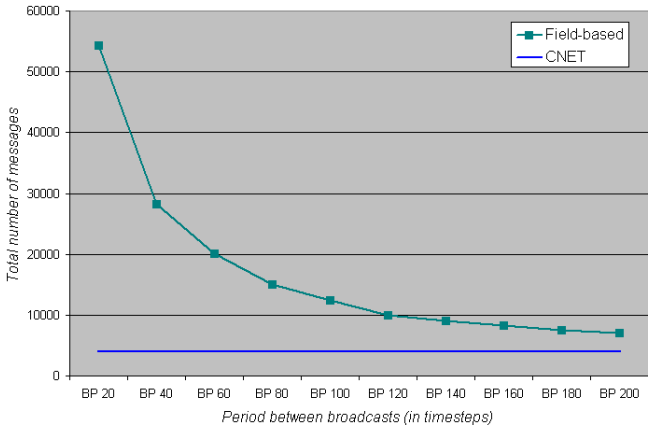


Figure 4: ***Amount of messages being sent.***

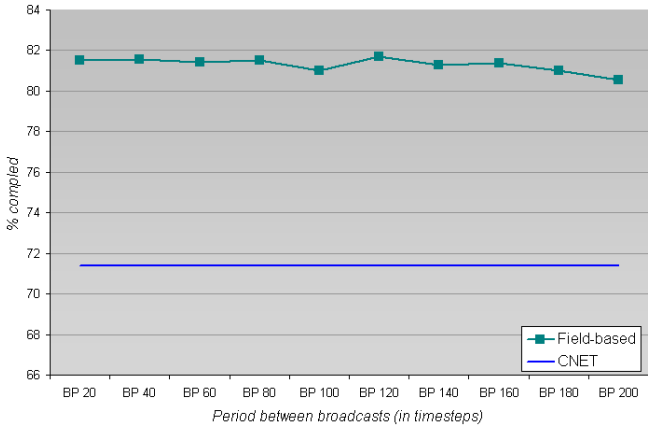


Figure 5: ***Percentage of completed transports.***

5.2.1 *Refreshment of Fields*

As stated before, maintenance of fields in our approach is achieved by periodically broadcasting the status of the fields (both position and strength). Obviously, the period between subsequent broadcasts strongly affects the amount of messages being sent. Making the period too short will produce a huge number of useless messages, but each agent in the system will have up-to-date information. On the other hand, if the broadcast period is too long, AGVs may have outdated information about the fields and probably miss some opportunities.

Fig. 4 shows the expected decrease in number of messages sent if the period between two broadcasts increases. BR20 corresponds to a field refresh each 20 time steps, i.e. each agent broadcasts its field status every second. With a field refresh each 200 time steps (BP200, i.e. every 10 s) the number of messages sent with the field-based approach is 1.7 times higher then with CNET. Of course what is interesting are the implications of reducing the field refresh on the performance of the system.

5.2.2 *Performance and Field Refresh*

Fig. 5 depicts the percentage of transports handled as a function of the field refresh rate. It can be seen that the percentage of completed tasks fluctuates around 81,5% and slowly decreases, the difference between BR20 and BR200 is only 1% but still significantly

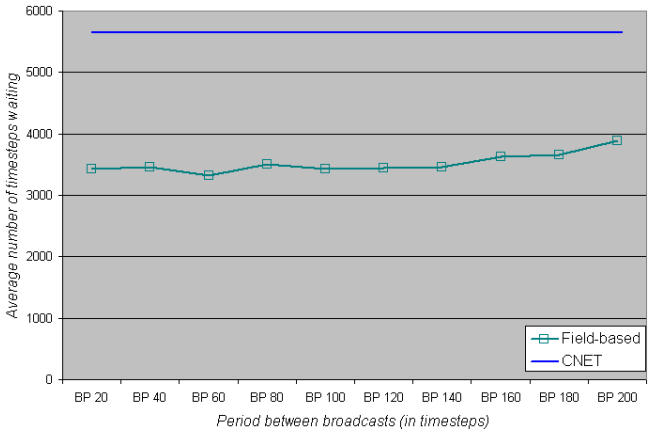


Figure 6: ***Average wait time for transports.***

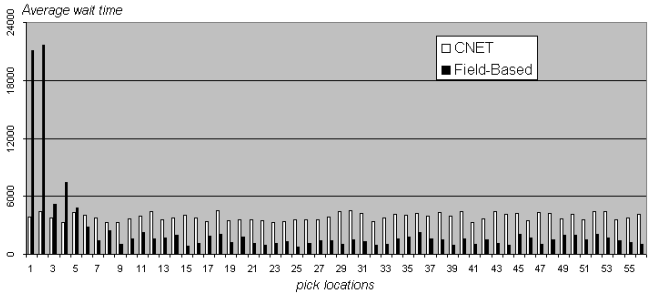


Figure 7: ***Average wait time for transports per pick location.***

better then the CNET approach. Fig. 6 illustrates that the average waiting time slowly increases with lower refresh rates, here the difference between BR20 and BR200 is 14%, which is still 31% better then the CNET approach.

These results clearly illustrate that communication overhead can be reduced by using a longer broadcast period, without significant performance loss. Overall, the throughput of the field-based approach is 10% higher than the throughput of CNET. The average waiting time of a transport is 39% lower, while the average number of transports waiting at each time step was around 20% lower for the field-based approach. [17] elaborates on the distance travelled by the AGVs. On average, the travelled distance of all AGVs with the field-based approach was 33% lower compared to CNET which is a result of a more optimal allocation of tasks.

5.2.3 *Average Waiting Time per Pick Location*

Although the average waiting time for transports is significantly better for the field-based approach compared to the reference algorithm, it is interesting to compare the average waiting time for transports per pick location.

Fig. 7 shows the average wait time for transports grouped by pick location. Clearly, the CNET reference algorithm achieves a more equal distribution. In particular, the wait times for pick locations 1 to 5 are significantly higher for the field-based approach. This drawback can be explained as follows: because the pick locations 1 to 5 are far away from the main traffic in the warehouse, the chance an AGV will be close to the pick location is significantly lower and this decreases the chance for immediately attraction an idle AGV when a new transport pops up. Starvation is prevented since the priority

of the transports on the remote locations gradually increasing when the load is not picked. It simply takes a longer time for the field to "grow" and attract AGVs compared to the immediate assignment of an AGV in the CNET protocol. A possible remedy to this problem is to increase the strength of fields of transports on isolated locations right from the moment the transport is created.

Finally, gradient field-based approaches are known to suffer from local minima. To deal with this problem agents do not use straight distances to sources of fields to construct calculated fields, but use a router that takes into account real distances along the paths on the map. As a result, we experienced little problems with local minima.

6. CONCLUSION

In this paper we presented a field-based approach for task assignment in AGV transportation systems. In this approach, each idle AGV is guided toward a load of an unassigned transport by following the gradient of a field that combines attracting fields received from transports and repulsing fields received from competing AGVs. By delaying the definitive assignment of a transport until the load is finally picked the approach achieves the required flexibility to exploit opportunities that may arise while AGVs search for transports in the highly dynamic environment.

Extensive tests show that the field-based approach outperforms standard CNET for throughput as well as for average waiting time of transports. On the other hand, the approach requires a limited amount of additional bandwidth, and the waiting time for isolated pick locations is higher compared to CNET. We experienced little problems with the traditional difficulties of field-based approaches, including parameter tuning and problems with local minima. As an overall conclusion, field-based task assignment is a promising approach in the domain of AGV transportation systems.

As future work, we aim to compare the field-based approach with other flexible approaches for task assignment. Currently we study a dynamic version of CNET in which transport agents and idle AGV agents continuously reconsider the assignment of transports until loads are picked, aiming to exploit opportunities that arise in the environment. Another interesting venue for future work, is to separate the management of fields from the agent logic by using a *virtual environment* as proposed in [21]. In this approach, agents may simply perceive fields in the virtual environment, while this latter–supported by middleware–takes the burden of managing the spreading of fields. This model improves separation of concerns by making the environment responsible for managing fields, which responsibility conceptually belongs to the environment.

7. REFERENCES

[1] Automatic Guided Vehicle Simulator, K.U.Leuven, 2005. http://www.cs.kuleuven.ac.be/∼distrinet/taskforces /agentwise/agvsimulator/.

[2] S. Aknine, S. Pinson, and M. F. Shakun. An extended multi-agent negotiation protocol. *Autonomous Agents and Multi-Agent Systems*, 8:5–45, 2004.

[3] N. Boucke, D. Weyns, T. Holvoet, and K. Mertens. Decentralized allocation of tasks with delayed commencement. In *Proceedings of European Workshop on Multiagent Systems, Barcelone, Spain*, 2004.

[4] G. Buchman, D. Cohen, P. Vernaza, and D. Lee. The University of Pennsylvania Robocup 2005 Legged Soccer Team. *http://www.cis.upenn.edu/robocup/UPenn05.pdf*, 2005.

[5] Egemin Modular Controls Concept. EMC ^{2} project, IWT, Belgium. *http://emc2.egemin.com/*, 2005.

[6] A. Farinelli, L. Iocchi, D. Nardi, and V. Ziparo. Task Assignment with Dynamic Perception and Constrained Tasks in a Multi-Robot System. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.

[7] J. Ferber. *Multiagent systems: An introduction to distributed artificial intelligence*. Addison-Wesley, London, UK, 1999.

[8] P. Hart, N. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 1968.

[9] A. Helleboogh, T. Holvoet, D. Weyns, and Y. Berbers. Extending time management support for multi-agent systems. In *Multiagent and Multiagent-based Simulation, New York, USA*, 2005.

[10] T. Knabe, M. Schillo, and K. Fischer. Improvements to the FIPA Contract Net Protocol for Performance Increase and Cascading Applications. *Proceedings of the Workshop on Multiagent Interoperability*, 2002.

[11] T. Knabe, M. Schillo, and K. Fischer. Interorganizational Networks as Patterns for Selforganizing Multiagent Systems. *Autonomous Agents and Multiagent Systems*, 2003.

[12] L.Breton, S. Maza, and P. Castagna. Simulation multi-agent de systèmes d'AGVs: comparaison avec une approche prédictive. 5e *Conférence Francophone de Modélisation et Simulation*, 2004.

[13] M. Mamei and F. Zambonelli. Motion coordination in the quake 3 arena environment: a field-based approach. *First International Workshop on Environments for Multiagent Systems*, 2003.

[14] M. Mamei and F. Zambonelli. Programming pervasive and mobile computing applications with the TOTA middleware. In *Proceedings of the 2nd International Conference on Pervasive Computing and Communications*. IEEE Computer Society, Washington, DC, USA, 2004.

[15] M. Mamei, F. Zambonelli, and L. Leonardi. Co-Fields: A Physicall Inspired Approach to Distributed Motion Coordination. *IEEE Pervasive Computing*, 2004.

[16] F. D. Paoli and G. Vizzari. Context dependent management of field diffusion: an experimental framework. *Workshop Dagli Oggetti agli Agenti, Villasimius, Italy*, 2002.

[17] W. Schols, T. Holvoet, N. Boucke, and D. Weyns. Gradient Field Based Transport Assignment in AGV Systems. In *CW-425, Technical Report, Departement of Computer Science, Katholieke Universiteit Leuven, Belgium. http://www.cs.kuleuven.ac.be/publicaties/rapporten/CW/2005/.*

[18] O. Shehory, S. Kraus, and O. Yadgar. Emergent cooperative goal-satisfaction in large scale automated-agent systems. *Artificial Intelligence*, 110(1):1–55, 1999.

[19] R. G. Smith. The contract net protocol: High level communication and control in a distributed problem solver. In *IEEE Transactions on Computers, C-29(12)*, 1980.

[20] D. Weyns and T. Holvoet. Formal Model for Situated Multiagent Systems. *Fundamenta Informaticae*, 63(2-3), 2004.

[21] D. Weyns, K. Schelfthout, T. Holvoet, and T. Lefever. Decentralized control of E'GV transportation systems. In *4th Joint Conference on Autonomous Agents and Multiagent Systems, Industry Track*, 2005.

[22] D. Weyns, E. Steegmans, and T. Holvoet. Protocol based communication for situated multiagent systems. In *3th Conference on Autonomous Agents and Multi-Agent Systems*, 2004.

[23] D. Weyns, E. Steegmans, and T. Holvoet. Towards active perception for situated multi-agent systems. *Applied Artificial Intelligence*, 18(8-9), 2004.

Learning the Task Allocation Game *

Sherief Abdallah† and Victor Lesser
University of Massachusetts
Amherst, MA
{shario,lesser}@cs.umass.edu

ABSTRACT

The distributed task allocation problem occurs in domains like web services, the grid, and other distributed systems. In this problem, the system consists of servers and mediators. Servers execute tasks and may differ in their capabilities, e.g. one server may take more time than the other in executing the same task. Mediators act on behalf of users, which can potentially be other mediators, and are responsible for receiving tasks from users and allocating them to servers.

This work introduces a new gradient ascent learning algorithm that outperforms state of the art multiagent learners on this problem. We experimentally show that our algorithm converges faster and is less sensitive to tuning parameters than other algorithms. We also provide an informal proof that WPL has the same convergence guarantee as the best known algorithm, GIGA-WoLF. We also show that our algorithm converges in Jordan's and Shapley's games where many other algorithms fail to converge. Finally, we verify the practicality of our algorithm in the distributed task allocation domain, comparing its performance to an optimal global solution.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms, Experimentation

*This material is based upon work supported by the National Science Foundation Engineering Research Centers Program under NSF Award No. EEC-0313747. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

†A student author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Keywords

Reinforcement Learning, Multiagent Systems, Task Allocation, Markov Decision Process

1. INTRODUCTION

The distributed task allocation problem occurs in domains like web services, the grid, and other distributed systems. The problem can be illustrated by Figure 1. There are L servers and N mediators. Servers execute tasks and may differ in their capabilities, e.g. one server may take more time than another server in executing the same task. Mediators receive tasks from users, which can potentially be other mediators. Mediators are responsible for allocating tasks to servers. The goal is to find an optimal allocation that minimizes the average turn around time.[1] Turn Around Time, or TAT, is the time interval between a task arrival and its completion. It includes both the task processing time and the task waiting time at the server. Average TAT is then averaged over tasks to produce TAT.

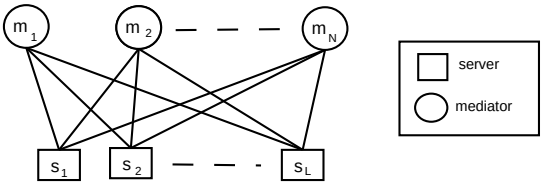


Figure 1: An example of distributed task allocation.

What makes this problem challenging is the *limited observability* of each mediator. In a large open system, a mediator will not be aware of all other mediators, their queues, their actions, or the queue length of each server. Ideally, one would like a learning algorithm that finds the optimal policy for each mediator. We identify certain characteristics that are desirable in such a learning algorithm.

- **Ability to learn a stochastic policy.** This is necessary in order to minimize TAT. For example, consider a system of two servers and three mediators. Any deterministic joint policy will overload one server more than the other, increasing the average TAT.
- **Convergence.** If the mediator policies do not converge, then oscillation between policies will occur making the system unstable.

[1]Other optimal criteria are also possible.

- **Adaptability.** This is directly related to the convergence property: not only do we need the algorithm to converge, we want it to converge quickly. This is important in a dynamic and open system that is constantly changing.

This work develops a gradient ascent learning algorithm that meets all these requirements and is better than the state-of-the-art algorithms. We analyze its performance in two-action games and prove its convergence. We also show that our algorithm converges in Jordan's and Shapley's game where many other algorithms fail to converge. Furthermore, we show that our algorithm converges faster and is less sensitive to tuning parameters when compared to previous approaches. Finally, we verify the practicality of our algorithm in the distributed task allocation domain, comparing its performance to an optimal global solution (computed using queuing theory).

This paper is organized as follows. Section 2 gives necessary definitions of game theory, which provides an elegant methodology for analyzing the performance of multiagent learners. Section 3 describes our algorithm and proves its convergence. Section 4 describes in more detail the state of the art algorithms. Section 5 verifies our claims in both simple games and distributed task allocation. Finally, we conclude in Section 6.

2. GAME THEORY

Game theory provides an elegant framework for analyzing learning in multiagent systems. It was used by previous researchers [4, 3, 2] to analyze the convergence properties of multiagent learning algorithms. This section gives an overview of the game theory framework, illustrating how it can be used to model distributed task allocation. The following section describes our algorithm and analyzes its convergence using game theory.

A game in normal form is defined by the tuple $\langle n, A_1, ..., A_n, R_1, ..., R_n \rangle$, where n is the number of players[2] in the game, A_i is the set of actions available to agent i, and $R_i : A_1 \times ... \times A_n \rightarrow \Re$ is the reward agent i will receive as a function of the joint action executed by all agents. If the game has only two players, then it is convenient to define their reward function as a payoff matrix as shown in Table 1. Each cell (i, j) in the grid represents the payoff received by the row player (Player 1) and the column player (Player 2), respectively, if the row player plays action i and the column player plays action j. The first three games in Table 1 were used in evaluating previous research [3, 1, 2].

The game in Table 1(d) represents the task allocation game at time t for two mediators and two servers. Action s_i corresponds to the row/column mediator assigning the task to server s_i. l_i^t is the expected TAT if the task is allocated to server i at time t. This is a compact representation of a set of games with constant payoff. The game the mediators play at time t depends on the status of the server queues at that time. However, due to the limited observability of each mediator, mediators can not distinguish between these games. In other words, an optimal policy would be the policy that performs best, on average, over time. While we will limit our theoretical analysis to games with payoffs that remain constant over time, Section 5 shows the effectiveness of our algorithm experimentally in the task allocation game with varying payoffs and involving more players and servers.

[2]We use the terms: agent, player, and mediator interchangeably.

	a1	a2
a1	2,1	0,0
a2	0,0	1,2

(a) coordination

	H	T
H	1,-1	-1,1
T	-1,1	1,-1

(b) matching pennies

	a1	a2
a1	0,3	3,2
a2	1,0	2,1

(c) tricky

	s1	s2
s1	$-(l_1^t+2), -(l_1^t+2)$	$-(l_1^t+1), -(l_2^t+1)$
s2	$-(l_2^t+1), -(l_1^t+1)$	$-(l_2^t+2), -(l_2^t+2)$

(d) task allocation

Table 1: 2-action games

A policy (or a strategy) of agent i at time t is $\pi_i^t \in PD(A_i)$, where PD is the set of probability distributions over actions A_i. In case of two-player-two-action games (like the ones in Table 1) we refer to the joint policy of the two players by (p, q), where $\pi_1 = (p, 1-p)$ and $\pi_2 = (q, 1-q)$. Let the vector r_i^t be the actual reward agent i would receive at time t for each of its actions. This is the only feedback signal agent i would receive. The expected payoff agent i would get given its policy is $V_i^t(\pi_i) = \pi_i^t.r_i^t$.

2.1 Learning and Convergence

The primary goal of any rational learner would be to maximize its expected payoff by changing its policy. The problem is that r_i^t is not fixed and depends on other agents' policies. Consider for example the matching pennies game in Table 1(b). Each player can play either head (H) or tail (T). The payoff of Player 1 (row player) is $r_1^t = (2q-1, 1-2q)$, which is dependent on Player 2's policy q. Therefore, an optimal policy for Player 1 depends on the current policy of Player 2 and vice versa.

This dependence between optimal policies may prevent the convergence of a simple policy gradient algorithm. Consider the matching penny example again. Assume that initially the joint policy of the two players is $(p^0 = 1, q^0 = 0)$ (i.e. Player 1 always plays H and Player 2 always plays T), this means $r_1^t = (-1, 1)$ and $r_2^t = (-1, 1)$. Player 1 realizes it can do better by moving its policy toward action T by a small step of size δ. The new joint policy is $(p^1 = 1 - \delta, q^1 = 0)$. The process continues, until time t where the joint policy is $(p^t = 0.5 - \delta, q^t = 0)$. In this case $r_2^t = (\delta, -\delta)$. Player 1 continues moving its policy toward $p = 0$ while player 2 starts increasing q. This process continues and the two players keeps oscillating around the joint policy $(p = 0.5, q = 0.5)$ as shown in the Figure 2.

The joint policy $(p^* = 0.5, q^* = 0.5)$ that both players are oscillating around is the *Nash Equilibrium* of the matching pennies game. A joint policy is a Nash Equilibrium, NE, iff no agent can get higher expected reward by changing its policy unilaterally. A pure NE is the one where all agents policies are deterministic. Otherwise the NE is mixed. The coordination game has two pure NE, $(1, 1)$ and $(0, 0)$. The other two games have one mixed NE, $(0.5, 0.5)$.

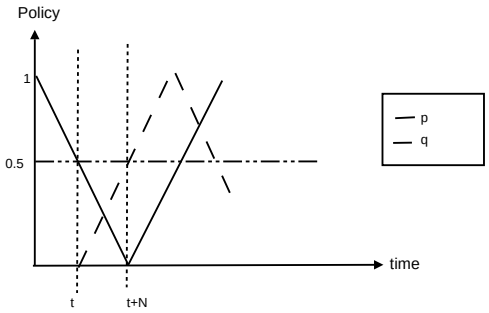


Figure 2: An illustration of policy oscillation.

The oscillation we have just described is common in policy gradient algorithms that change policies depending only on actual reward r_i^t [7, 10]. This is particularly true when the NE is mixed. The main idea behind our algorithm is to slow down learning such that this oscillation dampens until the joint policy gradually stabilizes to a NE.

3. THE WEIGHTED POLICY LEARNER ALGORITHM

Algorithm 1 illustrates our WPL algorithm. is the policy gradient which is used to update the policy π. The idea of our algorithm is to start learning fastest when changes its direction and then to gradually slow down learning if the policy gradient does not change its direction. We detect change in the direction of the gradient using the difference between action rewards (because the policy gradient follows the reward). If the reward of action a is decreasing, then the change in $\pi(a)$ is weighted by $\pi(a)$, otherwise it is weighted by $(1-\pi(a))$. Therefore, the change in $\pi(a)$, (a), is largest when either $\pi(a)$ is near 1 and $r(a)$ is lower than average reward $\bar{r}$, or $\pi(a)$ is low and $r(a)$ is higher than $\bar{r}$. The process repeats until convergence is reached and we get a damped oscillation as shown in Figure 3. The *limit* function is adopted from [10] with minor modifications to ensure fixed exploration: $limit(x) = argmin_{x':valid(x')}(x - x')$, i.e. $limit(x)$ returns a valid policy that is closest to x. A policy is valid if it sums to 1 and every action is played with miminum probability ϵ (for exploration). The following section discusses the convergence of WPL.

Algorithm 1: WPL: Weighted Policy Learner

```
begin
    Let r(a) be the (expected) reward for executing
    action a
    r̂ ← total average reward = Σ_{a∈A} r(a) / |A|.
    foreach action a ∈ A do
        (a) ← r(a) − r̂
        if (a) > 0 then (a) ← (a)(1 − π(a))
        else (a) ← (a)(π(a))
    end
    π ← limit(π + )
end
```

3.1 Convergence Proof

This section presents an informal proof that WPL converges for 2x2 games that is verified by experimental results involving 2x2 games as well as other games.[3] In the case of a pure NE, any gradient ascent player that follows the actual reward r^t will converge to the pure NE [7, 10]. The interesting case is when the NE is mixed. This is usually the case in the distributed task allocation problems, where mediators compete for servers.

Because $_1^t(a1) + _1^t(a2)$ will always sum to zero in 2x2 games, the only effect of the function *limit* is to constrain each action probability within 0 and 1. This actually helps convergence because it prevents players from moving further away from the NE (since by definition any NE is within the space of valid policies) [7]. Therefore, the limit function will be ignored for the rest of the proof.

LEMMA 1. *WPL learners stop learning iff they reached an NE.*

PROOF. From the WPL description, the change in the row agent policy, or $_1^t$ is zero only if one of the following is true: $p^t = 0, p^t = 1$, or $r_1^t(a) - \widehat{r}^t$. In practice, p is bounded by $[\epsilon, 1-\epsilon]$, where $\epsilon > 0$. This forces agents to keep exploring non-optimal actions in case the environment changes and non-optimal actions becomes optimal.[4] Therefore, ^{t} is zero iff $r_1^t(a1) = r_1^t(a2)$. The same argument goes for player 2, i.e. it stops learning iff $r_2^t(a1) = r_2^t(a2)$. This is only true in a Nash equilibrium.[5] □

LEMMA 2. *A player changes its gradient direction iff the other player crosses its NE strategy.*

PROOF. We say Player 1 *crosses the NE* at time t iff $p^{t-1} < p^* < p^{t+1}$, where p^* is Player 1's NE policy. For example, in Figure 2 Player 1 crosses the NE at time t. The lemma follows from the fact that when a player is at a NE, the values of both actions of the other player are equal. This means that the relative values of each of player 1's actions switch when player 2 crosses its NE strategy. This changes the gradient direction of player 1 because the gradients follow the action with highest expected reward. The lemma is depicted in Figure 3. □

LEMMA 3. *The two agents using WPL will continue to oscillate around the NE unless both converge to the NE.*

PROOF. Both players try to maximize their reward by moving their strategy toward the action that yields more rewards. At any time t, one agent (player 1) will be better off by moving its strategy closer to the NE strategy while the other agent (player 2) is better off by moving its strategy away from the NE strategy (because we are considering only competitive game, i.e. mixed NE). Eventually, Player 1 will cross its NE strategy. This will cause player 2 to switch its policy gradient direction toward the NE strategy, while player 1 is now moving away from the NE. This process continues, unless both players converge to an NE. □

[3]We are currently working on a more formal proof using differential equations.

[4]Even in a static environment agents need time to correctly estimate the rewards r_i^t either because of noise or other agents learning; ϵ then decreases gradually with time t such that $t \overset{lim}{\to} \infty\ (\epsilon^t) \to 0$.

[5]This is a property of two-action games (and can be generalized for larger games): if players are at a mixed NE then both actions for each player have the same expected reward.

LEMMA 4. *A WPL learner slows down (i.e. the magnitude of the policy gradient decreases) if its gradient direction does not change.*

PROOF. We will prove the lemma for Player 1 and a similar argument goes for the column player. Let $|\ _1^t| = |\ _1^t(a1) + \ _1^t(a2)| = 2|\ _1^t(a1)| = 2(1-p^t)(r_1^t(a1) - r_1^t(a2))$ be the magnitude of gradient change, assuming $r_1^t(a1) > r_1^t(a2)$. A WPL learner slows down its learning iff $|\ _1^{t+1}| < |\ _1^t|$. If the gradient direction is the same, then $r_1^{t+1}(a1) > r_1^{t+1}(a2)$, and therefore $|\ _1^{t+1}| = 2(1-p^{t+1})(r_1^{t+1}(a1) - r_1^{t+1}(a2))$. There are two cases:

- Either $r_1^{t+1}(a1) - r_1^{t+1}(a2) <= r_1^t(a1) - r_1^t(a2)$. In this case, the gradient magnitude decreases; i.e. $|\ _1^{t+1}| < |\ _1^t|$ because $1-p^{t+1} = 1-p^t - \ _1^t(a1) < 1-p^t$. If the gradient magnitude decreases then the learning slows down.

- Or $r_1^{t+1}(a1) - r_1^{t+1}(a2) > r_1^t(a1) - r_1^t(a2)$. We prove by contradiction that this case can not occur. If $r_1^{t+1}(a1) - r_1^{t+1}(a2) > r^t(a1)_1 - r_1^t(a2)$ then the only explanation is that Player 2 moved faster. However as we have just proved, Player 2 will slow down if its gradient $\ _2^{t+1}$ is moving in the same direction as $\ _2^t$. Therefore, Player 2 must have changed its gradient direction. However, if $\ _2^t$ increases $r^t(a1) - r^t(a2)$ then $\ _2^{t+1}$ must decrease this difference (because it is in the opposite direction), and therefore $|\ _1^t|$ actually decreases further.

□

LEMMA 5. *A WPL learner moves toward its NE strategy faster than it moves away from its NE strategy.*

PROOF. From the previous lemma, an agent slows down as long as it does not change its gradient direction. Therefore, after player 1 crosses its NE strategy (i.e. starts moving its policy away from the NE) player 1 does not change its gradient direction (while player 2 changes its gradient direction) and consequently player 1 moves away from its NE strategy slower than player 1 moves toward its NE strategy. □

Lemma 3 says that both players will continue to oscillate around the NE while Lemma 5 says that each player will slow down as long as it does not change its policy gradient direction. This leads to a damping effect as illustrated by Figure 3 until both players reach the NE. Our experimental results reflect this same behavior as Section 5 will show.

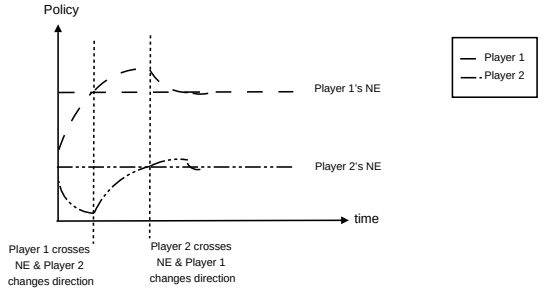


Figure 3: An illustration of WPL convergence.

4. RELATED WORK

There are many algorithms that have been proposed for multiagent learning. Unfortunately, all of them either fail to achieve all the three desired characteristics that are listed in Section 1 or assume agents can observe each other's actions. Independent Learners and Joint Learners [4] only learn a deterministic policy using the Q-Learning algorithm[8] and do not have any convergence guarantee. Joint Learners also require the knowledge of other agents' actions, an assumption that is not practical in large open systems.

Equilibrium learners such as Nash-Q [6], OAL [9], AWESOME [5] require that each agent observe other agents' actions. Each agent then computes a stable joint policy (NE). While these learners are guaranteed to converge, similar to joint learners, they are not applicable in open domains.

The gradient ascent learners are the most promising and relevant as they can learn a stochastic policy without observing other agents actions. These algorithms directly follow the expected reward. The Infinitisemal Gradient Ascent algorithm, or IGA [7] and its generalization, Generalized IGA, or GIGA [10] were the first algorithms to provide any convergence guarantees in a multiagent context. They converge to games with pure NE. However both algorithms failed to converge in games with mixed NE, and therefore are not suitable for the distributed task allocation problem.

Several modifications to IGA and GIGA were proposed to avoid divergence in games with mixed NE, including: PHC-WoLF [3], PHC-PDWoLF [1], and GIGA-WoLF [2]. They all used some form of the Win or Learn Fast heuristics [3], whose purpose is to speedup learning if the agent is moving toward its NE policy and slow down learning if the agent is moving away from the NE policy. The main problem with this heuristic is that an agent does not know what policy is an NE policy. Therefore, all the algorithms that use the WoLF heuristic use approximate methods to determine when an agent is moving toward or away from the NE. This is different from our approach, in which an agent slows down its learning as long as its policy gradient has the same direction. Except for GIGA-WoLF, none of the WoLF-based algorithms have any form of convergence guarantee. Therefore, while we will compare our algorithm to all of them, our experiments will focus on comparing our algorithm to GIGA-WoLF.

5. RESULTS

5.1 Practical Issues

5.1.1 Expected Reward vs Most Recent Reward

Our algorithm, along with all previous approaches, needs the expected reward of an action a at time t, or $r^t(a)$. However, this is not known a *priori* and needs to be computed. The common approach in reinforcement learning, which we also use, is the equation $r^{t+1}(a) \leftarrow \alpha R^t + (1-\alpha) r^t(a)$, where R^t is the sample reward received at time t and $0 \leq \alpha \leq 1$ is a learning parameter [8]. Tuning α is not an easy task when the expected reward itself is dynamically changing (because other agents are changing their policies concurrently). Smaller α means more samples contribute to computed expected reward and vice versa.

Using more samples to estimate the reward makes the estimate more accurate. However, the time required to get more samples may in fact degrade the accuracy of reward

estimate. This is because it will take more time for r^t to follow the dynamically changing expected reward. In other words, a higher α value increases the adaptability of the learning algorithm, but may affect convergence which requires r to reflect the expected reward, not a sample. We investigate this tradeoff in our experiments and show that our approach is less sensitive to the value of α than previous approaches.

5.1.2 *Fixing Learning Parameters*

In our experiments we have not used any decaying rates. The reason is that in an open system where dynamics can continuously change, one would want learning to be continuous as well. We have fixed the exploration rate ϵ to 0.1, the learning rate δ to 0.001. Two values of the learning parameter α have been used to measure the tradeoff between accuracy and responsiveness: 0.1 and 1.

5.2 Benchmark Games

To verify our theoretical result, we tested our algorithm for the three games described in 1. Figure 4 plots $\pi(r1)$ and $\pi(c1)$ against time, when the initial joint policy is $([0.1, 0.9]_r, [0.9, 0.1]_c)$ (several initial joint policies have been tested with similar results) and $\alpha = 0.1$. The algorithm converged to an NE in all the games. Note also that the plotted change of agents' policy over time in the two games with mixed equilibrium is almost identical to our theoretical prediction.

The tricky game deserves more attention as it is the most challenging among two-player-two-action games. Figure 5 shows the performance of PHC-WoLF, PHC-PDWoLF, and GIGA-WoLF in this game using the exact same settings of our approach. Only GIGA-WoLF converges but with slower rate than our approach (Figure 4(c)).[6] The performance of PHC-WoLF and GIGA-WoLF conforms to the results reported by their authors [3, 2] (actually PHC-WoLF performs here better than reported results). On the other hand, PHC-PDWoLF diverged in our experiments while it converged in the authors' experiments [1]. A possible explanation is that they used more tuned learning parameters, possibly with decaying rates. Due to the poor performance of both PHC-WoLF and PHC-PDWoLF in the rest of our experiments, the focus will be on GIGA-WoLF thereafter.

Table 2 shows two well known 3-action games used to evaluate GIGA-WoLF [2]. The NE in both games is mixed and equals $([\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]_r, [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]_c)$. GIGA-WoLF is known to converge in the rock-paper-scissors game and diverge in Shapley's game as we verify ourselves.

	c1	c2	c3
r1	0,0	-1,1	1, -1
r2	1,-1	0,0	-1,1
r3	-1,1	1,-1	0,0

(a) rock paper scissors

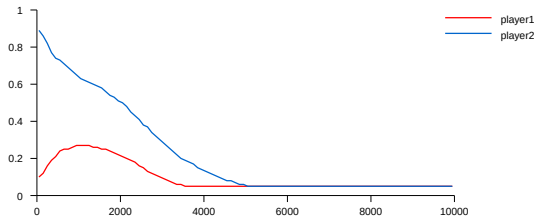
	c1	c2	c3
r1	0,0	1,0	0,1
r2	0,1	0,0	1,0
r3	1,0	0,1	0,0

(b) Shapley's

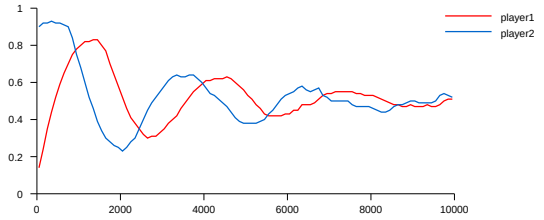
Table 2: 3-action games

Figures 6 and 7 plot the policy of the row player over time

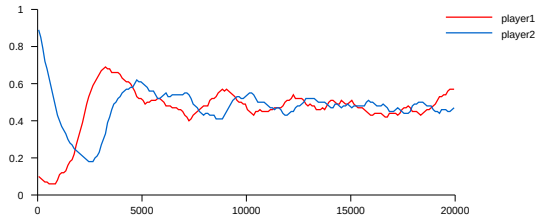
[6]We have tried increasing the learning rate of GIGA-WoLF to 0.01 in order to see if this can speed up its convergence. However, this resulted in GIGA-WoLF diverging.



(a) coordination



(b) matching pennies



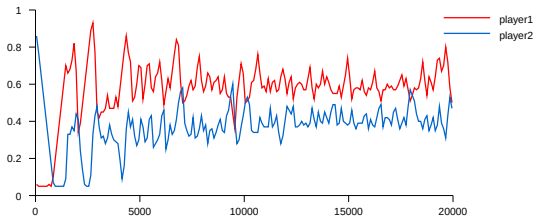
(c) tricky

Figure 4: Convergence of WLP in different two-player-two-action games. The figures plot the probability of playing the first action for each player (vertical axis) against time (horizontal axis).

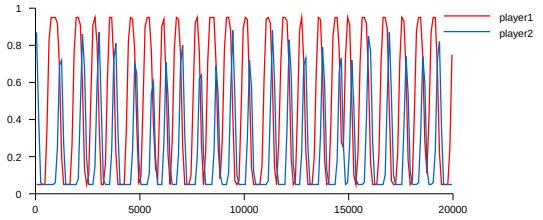
for the two games for both WPL and GIGA-WoLF when the initial joint strategy is $([0.1, 0.8, 0.1]_r, [0.8, 0.1, 0.1]_c)$, $\delta = 0.0003$, and for two values of α: 0.1 and 1. GIGA-WoLF's performance usually gets worse as α increases. GIGA-WoLF also keeps oscillating in Shapley's game regardless of the value of α. WLP on the other hand performs better as α increases and converges in Shapley's game when $\alpha = 1$. We believe the reason is that small α leads to an out of date reward estimate which in turn leads agents to continuously chase the NE without successfully reaching it. One of our future directions is to theoretically analyze this tradeoff between accuracy and recency of the reward and its effect on both convergence and optimality.

5.3 The Task Allocation Game

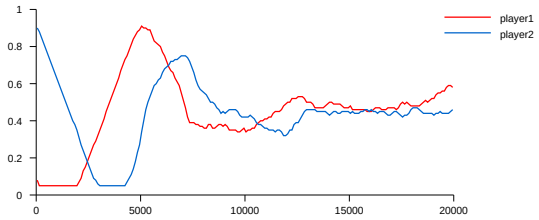
This section evaluates WPL in the distributed task allocation domain. Let N be the number of mediators and L



(a) PHC-WolF



(b) PHC-PdWolF

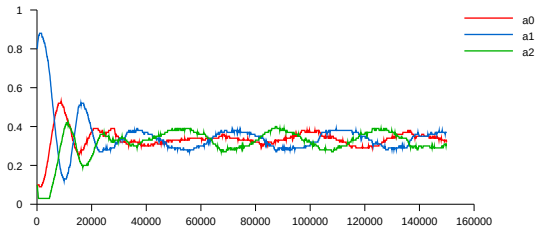


(c) GIGA-WolF

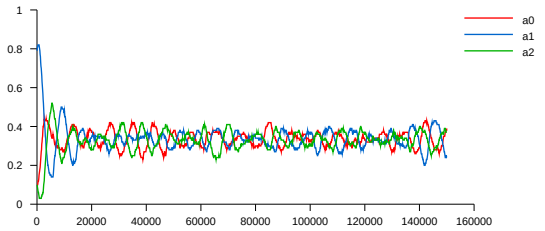
Figure 5: Convergence of the previous approaches in the tricky game. The figures plot the probability of playing the first action for each player (vertical axis) against time (horizontal axis).

be the number of servers. The turn-around-time, or TAT, is the time interval between a task arrival and its completion. An upper bound on the TAT can be computed using queuing theory as follows. Let μ_i be rate by which tasks are serviced at server i. Let λ_j be the rate by which tasks arrive at mediator j. Then the global task arrival rate $\lambda = \sum_j \lambda_j$ and the global service rate $\mu = \sum_i \mu_i$. The global utilization $u = \lambda/\mu$. Then a lower bound on the turn around time is given by the equation $\widehat{TAT} = \frac{1}{\mu}(1 + \frac{u}{1-u})$.

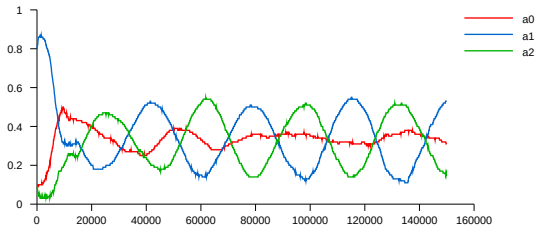
We define $\overline{TAT}$ to be the average TAT seen by each mediator when using WPL. Table 3 shows $\overline{TAT}/\widehat{TAT}$ (lower is better, where 1 is optimal) for different numbers of players N (columns) and different values of load u (rows). This data is collected from a system with 4 servers. Two servers have $\mu_i = 1$ task/time unit and the other two have $\mu = 0.5$ task/time unit. Tasks arrival rate is the same for all agents.



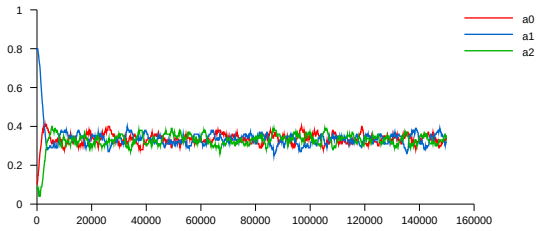
(a) GIGA-WolF, $\alpha = 0.1$



(b) WPL, $\alpha = 0.1$

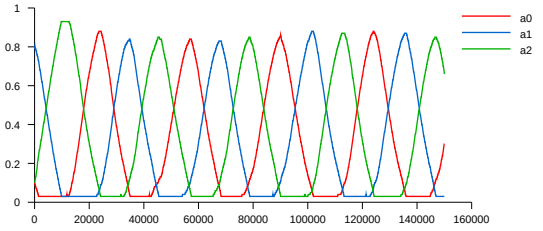


(c) GIGA-WolF, $\alpha = 1$

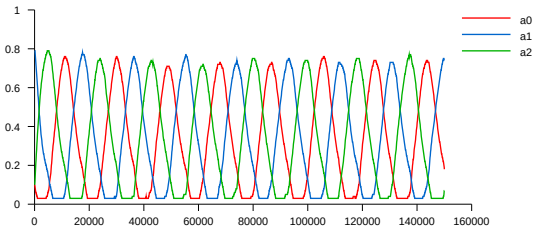


(d) WPL, $\alpha = 1$

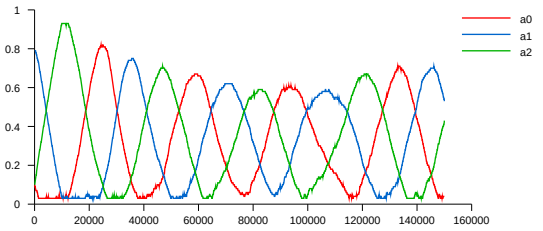
Figure 6: Convergence of GIGA-WoLF and WPL in the rock-paper-scissors game. The figures plot the probability of playing each action of the first player (vertical axis) against time (horizontal axis).



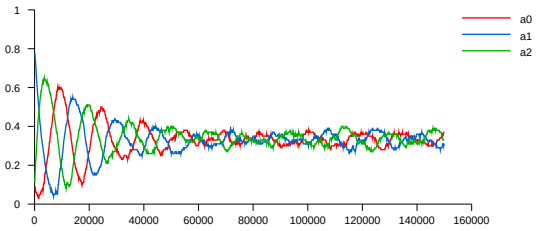
(a) GIGA-WolF, $\alpha = 0.1$



(b) WPL, $\alpha = 0.1$



(c) GIGA-WolF, $\alpha = 1$



(d) WPL, $\alpha = 1$

Figure 7: Convergence of GIGA-WoLF and WPL in Shapley's game. The figures plot the probability of playing each action of the first player (vertical axis) against time (horizontal axis).

The initial policy of all agents is to allocate all tasks to the first server. $\overline{TAT}/\widehat{TAT}$ approaches 1 as u increases. In other words, WPL is closer to optimal in high loads. This is expected because knowing the queue length of all servers is more advantageous in low loads (some servers may have an empty queue while others have nonempty queues). In high loads, all servers have on average constant queue length at all times. We can also notice that $\overline{TAT}/\widehat{TAT}$ does not change much for the same u and different N. This means that WPL scales well with the number of agents.

u \ N	3	4	5
0.3	3.13	3.34	3.44
0.6	2.27	2.3	2.3
0.9	1.68	1.875	1.9

Table 3: $\overline{TAT}/\widehat{TAT}$ for different values of N (columns) and u (rows)

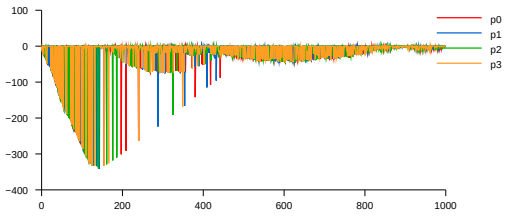
Figure 8 compares the performance of GIGA-WoLF and WPL with 5 mediators and 4 servers. The figure plots the TAT from individual mediators' perspective (TAT is plotted as a negative reward for each mediator). Note that the scale in GIGA-WoLF figure is almost double the scale in the WPL figure. There are two observations from the figures: time of convergence and maximum TAT. By 250 time steps, all mediators using the WLP algorithm witness TAT less than 50 time units. It took GIGA-WoLF more than 450 time units to achieve the same results. The maximum TAT that is ever witnessed by a WPL mediator is under 170 time units. GIGA-WoLF mediators have witnessed TAT of more than 320 time units.

Figure 9 illustrates how the policy of one player converges over time. As expected, for both WPL and GIGA-WoLF, the probability of sending a task to one of the fast servers (a2 and a3) becomes almost double the probability of sending a task to any of the slow servers. However, WPL learns a better policy, where the probabilities of sending a task to any of slower servers are equal (while in GIGA-WoLF there is a consistent difference).

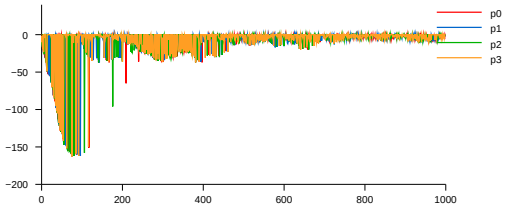
In conclusion, GIGA-WoLF converges slower than WPL and also reaches almost double the maximum TAT that WLP ever reaches. This gives WPL a clear advantage in dynamic environments where agents are continuously adapting to changes in the environment. We believe GIGA-WoLF's slow convergence is due to the way GIGA-WoLF works. GIGA-WoLF relies on learning a slowly moving policy in addition to the actual policy π to determine the direction of the gradient. This requires more time than the WPL algorithm, which changes its gradient direction according to the current expected reward.

6. CONCLUSION AND FUTURE WORK

This work presents WPL, a gradient ascent learning algorithm that outperforms state of the art multiagent learners in the task allocation. We experimentally show that our algorithm converges faster and is less sensitive to tuning parameters than other algorithms. We provide an informal proof of WPL's convergence. We also show that our algorithm converges in Shapley's game where many other algorithms fail to converge. Finally, we verify the practicality

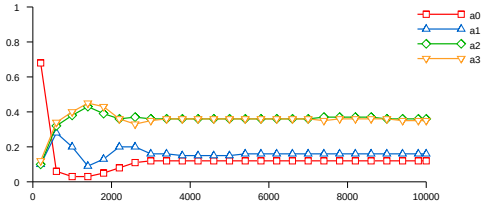


(a) GIGA-WolF

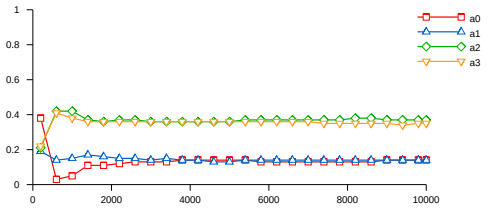


(b) WPL

Figure 8: Convergence of GIGA-WoLF and WPL in distributed task allocation. The figures plot the reward (negative TAT) obtained by each player (vertical axis) against time (horizontal axis).



(a) GIGA-WolF



(b) WPL

Figure 9: Convergence of GIGA-WoLF and WPL in distributed task allocation. The figures plot the policy of a player as the vertical axis (aka the probability of playing each action) against time (horizontal axis).

of our algorithm in the distributed task allocation domain, comparing its performance to an optimal global solution.

In the future, we want to extend our convergence guarantee to more than two players and two actions. We also want to evaluate WPL in more general task allocation settings where mediators may be connected to other mediators. This work also illustrates the importance of computing the expected reward and how it affects convergence. This issue was overlooked in previous work and assumed an implementation issue. Using more samples of the immediate reward may increase the accuracy of the reward expectation but may also make the expectation out of date. Finding the right balance and theoretically analyzing how it affects convergence are interesting research questions that we plan to address.

7. REFERENCES

[1] B. Banerjee and J. Peng. Adaptive policy gradient in multiagent learning. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems.*, 2003.

[2] M. Bowling. Convergence and no-regret in multiagent learning. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 209–216. MIT Press, Cambridge, MA, 2005.

[3] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[4] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI/IAAI*, pages 746–752, 1998.

[5] V. Conitzer and T. Sandholm. Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In *International Conference on Machine Learning*, pages 83–90, 2003.

[6] J. Hu and M. P. Wellman. Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

[7] S. Singh, M. Kearns, and Y. Mansour. Nash convergence of gradient dynamics in general-sum games. In *the 16th Conference on Uncertainty in Artificial Intelligence*, pages 541–548, 2000.

[8] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1999.

[9] X. Wang and T. Sandholm. Reinforcement learning to play an optimal nash equilibrium in team markov games. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*. MIT Press.

[10] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning*, pages 928–936, 2003.

Continuous Refinement of Agent Resource Estimates

David N. Morley Karen L. Myers Neil Yorke-Smith

Artificial Intelligence Center, SRI International
Menlo Park, CA 94025 U.S.A.
{morley,myers,nysmith}@ai.sri.com

ABSTRACT

The challenge we address is to reason about projected resource usage within a hierarchical task execution framework in order to improve agent effectiveness. Specifically, we seek to define and maintain maximally informative guaranteed bounds on projected resource requirements, in order to enable an agent to take full advantage of available resources while avoiding problems of resource conflict. Our approach is grounded in well-understood techniques for resource projection over possible paths through the plan space of an agent, but introduces three technical innovations. The first is the use of multi-fidelity models of projected resource requirements that provide increasingly more accurate projections as additional information becomes available. The second is execution-time refinement of initial bounds through pruning possible execution paths and variable domains based on the current world and execution state. The third is exploitation of additional semantic information about tasks that enables improved bounds on resource consumption. In contrast to earlier work in this area, we consider an expressive procedure language that includes complex control constructs and parameterized tasks. The approach has been implemented in the SPARK agent system and is being used to improve the performance of an operational intelligent assistant application.

Categories and Subject Descriptors

I.2.8 [**ARTIFICIAL INTELLIGENCE**]: Problem Solving, Control Methods, and Search—*Plan execution, formation, generation*

General Terms

Algorithms

Keywords

Belief-Desire-Intention agents, resource approximation, plan execution, procedural knowledge

1. INTRODUCTION

The challenge we address is to provide reasoning about projected resource usage within a hierarchical task execution framework to enable improved agent effectiveness. In particular, we seek to provide *guaranteed* bounds on resource consumption consisting of a lower bound that describes necessary resource requirements and an upper bound that describes possible resource requirements. The range between these bounds can stem from uncertainty about the current world, uncertainty about future actions and events, and ambiguity about the consumption requirements of actions.

Guaranteed resource bounds are essential for informing an agent's deliberations for task adoption and strategy selection because they enable it to make *conservative* commitments that eliminate the possibility of resource contention [15, 3]. With such bounds, an agent can avoid adopting tasks or performing actions that are resource infeasible given the agent's other commitments. In addition to being guaranteed, resource bounds should be *maximally informative*, meaning that they incorporate as much information as is available, in as timely a fashion as is possible. This requirement introduces the need to provide execution-time updates of bounds in response to both runtime events and the acquisition of relevant knowledge. By updating the bounds during execution, an agent can reassess its task set [11], for instance, to suspend or drop a task that consumes more of a resource than anticipated, or to adopt a task that becomes resource feasible.

Current techniques for agent resource projection are inadequate. Purely reactive agents perform no lookahead, acquiring knowledge about resource usage only as it occurs. Problems with resource contention are handled as they arise, which can be both costly and failure prone. Methods from the AI planning literature show how to derive guaranteed initial bounds on resource consumption [6, 2], but they require that estimates for all resource-consuming tasks be available before execution commences. Furthermore, they do not address the issue of updating resource projections at execution time. Previous work that combines lookahead and reactive techniques uses overly simplistic task models, reasoning about resource usage only for propositional tasks (i.e., not parameterized by variables) without branching or iteration [16].

Our approach is a middle ground focused on limited projection and runtime updates, in the spirit of [16]. The contributions are threefold. First, we adopt a *multi-fidelity* model of resource production and consumption that supports estimates whose accuracy can vary with the information available about tasks. One of the difficulties in estimating resource requirements accurately is that future tasks may depend on parameters that have not yet been instantiated. Our multi-fidelity models are valuable when bounds change significantly depending on how individual parameters are instantiated. For example, consider a task $(\mathsf{Flight}\ \langle\text{from}\rangle\langle\text{to}\rangle\langle\text{airline}\rangle\langle\text{class}\rangle)$. Without any knowledge of the starting point and destination, the bound on resource *money* would be very loose. If the $\langle\text{from}\rangle$ and $\langle\text{to}\rangle$ parameters are known, then the bound may be reduced substan-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profi t or commercial advantage and that copies bear this notice and the full citation on the fi rst page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specifi c permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

tially; similarly, the choice of airline could limit fares (e.g., budget vs. full-service airlines), as could the class (e.g., first vs. coach). Our multi-fidelity models allow different bounds to be specified for each of these variations, at the discretion of the domain modeler.

Second, we refine our initial resource projections as execution unfolds. These updates are performed at the end of the observation phase in the observe–decide–act cycle of the agent. They include not only pruning possible execution paths based on the current world and execution state, but also reasoning over and narrowing variable domains. Thus we ensure that information about resources derivable from execution-time events is made available to the agent's deliberation at the earliest possible stage. Third, we show how certain kinds of additional semantic information about tasks can be exploited to improve resource bounds.

Our work is grounded in an expressive procedure language that can be viewed as an abstraction of two sophisticated procedural reasoning frameworks for Belief-Desire-Intention (BDI) agents [13], namely, PRS [7] and SPARK [9]. The language supports a range of parameterized task types, including condition achievement, task performance, and waiting for or testing a condition. Complex procedures can be composed from these task types through the control constructs of sequencing, parallelism, conditional branching, iteration, and subgoaling.

Our resource projection methods have been realized in the SPARK system and applied to improve the effectiveness of resource usage within an intelligent personal assistant domain [14].

2. RESOURCES: MODELS AND USAGE

Resource projection requires models of production and consumption for the tasks that can be performed within a domain. In this section, we present a model for *deterministic* resource usage; the nondeterministic case is left for future work. An important distinction is between consumption of *replaceable* resources, which are returned by a task after it completes, and *non-replaceable* resources. For example, a conference room or a transferable software license is a replaceable resource, while money is non-replaceable. However, money is a resource that can be *produced*. We assume a simple temporal model in which a task consumes resources when it commences, and releases (or produces) resources when it (successfully) completes. We do not consider scheduling of tasks in this paper. Resources may be discrete or continuous, and sharable between tasks or not.

In certain cases, resource requirements can be modeled precisely in advance: when the amount of resource consumed or produced is either a constant (e.g., $15 to download an article from ACM, $100 to change an airline reservation), or a parameterized function of the task's parameters (e.g., (RegisterForConference$\langle X \rangle$) — fees vary across conferences but are known once registration opens). We call resource requirements that can be modeled precisely in advance *knowledge pre facto*. These resource requirements can be modeled as an n-ary function of some subset of the task's parameters, with $n = 0$ in the case of a constant resource requirement.

In other cases, resource consumption or production for a task may not be predictable in advance (e.g., the cost of an airline ticket). In such cases, only loose bounds on the resource requirements may be obtained unless explicit actions are taken to determine the requirements (e.g., by checking the cost online). Further, certain types of resource consumption cannot be measured precisely until the action completes (e.g., how much petrol to drive to the airport); we label such requirements *knowledge ad facto*. For other tasks, there may be some delay in determining the precise consumption level (e.g., posting of a charge in foreign funds to your credit card); we label such requirements *knowledge post facto*. Similarly, certain types of resource production may also be knowledge post facto.

We say that a task is *knowledge pre facto* (similarly, *ad* or *post facto*) if it is so for all its resources. In this paper we avoid the complications of knowledge post facto tasks by assuming that all consumption and production are knowledge pre or ad facto.

In our approach to modeling resource requirements, each task that directly consumes (produces) a resource includes an explicit effect for that consumption (production). A *resource model* consists of a set of *approximations* that provide strict lower and upper bounds on the amount of a particular resource R required for (produced by) a given task:

$$(\ \langle R \rangle\ \langle \mathit{appr}_0 \rangle \langle \mathit{appr}_1 \rangle \ldots \langle \mathit{appr}_m \rangle\ ) \tag{1}$$

The approximations $\langle \mathit{appr}_i \rangle$ are specified in terms of functions of some subset of the task parameters that specify lower and upper bounds on the resource consumption (production):

$$[\underline{f_i}(p_1, \ldots, p_{s_i}),\ \overline{f}_i(q_1, \ldots, q_{t_i})] \tag{2}$$

In cases where $\underline{f_i} = \overline{f}_i$, we say that the bound is *precise*. For simplicity of presentation, we assume that the functions used to compute lower and upper bounds share the same set of arguments (i.e., in (2) above, $s_i = t_i$ and $\forall j.\ p_j = q_j$). Projected resource estimates, as explained later, should take into account all possible approximations as they become computable, that is, as all the parameters to the approximation functions have been instantiated.

A typical usage of the construct (1) would be something like the following for a resource *money* consumed by an ArrangeCateringExternal task: $(\mathit{money}\ [(\times\ \$\mathsf{n}\ 10),\ (\times\ \$\mathsf{n}\ 20)]\ \ [120,\ 1000])$, where $\mathsf{n} is the number of people to be catered.

3. RESOURCE PROJECTION

3.1 Overview

Our resource projection framework provides *guaranteed* bounds on the use of resources for execution of a task, where the lower bound describes *necessary* and the upper bound describes *possible* resource requirements. Given the requirement of guaranteed bounds, it is sound for the projection to derive over-estimates of these bounds. Of course, excessively broad bounds may be of little or no value to an agent when deliberating over task feasibility.

Set against this background, our approach to resource projection is one of *limited* static lookahead prior to task execution, followed by dynamic updates during execution. The static projection considers the space of possible decompositions of the task, drawing on the set of plans available to the agent. The execution-time updates refine the static projections by incorporating information about execution decisions and changes to dynamic world state.

Motivated by the tasks and processes found in the personal assistant domain [11, 14], which are dominated by the use of resources to achieve tasks delegated by the user, we focus on resource consumption. Combining production and consumption is important but beyond the scope of this paper.

In the personal assistant domain, as in many others, failures can be classified as *minor* or *major*. Whereas many minor failures can be remedied by increasing resource consumption (e.g., the flight cost more than expected), major failure entails reconsideration of the task (e.g., there are no flights available and the meeting will need to be rescheduled).

Our resource projection framework does not explicitly account for task failure, focusing instead on resource needs for failure-free execution. Our expectation is that an agent that makes use of this framework would maintain a resource reserve from which it would

```
{defprocedure plan_group_visit
  cue:  [do: (planGroupVisit $visitors)]
  body: [sequence:
    [parallel:
      [do: (getPossibleDates $visitors $dates)]
      [do: (solicitInterested $visitors $people)]]
    [do: (scheduleDay $dates $visitors $people
                      $schedule)]
    [select:
       (ClearanceRequired $visitors)
          [sequence:
             [conclude: (InternalOnly $schedule)]
             [do: (applyForClearance $visitors)]]
       (True)  % otherwise
          [retract: (InternalOnly $schedule)]]
    [context: (Concat $visitors $people $all)
      parallel:
        [do: (informSchedule $all $schedule)]
        [do: (publicizeSeminar $schedule)]
        [do: (arrangeCatering $all $schedule)]]]}

{defprocedure apply_for_clearance
  cue: [do: (applyForClearance $visitors)]
  body: ...
  consumes: [(money (+ 100 (* 50 (length $visitors)))
                    [100 500])
             (hours 1)]
  requires: [(licenses 1)]}

{defprocedure publicize_seminar
  cue:  [do: (publicizeSeminar $schedule)]
  body: [sequence:
    [do: (publicizeSeminarInternal $schedule)]
    [do: (getPublicityCostEstimate $schedule $cost)]
    [select: (and ($cost < 50)
                  (not (InternalOnly $schedule)))
                [do: (publicizeSeminarExternal $schedule)]
             (True)  % otherwise
                [succeed:]]]}

{defprocedure arrange_catering_external
  cue:  [do: (arrangeCatering $everyone $schedule)]
  body: ...
  consumes: [(money [(* (length $everyone) 10)
                     (* (length $everyone) 20)]
                    [120 1000])
             (hours [1 3])]
  requires: [(licenses 1)]}
```

Figure 1: Plans for the group visit task

draw to remedy minor failures. For a major failure, a BDI agent will reconsider its commitments and decide upon a new course of action [13], which will in turn lead to the recomputation of projected resource bounds. For this reason, our projected bounds do not include resource requirements for addressing failures.

Our approach for projecting guaranteed resource bounds for a task presumes a task and plan model described in Sect. 3.2. The first step in the projection process is to generate a *task expansion tree* that encodes the space of possible decompositions for the task with respect to the agent's plan library (Sect. 3.3). For that structure, a *variable relationship graph* is then constructed that encodes relationships and constraints among parameters of the considered plans; constraint programming techniques are used on this structure to bound possible values of task parameters (Sect. 4). The resource projection itself involves determining resource requirements for leaf nodes in the task expansion tree, drawing on information from the variable relationship graph as to possible or actual bindings for task parameters, and then aggregating those values up through the task expansion tree to produce the overall resource summary for the root task (Sect. 3.4).

3.2 Tasks and Plans

To make concrete our description, we introduce an explicit task and plan model based loosely on the SPARK framework [9].

A *task* is a parameterized activity for the agent to execute. Tasks come in two types. *Simple* tasks (`achieve`, `do`, `wait`, `conclude`, `retract`, `succeed`, `fail`) are the more basic form of task. Of these, only `achieve` and `do` can consume resources. An `achieve` task is used to achieve a designated state condition, while a `do` task is used to perform a designated action. A simple task can be *primitive*, meaning that the agent's effectors can execute it directly, or *nonprimitive*. A nonprimitive task is executed by decomposing it into lower-level subtasks.

Composite tasks (`sequence`, `parallel`, `select`, `context`, `try`, `forall`, `while`) can be used to compose aggregate tasks from a set of *component* tasks in a functional manner. For example, the composite task `[sequence:` $T_1, \ldots, T_m$`]` denotes the sequential performance of the component tasks $T_1, \ldots, T_m$. The task type `select` provides for the choice among component tasks, conditioned on the outcome of a sequence of test conditions; `try` similarly supports branching, but conditioned on the successful performance of some test task; `context` provides for the testing of predicates for the purpose of binding variables. The semantics for the remaining task types follow standard interpretations (see [9]).

A *plan*, referred to as a *procedure* in SPARK terminology, describes one of possibly many ways of decomposing a nonprimitive task. (SPARK plans can also be used to respond to a change in the agent's beliefs — a capability not considered in this paper.) Each plan has a *cue* that indicates the event that triggers the plan, an optional *precondition* that states the conditions under which the plan is applicable, and a *body* that consists of a single task (either simple or composite) to be performed in response to the plan cue.

As noted above, resource models are associated with tasks. Given the definition of a plan body as a (possibly composite) task, we can similarly associate a resource model with a plan.

EXAMPLE 1. *Figure 1 shows extracts of plans for a group visit task in this language. Variables are prefixed by* `$`*. Resource approximations annotate the plans* `apply_for_clearance` *and* `arrange_catering_external`*. For instance, the latter has a resource approximation corresponding to* (*money* $[10n,\ 20n]$ $[120,\ 1000]$)*, where* n *is the length of the list* `$everyone`*. In this example, the resources money and hours are nonreplaceable (denoted by* `consumes` *in the resource approximation), while licenses are replaceable (denoted by* `requires`*).*

3.3 Task Expansion Tree

The *task expansion (TE) tree* is the principal data structure used for resource projection. Intuitively, a TE tree for a task T describes all possible resource-related (producing and consuming) subtasks that could be performed to accomplish T. It is obtained by considering how T can be expanded into subtasks via application of plans in the agent's plan library.

The root node of a TE tree corresponds to the task T. A TE tree contains both simple and composite task nodes. Each simple task node has a child node corresponding to every possible matching plan that could achieve the task; the child node denotes the task (either simple or composite) that defines the body of the associated plan. Thus, a simple task node has *or* semantics in that its children represent alternative means of achieving the task. We call the edge linking a simple task to one of its children a *plan edge*. Each composite task node has one or more child nodes corresponding to the component tasks for the composite task; links from a composite task node to its children are called *structural edges*.

The TE tree for a given task is constructed recursively. The child nodes of a simple task node are obtained by matching the possible plans, according to the task name and parameters (as far as they are instantiated), and current constraints on possible matchings. This matching considers static predicates only. The child nodes of a composite task node are obtained by decomposing the task into its component tasks. Construction halts on a given branch when the child node is either a primitive task or a task with a resource approximation. (Note that resource models can be specified for both primitive and nonprimitive tasks, at the discretion of the modeler.)

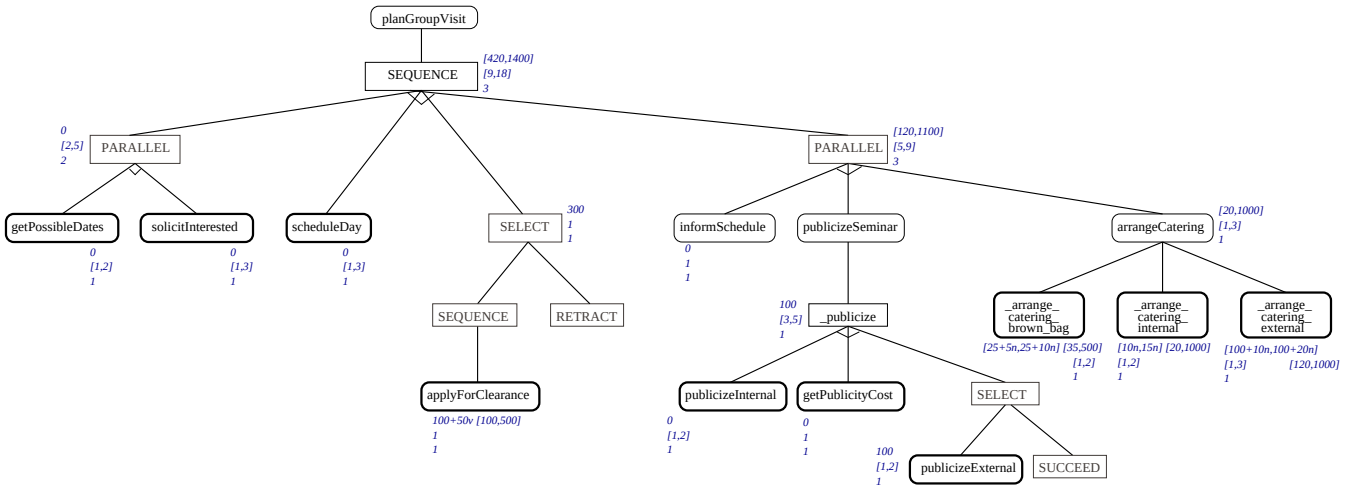


Figure 2: Task expansion tree for the task `planGroupVisit`

With the assumption of a nonrecursive library of plans, the tree and its construction are guaranteed to be finite.

Every node in a TE tree has a *resource summary* that provides for each resource the aggregate necessary and possible bounds on the consumption levels for that node and its descendants:

$$\langle\ (R_1\,[L_1,\,U_1]),\ \ldots,\ (R_m\,[L_m,\,U_m])\ \rangle \tag{3}$$

Here, L_i and U_i represent guaranteed lower and upper bounds, respectively, on the use of resource R_i for failure-free execution.

EXAMPLE 2. *The TE tree for the task* `planGroupVisit` *is displayed in Figure 2. Simple task nodes are depicted with rounded boxes and composite nodes with square boxes; bold indicates a task with a resource approximation model annotated. For simplicity, neither the parameters of tasks nor any* `conclude` *tasks are shown. Adjacent to each node is its resource approximation (for leaf nodes) or resource summary (for non-leaf nodes). These are shown as three lines for the resources money, hours, and licenses, respectively. Here, n represents the length of the list* `$everyone` *(recall Example 1), and v is the number of visitors, which is an input parameter to the root task.*

3.4 Computation of Resource Summaries

The computation of resource summaries for a TE tree involves first computing resource bounds for individual leaf nodes (Sect. 3.4.1), and then aggregating the bounds up to the root (Sect. 3.4.2). We assume that the domain model includes a *comprehensive* set of resource approximations for the domain, meaning that each primitive task that consumes resources has an associated resource approximation providing a guaranteed bound on its resource requirements.

3.4.1 Resource Bounds for Leaf Nodes

The resource summary for a leaf node T in a TE tree is determined by considering all resource approximations for the task of T (of the form (1)) whose parameters are fully instantiated. Let the bounds produced by this set of approximation functions be $\{[L_1,\,U_1],\ldots,[L_m,\,U_m]\}$. For a primitive task that does not consume resources (as reflected by the lack of a resource approximation in the domain model), this set is defined to be $\{[0,\,0]\}$. These bounds are strict, but represent different perspectives on the range of possible requirements; as such, they must be intersected as follows to produce the maximally refined bound for T:

$$\prod_{i=1,\ldots,m} [L_i,\,U_i] = [\max_{i\in\{1,\ldots,m\}} L_i,\ \min_{i\in\{1,\ldots,m\}} U_i] \tag{4}$$

It is not always necessary that a parameter to a resource approximation $[\underline{f},\,\overline{f}]$ be bound in order for the approximation to be used. Valid (but perhaps looser) bounds can be computed by inference over the variable's current domain, using techniques from constraint programming [1, 5]. In particular, in situations where the domain for the variable is restricted to a finite enumerable set $D = \{d_1,\ldots,d_k\}$, reasoning by cases can be applied to D yielding the bounds $L_i = \min_{d\in D} \underline{f}(d)$ and $U_i = \max_{d\in D} \overline{f}(d)$ on the individual approximations. The same enumeration technique applies to the Cartesian product of the domains of a set of finite domain variables, although care must be taken to consider only domains of tractable size.

3.4.2 Resource Aggregation for Non-Leaf Nodes

Resource aggregation computes the resource summary (3) for a non-leaf node T in a TE tree based on the summaries of its children. We assume no constraints between resources, that is, each resource is consumed and produced independently of others. In this case, it suffices to consider each resource in turn.

Consider first a simple task node T with child nodes $T_1,\ldots,T_m$ where T_i has resource summary $[L_i,\,U_i]$ for a given resource. As noted above, our model presumes that of the simple tasks, only `achieve` and `do` can impact resource usage. Recall that each child of a simple task node corresponds to the body of a possible plan for achieving the task. At runtime, exactly one of the T_i will be chosen for execution. Thus the resource summary for T consists of a lower bound that is the minimum of the L_i and an upper bound that is the maximum of the U_i, i.e., $[\min L_i,\,\max U_i]$.

Now consider a composite task node T with component task nodes $T_1,\ldots,T_m$; again let the child nodes have summaries $[L_i,\,U_i]$. Replaceable and non-replaceable resources differ in their aggregation. Adapting [16], we define the composition operator:

$$x \otimes y = \begin{cases} \max(x,y) & \text{for replaceable resources} \\ x+y & \text{for non-replaceable resources} \end{cases}$$

Aggregation for different types of composite nodes is defined as follows. We do not consider `context` tasks as we assume that condition testing does not impact resources.

- `[sequence:` $T_1,\ldots,T_m$`]`. T_i are tasks to be executed sequentially; the bound is $[\bigotimes L_i,\ \bigotimes U_i]$.

- `[parallel:` $T_1, \ldots, T_m$`]`. T_i are tasks to be executed in parallel; the bound is $[\sum L_i,\ \sum U_i]$.
- `[select:` $C_1T_1, \ldots, C_mT_m$`]`. The first T_i is chosen such that its condition C_i is satisfied. Since the C_i have no resource impact, the bound is $[\min L_i,\ \max U_i]$.
- `[wait:` $C\ T'$`]`. Task T' is executed once the condition C becomes true. Since we do not consider timing of tasks in this paper, and condition testing has no resource impact, the bound is simply that of T'.
- `[try:` $Q_1T_1, \ldots, Q_mT_m$`]`. The tasks Q_i are performed in order until some Q_s succeeds; the corresponding task T_s is then executed. Assuming that tasks fail at their end then a two-condition `try` statement expands to one of three sequences: $Q_1\ T_1$; $\mathbf{Q_1}\ Q_2\ T_2$; or $\mathbf{Q_1}\ \mathbf{Q_2}$, where bold denotes failure. The bound is the minimum and maximum of these possible sequences; for example, the lower bound is $\min(L_{Q_1} \otimes L_{T_1}, L_{Q_1} \otimes L_{Q_2} \otimes L_{T_2}, L_{Q_1} \otimes L_{Q_2})$.

Projection of replaceable resources through the iteration constructs `while` and `forall` is straightforward: if the projected resource estimate for a single execution of the loop body is $[L,\ U]$, then $[0,\ U]$ constitutes a guaranteed bound. Projection of non-replaceable resources for these constructs presents a challenge in that the number of iterations may depend on conditions or parameters that cannot be determined until runtime: for instance, looping over the number of people who reply to an invitation, or looping until a reply to an email request is generated.

The construct `[while:` $C\ T$`]` is particularly problematic, given the dependence on the dynamics of the termination condition C. Bounding the number of iterations requires both an upper bound d on the duration for which C is anticipated to hold, and a lower bound t on the time required to complete an iteration of T. If the projected resource estimate for a single execution of T is $[L,\ U]$, then $[0,\ U \times d/t]$ constitutes a guaranteed bound for the iteration.

Iteration based on the construct `[forall:` $x\ C(x)\ T(x)$`]` can be bounded in the following cases in which the exact number of iterations, n, can be determined. This occurs when (a) the quantification variable belongs to an enumerated type (e.g., iterate over people in the organization), (b) the iteration is defined over a concrete structure for which the number of elements can be ascertained (e.g., over a list), or (c) the scope of quantification is defined by a static predicate with bounded, known extent (e.g., purchase a flight for every member of the team). In these cases, if the projected resource estimate for a single execution of T is $[L,\ U]$, then $[L \times n,\ U \times n]$ constitutes a guaranteed bound for the iteration for a non-replaceable resource. By contrast, in the case when C is a dynamic predicate, precise determination of the number of iterations is not possible. If a bound v on the number of values for which C holds can be determined (e.g., register everyone who has a paper accepted), then $[0,\ U \times v]$ provides a guaranteed bound.

EXAMPLE 3. *Consider the task node* `arrangeCatering` *in Figure 2. Three possible plans could match the cue for this task: by brown bag, internal caterer, or external caterer. The plan for the last of these was shown in Figure 1. All three plans have resource approximations annotated as shown in Figure 2. For resource money, the approximations are functions of n, the total number of people to be catered, as before. For resource hours, the approximations are a single bound, and for licenses they are specified precisely. Since n is unknown prior to execution, the only computable approximation for money is the last, for example,* $[120,\ 1000]$ *for external catering. The resource aggregation produces the summary shown for node* `arrangeCatering` *from its three children:* $\langle$ (*money* $[20,\ 1000]$), (*hours* $[1,\ 3]$), (*licenses* 1) $\rangle$. *This information is propagated up the TE tree to its parent, the* `PARALLEL` *node. In the example, the number of visitors is a ground parameter to the root task: $v = 4$. Eventually, the resource summary derived for the root task node is* $\langle$ (*money* $[420,\ 1400]$), (*hours* $[9,\ 18]$), (*licenses* 3) $\rangle$.

By induction over the TE tree, the resource summary computed at the root node is a guaranteed enclosure of the actual resource consumption by the root task:

PROPOSITION 1. *Let T be a task and $[L,\ U]$ be the resource summary for resource R computed by the above algorithm on the TE tree rooted at T. Given that primitive tasks consume R within their resource approximations, then the amount of R consumed by T lies within $[L,\ U]$.* □

4. VARIABLE RELATIONSHIP GRAPH

Having described how to construct the TE tree and perform resource projection over it, we now describe the construction and use of the *variable relationship (VR) graph* for tracking relationships and constraints on variables. A VR graph enables both the propagation of parameter values that assist with computing resource bounds on tasks, and the pruning of impossible branches of a TE tree.

4.1 VR Graph Construction

We call the nodes of a VR graph *vertices*, to distinguish them from the nodes of a TE tree. Each vertex corresponds to a variable in a node of a TE tree. The (hyper-)edges of the VR graph correspond to relations between the variables.[1] Formally, the VR graph for a TE tree is a 4-tuple $\langle \mathcal{V}, \mathcal{D}, \mathcal{C}, \mathcal{E} \rangle$ where: $\mathcal{V}$ is a set of variables; $\mathcal{D}$ is a set of domains, one for each variable; $\mathcal{C}$ is a set of relations over $\mathcal{D}$; and $\mathcal{E}$ is a set of additional *possibility* relations, explained below. Thus $\mathcal{C}$ and $\mathcal{E}$ are two sets of constraints and $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \cup \mathcal{E} \rangle$ forms a classical *constraint satisfaction problem* (CSP) [1, 5].

The variables of a VR graph are divided into disjoint sets called *partitions*. Each partition corresponds to a different execution-time choice. These choices derive from two sources: (a) selection among alternative plan instances for a simple task (i.e., sibling plan edges in the TE tree), and (b) conditional branching via the `select` and `try` constructs. For each such choice, a partition is created to localize constraints and variables specific to that choice.

The two types of constraints, $\mathcal{C}$ and $\mathcal{E}$, represent relationships of different types between the variables. The edges in $\mathcal{C}$ encode constraints among variables *within* partitions; these constraints are derived from the predicates and functions used in the corresponding plans. In accord with terminology from the AI planning community, we say that two variables linked by an equality relation *codesignate* with each other. Second, the possibility relations in $\mathcal{E}$ link partitions. Each $\mathcal{E}$ constraint relates a variable in one partition to variables with which it may possibly codesignate in other partitions (depending upon which partition is chosen at execution time).

Consider a set of partitions created to represent selection among alternative plan edges for a simple task node T in the TE tree. For each task parameter `$x`, an edge in $\mathcal{E}$ represents the relationship from a vertex for `$x` to the vertices for the equivalent variables in the child tasks, as linked by the plan cue on the corresponding plan edge. For example, suppose T has two child nodes T_1 and T_2, with corresponding parameter variables `$x1` and `$x2`. The VR graph vertex for `$x` is related to each of `$x1` and `$x2` by a possibility relation. This describes that exactly one of `$x1` and `$x2` will codesignate with `$x` in the future, but which one is unknown at this time. This partitioning allows inference for different plan choices in the TE tree to proceed without cross-contamination from

[1] For simplicity, we assume (by introducing new variables, preconditions, and `context` tests) that the parameters in the cue of each plan and the parameters of the tasks in the body of a plan are all variables.

Algorithm 1 Pseudocode for VR graph constraint propagation (lines 1–10) and resource summary updating (lines 11–15)

```
 1: repeat
 2:    Choose a partition p such that V(p) is non-empty
 3:    Add the node N_TE(p) to Q
 4:    repeat
 5:       Pick a variable v ∈ V(p) and remove it from V(p)
 6:       for each constraint C involving v do
 7:          Update D(v') of each variable v' constrained by C
 8:          If any D(v') has changed, add v' to V(p_v)
 9:    until V(p) is empty
10: until V(p) is empty for all partitions
11: repeat
12:    Pick a node N ∈ Q and remove it from Q
13:    Update resource summary of N from the changed variable domains
14:    If resource summary of N changed, add the parent of N to Q
15: until Q is empty
```

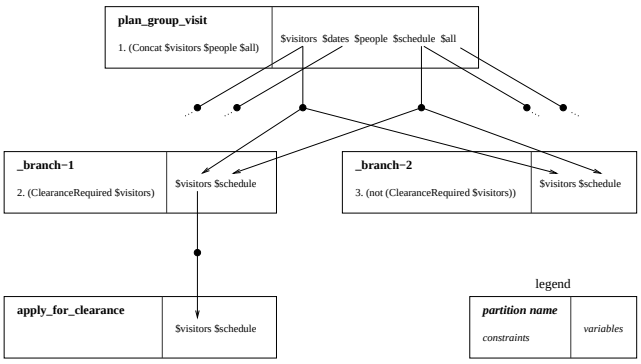


Figure 3: Variable relationship graph fragment

lower to upper levels; it also allows inferences from higher to lower levels, that is, from T to T_1 and T_2, for instance if the domain of `$x` is narrowed. If there is only one possible branch, then an equality rather than possibility relation links the variable vertices.

Similarly, each conditional branch (i.e., the `select` and `try` constructs) leads to a distinct partition with separate variables. For example, consider a plan containing the composite task `[select:` $(C_1$ `$x)` `[do:` $(T_1$ `$y)]` $(C_2$ `$x)` `[do:` $(T_2$ `$y)]]`. Separate partitions and new copies of the variables `$x` and `$y` are created for each branch. Possibility relations relate `$x` and `$y` to the corresponding branch variables $\texttt{\$x}_1$, $\texttt{\$y}_1$, $\texttt{\$x}_2$, and $\texttt{\$y}_2$. The partition for the first branch has the constraint $(C_1\ \texttt{\$x}_1)$; the partition for the second has the constraints $(C_2\ \texttt{\$x}_2)$ and $\neg(C_1\ \texttt{\$x}_1)$.

EXAMPLE 4. *Figure 3 shows part of the VR graph for the TE tree in Figure 2. The partition* `plan_group_visit` *contains the variables in the plan for the corresponding node in the TE tree, and a constraint derived from the* `context` *test in the plan. The* `select` *construct introduces two child partitions corresponding to the two branches, with constraints derived from the* `select` *conditions. Possibility relations link corresponding variables in parent and child partitions.*

The initial domain $D(v)$ of a variable v is derived from the type of v. We next describe how variable domains, and consequently resource summaries, are refined.

4.2 Propagation and TE Tree Pruning

Constraint propagation updates domains for variables in the VR graph, drawing both on information that is known when the TE tree is created, such as from variable domains and initial parameters, and information acquired during execution, such as when the domain of a variable changes or the graph is structurally modified. These updates in turn enable resource summaries to be refined as variables are instantiated or their domains are narrowed.

We perform a two-level constraint propagation that exploits the partitioned structure of the VR graph. The narrowing of a domain of a variable is propagated through the constraints of the partition containing the vertex for that variable, possibly resulting in the narrowing of other variable domains. Once a quiescent state is reached within a partition, the domain narrowings are propagated across the possibility relationships to neighboring partitions and the process is repeated. The domains are passed down to child partitions, but are merged with domains for the corresponding parameters in alternative partitions before being passed up.

Algorithm 1 gives the pseudocode for the VR graph constraint propagation and the subsequent updates to the resource summaries in the TE tree. In the pseudocode, let $V(p)$ be the set of variables in partition p that have changed and let p_v be the partition of variable v. Let $N_{\text{TE}}(p)$ be the TE tree node from which partition p is derived. Let Q be a queue of nodes in the TE tree, initially empty.

Lines 1–10 propagate domain changes through the VR graph constraints; lines 11–15 then propagate the effects in the TE tree. The loop starting at line 6 considers constraints in $\mathcal{C}$, which may affect the domains of related variables within partition p, and constraints in $\mathcal{E}$. Consider a hyperedge from variable v to variables $v_1, \ldots v_n$ representing a constraint in $\mathcal{E}$. If $D'(v)$ represents the new value of the domain of v after propagation, then $D'(v) = D(v) \cap \bigcup_i D(v_i)$ and $D'(v_i) = D(v_i) \cap D'(v)$.[2]

The narrowing of a variable domain to the empty set indicates that the containing partition is inconsistent. When propagated, this narrowing renders all descendant partitions inconsistent, and may make the parent inconsistent as well. A parent partition is made inconsistent if and only if all its child partitions are inconsistent.

Inconsistent partitions can be pruned from the VR graph; they can also be used to avoid expanding branches of the TE tree that are not viable. Moreover, if the constraints in one partition of the VR graph become inconsistent, given knowledge about possible variable bindings from the parent partition, the corresponding plan branch from the TE tree can be pruned.[3]

EXAMPLE 5. *Consider the VR graph in Figure 3. Since the top* `$visitors` *variable is bound to a specific list (in this example, of length four), the domain (not shown) associated with the variable is narrowed to a single value, that is, to that list. This narrowing is propagated to the domains of the* `$visitors` *variables in the branch partitions and then down to* `apply_for_clearance`*. The constraint associated with one of the branch partitions, say* `_branch_2`*, is now violated, leading to that branch being pruned. In that case, the unique value for* `$visitors` *allows refinement of the resource bounds on* `apply_for_clearance` *from* $[100, 500]$ *to the precise value* 300*, which is then propagated upward in the TE tree.*

Many agent and planning systems distinguish *static* predicates, which do not change over time, from *dynamic* predicates (i.e., fluents). Static predicates, such as equality and list membership, can be exploited to constrain variables in the VR graph, but because the truth value of dynamic predicates cannot be known until execution time, their usefulness for pruning the VR graph is limited. Hence, we do not reason about future values of dynamic predicates.[4]

[2] Note that for line 12 it is most efficient to handle all descendants of a node before handling the node itself. This means that Q should be partially ordered by $N_1 \prec N_2$ iff N_1 is a descendant of N_2 in the TE tree. Q can be kept sorted by inserting nodes in the correct place in lines 3 and 14.

[3] Note that this pruning may be more aggressive than in plan selection for an ordinary BDI agent: it may be possible using the VR graph to deduce that a plan must later fail due to constraint violations even if its preconditions are satisfied.

[4] Prior work in the planning community has investigated the use of the modal truth cri-

The partitioning of the CSP does not detract from the soundness or completeness of the inference, although it is worth noting that the agent's inference depends on the power of the constraint propagation techniques it has available, which we denote *Prop*. By standard results on *chaotic iteration* over the combined CSP [1]:

PROPOSITION 2. *Two-level constraint propagation over the VR graph $\langle \mathcal{V}, \mathcal{D}, \mathcal{C}, \mathcal{E} \rangle$, using Prop at each stage, achieves the identical outcome as Prop on the combined CSP $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \cup \mathcal{E} \rangle$.* □

5. RESPONDING TO EXECUTION EVENTS

A TE tree captures the anticipated resource usage for a task at a given instant in time. We have described how to construct TE trees prior to the start of task execution. In order to produce maximally informative bounds in the resource summaries, it is necessary to update the summaries at execution time as choices are made and events occur that could impact resource usage.

A typical BDI agent maintains its current intentions in data structures analogous to subtrees of the TE trees [13, 9]. As plans execute successfully, variables are bound to values, and these subtrees grow through the addition of task nodes. The VR graph tracks these changes by incorporating the variable bindings and pruning both the nodes in the TE trees and the partitions in the VR graphs that correspond to the choices *not* selected. In each of these cases, the additional information is propagated through the VR graphs and updated resource bounds are aggregated up the TE trees.

EXAMPLE 6. *Consider the point at which the* `arrangeCatering` *task of Figure 2 is to be executed. When the plan corresponding to the* `arrange_catering_external` *node is selected, the sibling nodes can be pruned, leading to a more precise resource bounds estimate. For the resource money, for instance, supposing the value of n is not yet known, the bound at the node* `arrange_catering` *becomes* $[120, 1000]$.

The effects of successful execution are *monotonic* in that they lead only to refinement, not expansion, of previously computed resource bounds. This monotonicity results because (a) the operations do not invalidate previous variable bindings or constraints, and prune only those parts of the VR graph that are certain not to impact resource consumption, and (b) elimination of options can only narrow, not expand, bounds. In contrast, task failures can lead to nonmonotonic changes to resource bounds, since an agent must reconsider its plans and may potentially prune failed subtrees in the TE tree in favor of previously rejected alternatives. Such failures require the recomputation of the resource bounds for the impacted portions of the TE tree.[5]

6. EXPLOITING SEMANTIC KNOWLEDGE

This section presents two further types of execution-time refinements to resource projections. Both leverage additional semantic knowledge about relationships among task parameters.

6.1 Active Solicitation of Requirements

In certain situations, an agent may actively gather information to provide explicit estimates for the resource requirements of a task. For example, the cost of a reception would generally be solicited as part of the process of organizing a workshop. We call a task that produces such an upper estimate a *resource estimation task*.

When a resource estimation task produces an estimate e for a given task T, the prior estimate $[l, u]$ for T can be refined to $[l, e]$. In some cases, it may even be possible to obtain precise values, as is the case when booking a hotel. We call a task that produces a precise characterization of resource consumption a *resource bid task*. When a resource bid task produces a bid b for T, the bounds for T can be set to $[b, b]$.

In certain cases, it may be appropriate for the agent to obtain multiple estimates for a resource-consuming task. If all such estimates $E = \{e_1, \ldots, e_m\}$ can be obtained together, then the resource bounds for the task can be set to $[\min E, \max E]$.[6]

6.2 Selection and Narrowing

The relationships among variables tracked in the VR graphs derive from codesignation constraints among variables both within and across plans (the latter arise from plan application to a task). Further inference can be enabled by augmenting the VR graphs to exploit additional types of inter-variable relationships. Here, we describe two common task idioms that produce such relationships, and indicate how a VR graph can be extended to reflect those relationships and support additional execution-time refinements of resource bounds.

The first idiom focuses on *selection* tasks. Consider a task (Select `$c` `$s`), where `$c` represents a set of values under consideration for some role, and where execution of the task results in the binding of the variable `$s` to a value for that role. The binding of `$c` to a list of values restricts the domain of values for `$s` to the same list. Although the binding for `$s` is not yet known, reasoning by cases can be applied over the possibilities, *before* the task (Select `$c` `$s`) is performed, to any resource approximation for a downstream resource-consuming task that depends on a parameter linked from `$s`. However, such a restriction on the domain of `$s` cannot be exploited within the VR graph as formulated, since no explicit constraint connects the variables `$c` and `$s`.

The second idiom focuses on tasks that perform some kind of *narrowing* operation that reduces the domain of a variable that is used subsequently to provide a resource approximation for a task. That is, the task reduces some *source* domain to a smaller *result* domain. For instance, in the flight purchase example given in Sect. 1, one narrowing operation would be the decision to consider only budget airlines, which precludes tickets from full-service carriers and thus refines the set of candidate flights.

To enable these forms of projective reasoning, we extend a VR graph as follows. We require semantic annotations that identify *selection* tasks along with their *candidate* and *selection* parameters, and *narrowing* tasks with their *source* and *result* parameters. Let (Member `$e` `$s`) define the constraint `$e` $\in$ `$s` and (Subset `$s1` `$s2`) define the constraint `$s1` $\subseteq$ `$s2`. We extend the constraints $\mathcal{C}$ of the VR graph as follows:

- If $T(x_1, \ldots, x_n)$ is a selection task with candidate parameter x_c and selection parameter x_s, then add (Member x_c x_s)
- If $T(x_1, \ldots, x_n)$ is a narrowing task with source parameter x_s and result parameter x_r, then add (Subset x_r x_s)

Exploitation of these additional VR graph links involves a straightforward generalization of the propagation algorithm of Sect. 4.2; in the interest of space, we forego a detailed description here.

teria for predicting the future state of dynamic predicates that model conditions under control of the system [2]. Many dynamic predicates, however, represent exogenous domain properties that lie beyond the agent's control (e.g., temperature, time of arrival of a flight) and so cannot be predicted using those methods.

[5] Thangarajah et al. [16] point out that, depending on the intended use of resource information, it may be possible to update data structures such as TE trees lazily, i.e. mark them as 'dirty' when change occurs, but not compute the new values until required.

[6] When estimates are solicited independently, there is an operational issue of knowing when the last estimate has been obtained so that the revised bound can be computed. In particular, if bounds are computed prematurely they may be overly narrow, thus violating our requirement for guaranteed bounds. Static analysis to determine the last estimation task is made difficult by the temporal unpredictability of asynchronous threads and loops, and the possibility of conditional branching.

7. DISCUSSION AND RELATED WORK

Our task expansion tree and associated aggregation of resource summaries are similar in spirit to the *goal-plan tree* of [16]. The latter is composed of alternating layers of goal nodes (analogous to our simple task nodes) and *plan nodes*; their plans are limited to sequential or parallel execution of propositional tasks. The same authors reason with a failure model whereby, if a plan T_1 fails for a task T, then every other possible plan $T_2, \ldots, T_m$ to achieve T is attempted in turn, until one succeeds or all have failed (when T is said to have failed). This failure model is reflected in their construction of the goal-plan tree, but is overly conservative and so may limit task throughput (and, hence, agent effectiveness). Last, the focus of [16] and their related work is to determine when tasks might be in potential conflict, and to resolve conflicts where possible by scheduling the task executions.

We have described a limited form of lookahead, with respect to resources. de Silva and Padgham [4] go further in deliberately incorporating full AI planning over possible plans (recipes) within an agent architecture, by the action of invoking a hierarchical planning system. Techniques for incorporating resource feasibility testing into hierarchical planning have been investigated previously [6, 2]. These approaches perform a kind of static analysis similar to that performed by our aggregation over the task expansion tree, although they support replaceable resources and resource production through the incorporation of a modal truth criteria analysis. The work assumes a single known value or single bound for the resource requirements of individual tasks, in contrast to our multi-fidelity approach. Furthermore, they do not address the issue of execution-time refinement of aggregate resource bounds.

Resource estimation is well studied in scheduling and temporal reasoning. For instance, Laborie [8] describes propagation of resource constraints in constraint-based scheduling, while Muscettola [10] computes an *envelope* encompassing the predicted evolution of resource levels over time. The emphasis in scheduling is to compute ahead of execution a feasible schedule that guarantees no resource conflicts. In contrast, detecting plan failure and recovering from it is an integral part of a reactive agent. Thus our projection of resources need not be complete; it has value even when partial since any sound inference is informative to the agent. Nonetheless, techniques particularly from constraint-based scheduling hold potential for extending our work to consider predicting resource evolution over time. Such work also points to the scheduling of tasks to avoid possible resource conflicts, as also considered by [16].

8. CONCLUSION

The framework for resource projection presented here provides a pragmatic middle ground, combining the static computation and dynamic refinement of guaranteed bounds on projected resource usage by a BDI agent. Our framework is distinguished by its use of parameterized multi-fidelity models of expected resource consumption, its focus on maximally informative resource bounds, and an expressive plan language that includes parameterized task types.

We have implemented our resource projection framework in the SPARK system, and are applying it to real-world domains derived from the CALO cognitive assistant project [14]. These domains include a full version of the group visit example in this paper, domains describing purchase of office equipment, and registration and travel to a conference. Construction of the TE trees and associate VR graphs, constraint propagation on the variables, and initial pruning of the trees are all performed with a few hundred milliseconds of runtime; the overhead is in line with required performance of CALO as a mixed-initiative task execution assistant. In addition to the minimal overhead of the static resource projection, the dynamic updates employ CSP propagation techniques of low-order polynomial complexity [5], and thus scale effectively.

Our work enables improved task throughput and execution quality, with reduced failure. It is straightforward to construct scenarios in which our approach will substantially outperform existing techniques. However, a truly meaningful evaluation of benefits will require analysis of extended agent operations within a real (or realistic simulated) environment. We will be collecting such data as our system is deployed within the CALO user community.

The work presented here focuses on guaranteed bounds for resource consumption. An interesting next step would be to develop a probabilistic counterpart to this framework that establishes the *likelihood* of resource contention by incorporating stochastic models of event likelihood and task success, building on a framework such as the probabilistic representation for process models of [12].

As noted earlier, future work is to consider a broader range of task resource consequences, including knowledge post facto costs and nondeterministic resource consumption. In addition, while much of the work described in this paper applies to production as well as consumption (on which we have focused), resource production and its interaction with consumption requires a model with temporal extent, by which the agent can reason on the relative timing and interleaving of production and consumption, including the return of replaceable resources.

Acknowledgments. We thank the reviewers for their comments. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010.

9. REFERENCES

[1] K. R. Apt. The essence of constraint propagation. *Theoretical Computer Science*, 221(1–2):179–210, 1999.

[2] B. J. Clement, A. C. Barrett, G. R. Rabideau, and E. H. Durfee. Using abstraction in planning and scheduling. In *Proc. of ECP'01*, 2001.

[3] M. Dastani and L. van der Torre. Specifying the merging of desires into goals in the context of beliefs. In *Proc. of EurAsia ICT '02*, 2002.

[4] L. P. de Silva and L. Padgham. Planning on demand in BDI systems. In *Proc. of ICAPS'05 Poster Session*, pages 37–40, June 2005.

[5] R. Dechter. *Constraint Processing*. Morgan Kaufmann, San Francisco, CA, May 2003.

[6] B. Drabble and A. Tate. The use of optimistic and pessimistic resource profi les to inform search in an activity based planner. In *Proc. of AIPS'94*, pages 243–248, Chicago, IL, June 1994.

[7] M. P. Georgeff and A. L. Lansky. Procedural knowledge. *Proc. of the IEEE*, 74(10):1383–1398, 1986.

[8] P. Laborie. Algorithms for propagating resource constraints in AI planning and scheduling. *Artificial Intelligence*, 143(2), 2003.

[9] D. Morley and K. Myers. The SPARK agent framework. In *Proc. of AAMAS'04*, pages 714–721, New York, NY, July 2004.

[10] N. Muscettola. Computing the envelope for stepwise-constant resource allocations. In *Proc. of CP'02*, pages 139–154, Sept. 2002.

[11] K. L. Myers and N. Yorke-Smith. A cognitive framework for delegation to an assistive user agent. In *AAAI 2005 Fall Symposium on Mixed-Initiative Problem-Solving Assistants*, Nov. 2005.

[12] A. Pfeffer. Functional specifi cation of probabilistic process models. In *Proc. of AAAI'05*, pages 663–669, Pittsburgh, PA, July 2005.

[13] A. S. Rao and M. P. Georgeff. Modeling agents within a BDI-architecture. In *Proc. of KR'91*, pages 473–484, 1991.

[14] SRI International. CALO: Cognitive Assistant that Learns and Organizes. `www.ai.sri.com/project/CALO`, Mar. 2005.

[15] J. Thangarajah, L. Padgham, and J. Harland. Representing and reasoning for goals in BDI agents. In *Proc. of the Australasian Conference on Computer Science*, Melbourne, Australia, Jan. 2002.

[16] J. Thangarajah, M. Winikoff, L. Padgham, and K. Fischer. Avoiding resource conflicts in intelligent agents. In *Proc. of ECAI-02*, 2002.

How Equitable is Rational Negotiation?

Sylvia Estivie[1], Yann Chevaleyre[1], Ulle Endriss[2] and Nicolas Maudet[1]
[1]LAMSADE, Université Paris-Dauphine, France
[2]ILLC, University of Amsterdam, The Netherlands
[1]{estivie,chevaley,maudet}@lamsade.dauphine.fr, [2]ulle@illc..uva.nl

ABSTRACT

Notions of fairness have recently received increased attention in the context of resource allocation problems, pushed by diverse applications where not only pure utilitarian efficiency is sought. In this paper, we study a framework where allocations of goods result from distributed negotiation conducted by autonomous agents implementing very simple deals. Assuming that these agents are strictly self-interested, we investigate how equitable the outcomes of such negotiation processes are. We first discuss a number of methodological issues raised by this study, pertaining in particular to the design of suitable payment functions as a means of distributing the social surplus generated by a deal amongst the participating agents. By running different experiments, we finally identify conditions favouring equitable outcomes.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*; J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Economics, Theory, Experimentation

Keywords

Multiagent resource allocation, Negotiation, Social welfare, Fair division

1. INTRODUCTION

In many applications of multiagent systems it is important to achieve a *fair*, or *equitable*, allocation of the available resources amongst the agents in the system. However, this fairness requirement will often compete with the *rational* interests of individual agents. If agents are allowed to freely negotiate their preferred allocations of resources, the emerging allocations will typically be good on average, but give a rather low payoff to an unfortunate minority. In this paper, we investigate to what degree it is possible to reconcile rationality and equitability requirements. We do this by running a number of simulations where agents negotiate in a rational manner and where we track the evolution of allocations in view of their equitability.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Negotiation Problems

Take a set of (indivisible) goods $\mathcal{R}$ and a set of agents $\mathcal{A}$. A resource *allocation* is a partitioning of the resources of the set $\mathcal{R}$ amongst the n agents in $\mathcal{A}$ (that is, every resource has to be allocated to a unique agent). As an example, allocation A, defined through $A(i) = \{r_1\}$ and $A(j) = \{r_2, r_3\}$, would allocate resource r_1 to agent i, while resources r_2 and r_3 would be owned by agent j. Each agent has individual preferences regarding the bundles of resources it may hold. We are going to model the preferences of agents by means of *utility functions* mapping bundles of resources to real numbers. Assuming that agents are only concerned with resources they personally own, we will use the abbreviation $u_i(A)$ for $u_i(A(i))$, representing the utility value assigned by i to the bundle it holds for allocation A.

Definition 1 (Negotiation problems). *A negotiation problem is a tuple $\mathcal{P} = \langle \mathcal{R}, \mathcal{A}, \mathcal{U}, A_0 \rangle$, where*

- *$\mathcal{R}$ is a finite set of indivisible resources;*
- *$\mathcal{A} = \{1, \ldots, n\}$ is a finite set of agents ($n \geq 2$);*
- *$\mathcal{U} = \langle u_1, \ldots, u_n \rangle$ is a vector of utility functions, such that for all $i \in \mathcal{A}$, u_i is a mapping from $2^{\mathcal{R}}$ to $\mathbb{R}$;*
- *$A_0 : \mathcal{A} \to 2^{\mathcal{R}}$ is an initial allocation.*

From a designer's perspective, we will be interested in assessing the well-being of the whole society, or its *social welfare* [1], which is often defined as the sum of utilities of all the agents:

$$sw_u(A) = \sum_{i \in \mathcal{A}} u_i(A) \quad (1)$$

This is the *utilitarian* definition of social welfare (sw_u), but other criteria of assessment exist as well. In the next section, for instance, we are going to introduce the notion of *egalitarian social welfare*, which is a measure for the equitability of a given allocation. In this paper, we focus in general on equitability criteria. To decide how to (best) allocate the resources amongst the agents in a system is to solve a resource allocation problem.

Resource allocation problems are typically addressed from a centralized point of view (this is the case for instance in combinatorial auctions [4]). When no central authority is available, or when the computational burden would overwhelm a single agent, this approach is not well suited. A different approach, advocated by several authors [15, 7, 5], consists of distributing this process.

Distributed Approach

In the distributed perspective, instead of trying to centrally compute the optimal allocation, we shall progress in a stepwise manner towards an optimum. To do so, agents may agree on a *deal* to exchange some of the resources they possess. It transforms the current allocation of resources A into a new allocation A'; that is, we can define a deal as a pair $\delta = (A, A')$ of allocations (with $A \neq A'$). When such a deal improves the utilitarian social welfare (*i.e.* when $sw_u(A') > sw_u(A)$), we say that it is *socially beneficial*. Note that a single deal may involve the displacement of any number of resources between any number of agents. This level of generality is hardly realistic in practice. Sandholm [15] has proposed a typology of different types of deals, such as *swap deals* involving an exchange of single resources between two agents or *cluster deals* involving the transfer of a set of items from one agent to another. In this paper, we shall only consider the simplest type of deals (1-deals), *i.e.* those involving only a single resource (and thereby only two agents). The set of agents involved in the deal δ will be denoted as $\mathcal{A}^\delta$.

The above is a condition on the *structure* of a deal. Other conditions relate to the *acceptability* of a deal to a given agent. We assume that agents are *rational* in the sense of aiming to maximise their individual welfare. Furthermore, agents are assumed to be *myopic*. This means that agents will not accept deals that would reduce their level of welfare, not even temporarily, because they are either not sufficiently able to plan ahead or not willing to take the associated risk (see also [15] for a justification of such an agent model). Also agents will not behave strategically by, for instance, postponing a rational deal in the hope of finding an even better opportunity. We will, however, permit agents to enhance deals with *monetary side payments*, in order to compensate for a possible loss a utility. This is modelled using a so-called *payment function* $p : \mathcal{A} \to \mathbb{R}$ satisfying $\sum_{i \in \mathcal{A}} p(i) = 0$. A positive value $p(i)$ indicates that agent i *pays* money, while a negative value means that that agent *receives money*. In summary, the following rationality criterion will define the acceptability of deals:

DEFINITION 2 (RATIONALITY). *A deal $\delta = (A, A')$ is individually rational iff there exists a payment function p such that $u_i(A') - u_i(A) > p(i)$ for all $i \in \mathcal{A}$, except possibly $p(i) = 0$ for agents i with $A(i) = A'(i)$.*

The following result, due to Endriss et al. [7], makes the connection between the local decisions of agents and the global behaviour of the system explicit:

LEMMA 1. *A deal $\delta = (A, A')$ is individually rational iff $sw_u(A) < sw_u(A')$.*

A corollary of Lemma 1, originally due to Sandholm [15], is that *any* sequence of rational deals will eventually result in an allocation of resources with maximal social welfare. However, deals involving any number of agents and resources may be required to do so [15, 7]. In particular, 1-deals alone are certainly *not* sufficient for negotiation between agents that are not only rational but also myopic. Of course, for some particular negotiation problems, rational 1-deals *will* be sufficient. This is, for instance, the case when all agents are using *modular* utility functions [7].[1]

Paper Overview

The remainder of this paper is structured as follows. Section 2 presents the challenges posed by the study of egalitarian outcomes of distributed negotiation. Different types of payment functions are then introduced (Section 3), and we discuss how the choice of a particular function would affect the egalitarian nature of negotiation outcomes. Our experimental setup is introduced in Section 4. We then report results on different series of tests, both in modular (Section 5) and non-modular domains (Section 6). Section 7 concludes.

2. EGALITARIAN FRAMEWORK

While the utilitarian interpretation of the concept of social welfare is the definition usually adopted in the multiagent systems literature [16, 17], there are also several other formal tools for assessing the level of welfare of a society of agents that have been developed in the socio-economic sciences [1, 3, 12] and that have potential applications in the context of multiagent resource allocation as well [7].

Efficiency and Fairness

The well-known Pareto condition and the utilitarian social welfare ordering address the *efficiency* of an allocation of resources. In many situations, however, the efficiency of an outcome alone is not sufficiently accurate a criterion to assess the quality of an allocation of goods. Rather, the quest for efficiency needs to be balanced with certain *fairness* requirements.

This is, for instance, the case when a multiagent resource allocation system is used to automatically allocate airport slots to different airlines [9]: we want to minimise the overall delay of all flights (efficiency), without risking to completely ignore the requests of a particular flight that may be difficult to integrate into the current schedule (fairness). Another example are reverse combinatorial auctions for industrial procurement, where a single buyer solicits offers for, say, the parts required to assemble 500 cars [8]. Here it may be in the interest of the buyer to try to contract deals with more than one seller, even when that agreement does not entail the lowest possible price, in order to avoid being dependent on the good will of a single provider. This kind of safety constraint (from the viewpoint of the buyer) may also be interpreted as a fairness requirement (from the viewpoint of the competing sellers). The case of earth observation satellites discussed by Lemaître et al. [10] offers a particularly interesting and original example. In this application, several stakeholder (e.g. countries) are jointly funding a satellite that is circling the earth and can take photos, subject to various technical constraints. Different stakeholders request different (and often incompatible) photos, and a solution needs to be found that is both efficient (in the sense of

[1] A utility function is modular iff the utility assigned to a set of resources is always the sum of utilities assigned to its members.

exploiting the satellite as well as possible) and fair (in the sense of honouring an appropriate share of the requests of each of the stakeholders). Finally, Porter et al. [13] justify their study of fairness by invoking the following application: In many countries, in crisis situations, the state can commandeer diverse resources (e.g. aircrafts) from companies to cope with the emergency. However, this should be done in such a way that it minimizes the total true cost incurred by airline carriers (and by the state), and distribute this cost fairly amongst all carriers (so as not to penalize a carrier).

What these examples show is the relevance of notions such as *fairness*, *equitability*, or *envy-freeness* in many (diverse) applications. This paper will focus on what may be considered the most "basic" measure of the equitability of given allocations of goods, known as the *egalitarian social welfare*. The egalitarian social welfare is given by the utility of the agent that is currently worst off [1, 12]. That is, maximising egalitarian social welfare amounts to maximising the minimum utility in the system. This *maxmin principle* is also called 1-efficiency in the work of Porter et al. [13].

Optimal Outcomes and Local Deals

We are going to be interested in mechanisms that allow agents to reach an allocation with high (and ideally, maximal) egalitarian social welfare by means of a sequence of locally agreed deals. These deals should be *local* also and particularly in the sense that the agents should be able to decide locally within the group of agents about to agree on a given deal whether or not that deal should be deemed acceptable. In other words, we are not looking for just any algorithm to optimise with respect to egalitarian social welfare, but for a distributed negotiation scheme.

The first kind of deal we are going to consider are the *individually rational* deals introduced earlier. It is easy to show that individually rational deals alone are not sufficient to allow agents to find an allocation with maximal egalitarian social welfare in all cases (while, on the contrary, this is possible in the case of utilitarian social welfare [15, 7]). This can be seen in the following example. Assume there are two agents and two possible allocations (*i.e.* there may be just a single resource):

$$u_1(A) = 5 \quad u_1(A') = 2$$
$$u_2(A) = 1 \quad u_2(A') = 3$$

Then A' has maximal egalitarian social welfare, but in case A is the initial allocation there is no individually rational deal leading from A to A' (because the amount of utility lost by agent 1 would be greater than the utility gained by agent 2).

Another approach would be to change the notion of individual rationality and to require agents to use a different criterion when deciding on the acceptability of a given deal. This idea has been followed in previous work where the class of so-called *equitable deals* has been proposed [7]. This criterion, which requires the minimum utility within the group of agents contracting a deal to increase, can guarantee convergence to an egalitarian optimum. However, as also pointed out in [7], this is not a precise criterion in the sense that it is still possible to contract further equitable deals after an allocation with maximal egalitarian social welfare has been found. In fact, it is *impossible* to design a deal acceptability criterion that would be precise in this sense. This is related to the fact that the egalitarian social welfare ordering is not *separable* [12], which means that it is not possible to decide whether or not egalitarian social welfare has increased simply by looking at the utility levels of the agents that have experienced a change in utility after a deal. Hence, it is also not possible for the agents involved in a deal to locally verify whether the deal they are about to implement would affect egalitarian social welfare in a positive manner.

Besides such technical considerations, of course, the individual rationality criterion is also much more realistic than the idealised concept of equitable deals. Therefore, in the present paper, we are going to follow a different approach. Rather than designing agents that are "good" for egalitarian social welfare, we are going to study just how "bad" the standard utility-maximising agents are from an egalitarian point of view. To put it differently, we are going to analyse how egalitarian social welfare develops in a society where agents negotiate a sequence of individually rational deals.

Egalitarian Social Welfare and Money

Before getting further in that direction, we need to address the first challenge this approach poses. Typically, a social welfare measure does *not* take a monetary component into account. There is an obvious reason for that when utilitarian social welfare is concerned: the payment function p has been defined in such a way that the sum of payments is always zero. Hence, side payments can never change the sum of utilities, *i.e.* they can never affect utilitarian social welfare. Things are different for egalitarian social welfare. Not taking into account monetary payments would amount to assessing buyers as being more satisfied than sellers, whatever the price they pay to get the resource(s) may be. In the most extreme case, an agent selling all its goods for very good (high) prices (and thereby certainly being "happy") would still provoke a very low egalitarian social welfare (because its utility function defined over bundles of resources alone would return a zero value).

To circumvent this problem, payments have to be included into agents' utility functions. Let the *balance* of agent i be defined as the sum of all the payments $p(i)$ of that agent paid out (or received, in the case of negative payments) during all previous deals. By Lemma 1, if agents only negotiate individually rational deals, no allocation can be visited more than once. So we can index an agent's balance with the name of the allocation in question: $bal^A(i)$ is the balance of agent i once allocation A has been reached. Then we define for each agent i a second utility function u_i' that combines the utility derived from the bundle of resources it currently holds and its monetary balance:

$$u_i'(A) \quad = \quad u_i(A) - bal^A(i) \tag{2}$$

In the literature, such utilities are often called *quasi-linear* utilities, in which case u would be referred to as a valuation and u' as the actual utility [4]. The egalitarian social welfare of an allocation with money is now simply defined as the egalitarian social welfare with respect to the utilities u_i':

$$sw_e(A) \quad = \quad min\{u_i'(A) \mid i \in \mathcal{A}\} \tag{3}$$

3. PAYMENT FUNCTIONS

For any given deal that is individually rational, there will be a range of possible payment functions (in theory, there are infinitely many, in practice the number will depend on the granularity of the currency used). This raises the question

what payments to implement in a simulation context, where agents do not actually negotiate prices with each other. Choosing a payment function p for an individually rational deal $\delta = (A, A')$ means deciding how to distribute the social surplus $sw_u(A') - sw_u(A)$ amongst the agents in the system. The only condition that the payment function has to meet is to ensure that side payments add up to zero.

Amongst these payment functions, some will ensure that any socially beneficial deal is indeed individually rational (and vice versa), that is that for each agent i involved in the deal, we indeed have $u_i(A') - u(A) > p(i)$: we shall call these *rational-compatible payment functions*.

Locally Uniform Payment Function

We begin with what we call the *locally uniform payment function*, which ensures an equal payoff amongst all the agents involved in the deal (and gives no payment to any of the non-involved agents):

$$p(i) = \begin{cases} u_i(A') - u_i(A) - \frac{sw_u(A') - sw_u(A)}{|\mathcal{A}^\delta|} & \text{if } i \in \mathcal{A}^\delta \\ 0 & \text{otherwise} \end{cases}$$

Note that these payments do indeed add up to 0 as required, and that each agent $i \in \mathcal{A}^\delta$ receives the same positive payoff $u_i(A') - u_i(A) - p(i) = \frac{sw_u(A') - sw_u(A)}{|\mathcal{A}^\delta|}$.

The uniform payment function is not an arbitrary choice, but intended to be a reasonable approximation of actual payments that agents would agree upon in a real negotiation context (rather than a simulation). Equally dividing the social surplus amongst all the agents concerned also means maximising the product of individual payoffs. In the case of deals between just two agents, this corresponds to the result we would get if agents were to adopt the well-known Zeuthen strategy to negotiate the payments for a given deal [14]. This is true, at least, if we assume that agents calculate their "willingness to risk conflict" with respect to the payoff for the deal at hand (which is a reasonable assumption if agents have comparable levels of utility to begin with).

Globally Uniform Payment Function

The next type of payment that we introduce is one which ensures an equal payoff amongst *all* the agents within the society:

$$p(i) = u_i(A') - u_i(A) - \frac{sw_u(A') - sw_u(A)}{|\mathcal{A}|}$$

Observe that this function is also rational-compatible (for the same reason that the locally uniform one is), although agents involved in deals will typically receive a much more modest share of the social surplus. Note that this payment function is not inequality reducing, but not inequality increasing either. By distributing the social surplus equally amongst all agents of the society, everyone simply benefits with any deal and enjoys the same welfare increase.

It is easy to see that with such a payment function, the egalitarian social welfare depends both on the gain of utilitarian social welfare during the entire negotiation, and on the initial satisfaction of the agents. More precisely, consider a sequence $[A_0, A_1, \ldots, A_f]$ of allocations visited during a negotiation process. For all $t \in \{0..f-1\}$, we have (where $p_t(i)$ denotes the payment of agent i at time t):

$$u_i(A_{t+1}) - u_i(A_t) - p_{t+1}(i) = \frac{sw_u(A_{t+1}) - sw_u(A_t)}{|\mathcal{A}|}$$

Observe that:

$$u_i(A_f) - bal^{A_f}(i) = u_i(A_0) + \frac{sw_u(A_f) - sw_u(A_0)}{|\mathcal{A}|}$$

Thus, we finally have:

$$sw_e(A_f) = min_i \left\{ \frac{sw_u(A_f) - sw_u(A_0)}{|\mathcal{A}|} + u_i(A_0) \right\}$$

It is interesting to note that:

$$sw_e(A_f) - sw_e(A_0) = \frac{sw_u(A_f) - sw_u(A_0)}{|\mathcal{A}|}$$

This has two important consequences: first, the gain of egalitarian social welfare is proportional to the gain of utilitarian social welfare. Also, the optimal allocation(s) w.r.t. utilitarian social welfare is/are the *same* as the optimal allocation(s) w.r.t. egalitarian social welfare.

It should also be pointed out that in the case of two agents, locally uniform payment functions are of course equivalent to globally ones. As reported later on in this paper, we carried out experiments to find out from what number of agents on do locally uniform payment functions give rise to negotiations exhibiting significantly different outcomes (see Fig. 3).

Fully Locally Equitable Payment Function

In theory, it is possible to conceive an equitable payment function which would, at each step, compute a payment such that each agent (involved in the deal) would enjoy the same utility level after the deal has been achieved. This idea is very similar to that of *fair imposition* proposed in [13]. Such a payment would take the following form:

$$p(i) = \begin{cases} u_i(A') - bal^A(i) - \frac{\sum_{j \in \mathcal{A}^\delta} (u_j(A') - bal^A(j))}{|\mathcal{A}^\delta|} & \text{if } i \in \mathcal{A}^\delta \\ 0 & \text{otherwise} \end{cases}$$

Note that here, the balance $bal^A(i)$ refers to the balance of the previous allocation (A). The above must be read as follows: compute the restricted social welfare divided it by the number of agents involved (that would be the optimal repartition), then subtract the payment such that each agent reaches exactly that level of satisfaction. This is exemplified on the following example (assuming this is the first deal taking place, hence a balance of 0 for each agent):

$$\begin{array}{ll} u_1(A) = 3 & u_1(A') = 2 \\ u_2(A) = 1 & u_2(A') = 6 \end{array}$$

The final allocation exhibits a restricted (utilitarian) social welfare of 8, hence 4 for each agent would be the optimal situation. This is obtained by computing the payment function which pays +2 to agent 1 and -2 to agent 2. However, such a payment would not always be rational: in many situations, the payment would be such that at least a self-interested agent would have no interest in implementing the deal. This is for instance the case on the following example:

$$\begin{array}{ll} u_1(A) = 3 & u_1(A') = 2 \\ u_2(A) = 1 & u_2(A') = 3 \end{array}$$

In this case, the computed payment is +/- 0.5, and it is easy to see that agent 1 would not rationally accept such a deal. In general, a deal is only rational if every agent enjoys before the deal takes place a utility that is strictly lower than the mean of the restricted social welfare after the deal. This is then an example where a socially beneficial deal would not be individually rational for agents if they were to use this payment function, *i.e.* this proves that this payment function is not rational-compatible. The consequence is that it will change the structural properties of the framework: by restricting the number of feasible deals, certain sequences of deals (that could be rational with a different payment function) would not be explored. Clearly, this can be a good thing in some situations, as exemplified below:

$$\begin{array}{lll} u_1(\{r_1\}) = 2 & u_1(\{r_2\}) = 2 & u_1(\{r_1, r_2\}) = 104 \\ u_2(\{r_1\}) = 100 & u_2(\{r_2\}) = 0 & u_2(\{r_1, r_2\}) = 100 \\ u_3(\{r_1\}) = 0 & u_3(\{r_2\}) = 101 & u_3(\{r_1, r_2\}) = 101 \end{array}$$

We first suppose that agents use *any* rational-compatible payment function, and that a_2 initially holds the bundle $\{r_1, r_2\}$ (initial social welfare is 100). Now the following deal sequence takes place: (i) a_2 gives r_2 to a_1 (leading to a social welfare of 102), (ii) a_2 gives r_1 to a_1 (leading to a social welfare of 104). No more rational deal is possible (giving any resource would result in a loss of 102).

Of course, a social welfare of 201 would have been attainable, by simply offering r_2 to a_3 as a first deal. Equitable deals (payment 0.5) would indeed have discarded the deals implemented in the suboptimal sequence, leaving only this deal possible. Whether this is a good thing or not in general, *i.e* whether this payment function behaves as a good heuristic in this context will be experimentally evaluated later on in this paper (see Fig. 2 and 4).

Rational Locally Equitable Payment Function

What would be the "most equitable" payment function that would still be rational-compatible? A slight modification of the previous payment function would in fact suffice to make it rational-compatible: compute the payment function such that it makes every agent marginally better off, then allocate the remaining payments induced by the deal so as to reduce inequalities as much as possible (that is, give the remaining social surplus to the agent currently worse off).

For a deal $\delta = (A, A')$, formally, this payment can be defined as the function p maximizing

$$min_{i \in \mathcal{A}^\delta} \left\{ u_i(A') - bal^A(i) - p(i) \right\}$$

where $bal^A(i)$ is the balance of agent i before the deal, and under the two following constraints: p has to be a payment function ($\sum_{i \in \mathcal{A}} p(i) = 0$) and the deal must be rational ($\forall i \in \mathcal{A}^\delta, u_i(A') - u_i(A) > p(i)$). Obtaining an analytic formulation for p in the general case is difficult. However, in the case of bilateral negotiation, the solution to the above constraint equation for any socially beneficial deal $\delta = (A, A')$ involving agents i and j, can easily be computed. First note that in case the deal is rational with the fully locally equitable payment, this defines our payment function. Otherwise, observe that there is necessarily a single agent, say i, such that $u_i(A) \geq (u_i(A') + u_j(A'))/2$. This is so, because if both agents met this condition, then the deal would be fully equitable, while if neither of them met the condition, the deal would clearly not be be individually rational. Now we fix the payment for that agent i, in such a way that this agent is just marginally better off, in order to make the deal rational. That is, $p(i) = u_i(A') - u_i(A) - \epsilon$ for some suitably small constant $\epsilon > 0$. Note that $p(i)$ could be a positive or negative value; the payoff for i will be $+\epsilon$ either way. Then we define the other payment as $p(j) = -p(i)$.

We take again our previous example (that proved fully equitable payment functions not to be rational-compatible), to show how this works. Agent 1 has been better off before the deal than the average utility after the deal. Its payment will $p(1) = 2 - 3 - \epsilon$, *i.e.* it *receives* an amount of $1 + \epsilon$ from agent 2. After the deal, we have the following utilities (assuming there have been no earlier payments):

$$\begin{array}{l} u'_1(A') = 3 + \epsilon \\ u'_2(A') = 2 - \epsilon \end{array}$$

Whether this payment function outperforms other functions (as far as egalitarian social welfare is concerned) will be evaluated experimentally in this paper (see Fig. 1 and 5).

4. EXPERIMENTAL METHODOLOGY

Our experimental results are based on the following methodology: we first *create the system* (agents' utility functions and the initial allocation of goods), we then *compute the optimal utilitarian social welfare*, and we finally *run negotiations* (that is, let agents negotiate until no more deals are possible). Agents typically desire to hold several bundles (the details of how these utility functions are generated depends on the domain studied, and will be given later on). To avoid favouring a specific agent, the negotiation protocol is randomized: we pick an agent at random, and try to identify one rational deal involving this agent. Remember that as soon as one such rational deal meets the rationality constraints of the seller and the buyer, the deal is implemented and the allocation is updated. The results are averaged over 500 negotiation runs.

The reason why we compute the optimal utilitarian social welfare is that it serves to derive a (very generous) upper bound on optimal egalitarian social welfare: The best you can do is to optimize overall utility ($=sw_u$) and then use payments to distribute the overall wealth as evenly as possible (it is then simply $1/n$ times the maximal utilitarian social welfare). In the restricted negotiation framework that we investigate here, this optimal value will seldom be attained. What will be highly significant will be the value of the final utilitarian social welfare, which would represent an upper bound under the constraints of the negotiation framework (*e.g.* restrictions to 1-deals, rationality constraints). Note that affecting the class of acceptable deals by employing a non-rational-compatible payment function falls into that category. Again, distributing the overall wealth as evenly as possible would be the optimal egalitarian outcome (recall that a *theoretical* equitable payment function would allow to attain this value).

To sum up, the experimental results reported in this paper will typically mention the following outcome values:

(1) the value of the *optimal utilitarian welfare* (sw_u^{opt}/n), upper bound in unrestricted negotiation frameworks;

(2) the value of the *final utilitarian welfare* (sw_u/n), upper bound under the constraints imposed during this negotiation;

(3) the value of the *final egalitarian welfare* (sw_e), after negotiation took place.

The difference between (1) and (2) is an indicator for what we shall call the *protocol bias*. The difference between (2) and (3), more interesting to us, permits to appreciate the *payment function bias*. Comparison with the initial value of the egalitarian welfare proved to present very little interest in our experiments: it has been omitted for the sake of readability.

Figures presented in the next two sections exhibit curves defined by the following parameters: (1) the type of social welfare measured (`SWe` or `SWu/n`); (2) whether the social welfare shown is the `optimal` value or the `final` value; (3) the payment function used; and (4) the type of distribution used.

5. MODULAR DOMAINS

We start with very simple domains of negotiation, where no synergies can occur between different items of the bundles agents hold. In this case agents' utility functions are said to be *modular*. We generate these functions by simply picking 50 resources for each agent, and assigning random coefficients between 0 and 100. Note that in this case the optimal utilitarian social welfare can be easily computed: It suffices to assign each item to the agent who values it the most. (This allows in particular to run experiments with a large number of items). A recent (somewhat surprising) complexity result, due to Bouveret and Lang [2], shows that the difficulty of the corresponding problem largely differs for egalitarian frameworks *without money:* Optimising the egalitarian social welfare is an NP-hard problem, even when agents use modular functions (this is even true when they all use the same utility function). This is *not* the case in our setting though, because of the use of money (which allows us to separate the process of maximising overall utility and then distributing that utility evenly amongst the agents). A final important point to note is that the complexity of the overall negotiation process remains polynomial in this case, because the length (number of deals) of rational negotiations in modular domains is known to be linear [6].

Locally Uniform vs. Rational Locally Equitable

The first set of experiments has been run with 5 agents using modular utility functions, deals enhanced with rational-compatible payment functions. The number of resources varies between 50 and 150. Recall that in this case, we know that the optimal utilitarian social welfare will be reached (*i.e.* there is no "protocol bias"): the experiment will really measure the "payment function bias". To put it another way, if there were such a thing as a rational-compatible fully equitable payment function, both curves would be equivalent.

Fig. 1 shows the final egalitarian social welfare attained when locally uniform or rational equitable payments are used. It is striking that, first, both payment functions perform reasonably well (the ratio (sw_e/sw_u^{opt}) is constantly higher than 70% and improves when the number of resources increases). The rational equitable payment is, as expected, significantly above the locally uniform one (84% vs. 70% for 60 resources, 93% vs. 80% for 120 resources, 96% vs. 83% for 150 resources). However, given that this latter payment is, as argued before, the "most equitable" conceivable

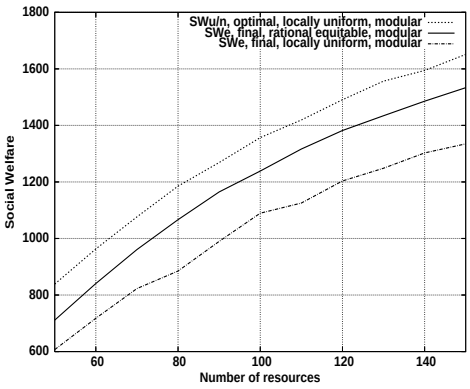


Figure 1: Modular Domains (I)

rational-compatible payment function, it is still surprising to see how close the "strategically justified" locally uniform payment stands.

Fully Locally Equitable

The second series of experiments (Fig. 2) carried out in modular domains tests the assumption that fully equitable deals can be a good heuristic in our negotiation setting, despite reducing the number of feasible deals. The results are unambiguous: in modular domains, this is far from being the case. The protocol bias (which exists in this case), is illustrated by the ratio between the optimal value and the final sw_u/n reached when allowing only fully equitable deals. For 50 resources already, the figure is around 75%, and drops to 50% for 150 resources. This really measures the loss due to the restriction put on the framework. Worse than that, the final sw_e exhibits a figure (sw_e/sw_u^{opt}) of 45% for 50 resources, shrinking down to 16% for 150 goods. Actually, the final egalitarian social welfare hardly increases in this setting. This really proves that this restriction is simply too hard to be met, and almost freezes the negotiation process.

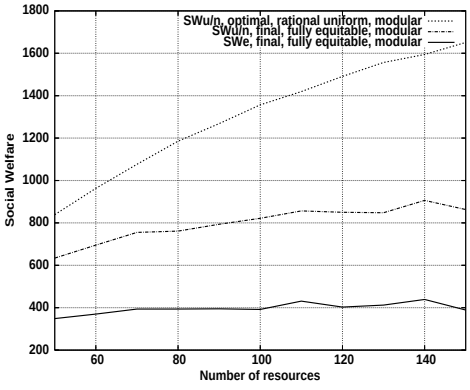


Figure 2: Modular Domains (II)

6. NON-MODULAR DOMAINS

In many situations, the assumption that resources have no synergies between them will be highly unrealistic. It is then very important to investigate the outcome of negotiation in such (non-modular) domains. The first experiments that we

envisaged were produced using a uniform distribution for the generation of the structure of utility functions. The results (not reported here), regarding the final egalitarian welfare, turned out to be very good. This simply reflected the fact that uniform distributions tended to distribute evenly the participants' needs: agents were seldom in conflict.

Realistic Distributions

In order to get realistic agent preferences we decided to make use of the CATS bid generator [11], primarily designed for the use with combinatorial auctions. The output of the CATS software is a set of bids, taking the form of XOR demands $\langle b_1, p_1\rangle \ldots, \langle b_n, p_n\rangle$, where b_i stands for the bundle of resources, and p_i stands for the value the agent assigns to this bundle. One problem that we faced when using CATS is the fact the output file contains only anonymous bids. This is perfectly suited for the case of studies of algorithms for the Winner Determination Problem in combinatorial auctions [4], because then you are not really concerned with associating resources with agents who claimed them, but only with computing the optimal solution. What we did to deal with this was to generate a (large) number of bids in a single shot (interacting in the same domain, it is likely that agents share a common dependency structure between goods). However, by allowing dominated bids to be generated, we permit agents to express different degree of preferences over the same bundles of goods. And finally, by picking (at random) only a subset of these generated bids, we allow agents to demand different bundles and admit that they may only partially share this common structure. The experiments reported in this section have been conducted using the CATS *arbitrary* distribution.

Locally Uniform vs. Globally Uniform

Fig. 3 shows the egalitarian social welfare reached when an increasing number of agents negotiate over 15 resources using locally uniform payments. Each agents claims 10 bundles. Recall that, as mentioned in the previous section, when there are no more than two agents, the locally uniform payment and the globally uniform payment coincide.

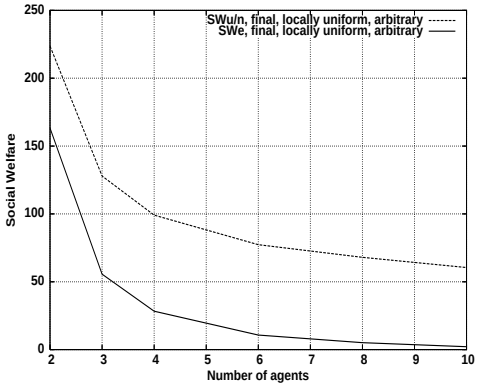


Figure 3: Non-Modular Domains (I)

First notice that, as expected, the final sw_u/n decreases exactly as a $1/n$ curve. Intuitively, we could have expected the final sw_e to decrease at the same speed. This turns out not to be the case: from 2 to 3 agents, sw_e decreases by 65%, whereas a $1/n$ decrease rate would have been of 33%. This observation can be explained in the light of the locally uniform payment function singularity: when there are no more than two agents, the locally and globally uniform payment are identical, in which case the inequalities between both agents will not increase during the negotiation. In other words, as explained in the previous section, the gain of sw_e between the beginning and the end of the negotiation will be equal to the gain of sw_u divided by n. This explains why the final sw_e for two agents is close to the final sw_u.

On the contrary, for more than 2 agents, each bilateral deal will only favor the two agents involved in it, and will thus potentially increase inequalities between agents contracting many deals, and others. This allows us to explain the rapid decrease of sw_e at the beginning of the curve.

Fully Locally Equitable vs. Locally Uniform

Next we compare fully locally equitable payment functions and locally uniform payment functions (Fig. 4). The results show that negotiation using fully equitable payments is notably less equitable than negotiation with uniform payments. This is so because the criterion is, again, too restrictive and prevents agents from contracting deals that could be rational (and hence improve social welfare). The fact that this function helps to select highly beneficial deals does not compensate for the loss incurred by these missed opportunities. This completes the answer to the question we left pending in Section 5: this payment function does not behave as a good heuristic, in this type of domains neither. The reason why the difference is far less important with uniform payment functions in these domains lies mainly in the fact that, typically, much fewer deals are contracted, and that values assigned to bundles are typically higher than those assigned to single resources (augmenting the probability to contract a fully equitable deal).

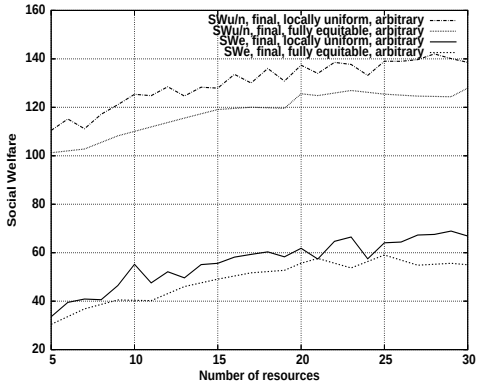


Figure 4: Non-Modular Domains (II)

Rational Locally Equitable vs. Locally Uniform

In this last series of experiments, we show that rational equitable payments slightly outperform uniform payments as far as the satisfaction of the poorest agent is concerned. (Remember that feasible deals being the same for these two types of payments, and payment functions adding up to zero at the end of a given negotiation, the fact that final values of averaged utilitarian welfare are similar should come as no surprise.) In Fig. 5, we observe that the difference with uniform payments is marginal for small number of resources,

but becomes slightly larger when this number increases. This is explained as follows: when there are few resources to be distributed, the number of deals that will take place during a negotiation will typically be very small. Each contracted deal increases the probability of redistributing some wealth to the poorest agent, the more deals contracted the happier that agent should get (on average).

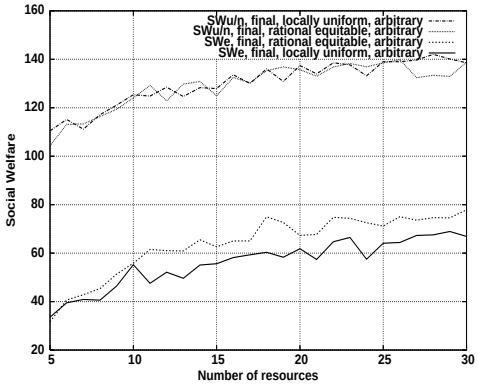


Figure 5: Non-Modular Domains (III)

7. CONCLUSION

This paper has investigated the egalitarian properties of some distributed resource allocation processes. An important aspect of the framework, that has radical consequences on egalitarian outcomes, is the choice of a specific payment function (*i.e.* the way to redistribute the social surplus induced by any given deal). We have discussed a variety of payment functions, and shown in particular that a perfectly equitable payment function is not rational-compatible. Our study has been complemented with several experiments, that have allowed us to identify conditions favourable to egalitarian outcomes. In particular, we should stress that:

- In modular domains, we observed that locally uniform payments give rise to reasonably equitable outcomes, which becomes comparatively more equitable as the number of resources grows. This remains significantly below (but closer than expected) to what could be achieved at best within the class of rational-compatible payments.
- It is typically not beneficial to enforce agents to obey to implement a fully equitable payment function, because this would conflict with rationality principles and prevent agents from conducting socially beneficial deals. This is striking in modular domains, less obvious in non-modular domains where there are typically fewer deals taking place during the negotiation.
- Locally uniform payment functions significantly differ from globally uniform payment functions for more than 2 agents. More generally, as the number of agents grows, we cannot expect any local payment to compensate for the inequalities.

A possible extension of this work will be to investigate how the conclusions drawn here will vary for other "standard" CATS distributions. A longer-term issue will be the study of other equitability measures: the *maximin* egalitarian principle is but one (admittedly rough, because it only takes into account the poorest agent) equitability measure. To refine our study, we plan to investigate other measures, such as Gini-like indexes for instance [12], which give a more global measure of a society's wealth distribution.

8. REFERENCES

[1] K. J. Arrow, A. K. Sen, and K. Suzumura, editors. *Handbook of Social Choice and Welfare*. North-Holland, 2002.

[2] S. Bouveret, H. Fargier, J. Lang, and M. Lemaître. Allocation of indivisible goods: A general model and some complexity results. In *Proc. AAMAS-2005*. ACM Press, 2005.

[3] S. J. Brams and A. D. Taylor. *Fair Division: From Cake-cutting to Dispute Resolution*. Cambridge University Press, 1996.

[4] P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006.

[5] P. E. Dunne, M. Wooldridge, and M. Laurence. The complexity of contract negotiation. *Artificial Intelligence*, 164(1–2):23–46, 2005.

[6] U. Endriss and N. Maudet. On the communication complexity of multilateral trading: Extended report. *Journal of Autonomous Agents and Multiagent Systems*, 11(1):91–107, 2005.

[7] U. Endriss, N. Maudet, F. Sadri, and F. Toni. Negotiating socially optimal allocations of resources. *Journal of Artificial Intelligence Research*, 2006. To appear.

[8] A. Giovannucci, J. A. Rodríguez-Aguilar, A. Reyes, F. X. Noria, and J. Cerquides. *i*Bundler: An agent-based decision support service for combinatorial negotiations. In *Proc. AAAI-2004*. AAAI Press, 2004.

[9] G. Jonker, J.-J. Meyer, and F. Dignum. Efficiency and fairness in air traffic control. In *Proc. 17th Belgian-Dutch Conf. on Artificial Intelligence*, 2005.

[10] M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille. Selecting and scheduling observations of agile satellites. *Aerospace Sciences and Technology*, 6:367–381, 2002.

[11] K. Leyton-Brown, M. Pearson, and Y. Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proc. EC-2000*, 2000.

[12] H. Moulin. *Axioms of Cooperative Decision Making*. Cambridge University Press, 1988.

[13] R. Porter, Y. Shoham, and M. Tennenholtz. Fair imposition. *Journal of Economic Theory*, 118(2):209–228, 2004.

[14] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter*. MIT Press, 1994.

[15] T. W. Sandholm. Contract types for satisficing task allocation: I Theoretical results. In *Proc. AAAI Spring Symposium: Satisficing Models*, 1998.

[16] T. W. Sandholm. Distributed rational decision making. In *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

[17] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley and Sons, 2002.

Mathematical Programming for Deliberation Scheduling in Time-limited Domains

Jianhui Wu*
EECS Department, University of Michigan
Ann Arbor, MI 48109 USA
jianhuiw@umich.edu

Edmund H. Durfee
EECS Department, University of Michigan
Ann Arbor, MI 48109 USA
durfee@umich.edu

ABSTRACT

Deliberation scheduling is the process of scheduling decision procedures to maximize overall performance. In this paper, we present novel mathematical programming algorithms for scheduling deliberations, and illustrate them through several increasingly complex classes of deliberation scheduling problems. In comparison to previous work, our algorithms are able to optimally or near-optimally solve deliberation scheduling problems for decision procedures that have more general performance profiles, and are applicable in complex domains with uncertainty. We also illustrate that, thanks to the mathematical programming formulation, our algorithms can be easily extended to model additional system aspects.

Categories and Subject Descriptors

I.2.8 [**ARTIFICIAL INTELLIGENCE**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms

Keywords

Deliberation scheduling, control of reasoning, piecewise linear approximation, linear programming, mixed integer linear programming

1. MOTIVATION AND INTRODUCTION

Many complex problems are too difficult for autonomous agents to solve within deadlines. For example, an autonomous aircraft flying a prolonged mission might not have time to prepare a plan that specifies actions and reactions for all possible contingencies over the entire mission before it starts to execute the plan. In this situation, if the problem can be divided into multiple independent sub-problems (such as takeoff, ingress, rescue, egress, and landing), which we call *phases*, then the agent might do better to focus only on near-term phases. Then, while executing the plan for earlier phases, the agent could use available computational time during execution to reconsider aspects of the problem and improve its solutions for current and future phases. This assertion is not surprising. The challenge, though, is in automating the process of allocating computational time to appropriate phases given the uncertainty and complexity of the problem domain.

Deliberation scheduling is the process of scheduling decision procedures to maximize overall performance. Boddy and Dean proposed an optimal deliberation scheduling algorithm for a subclass of problems with piecewise linear concave value functions [2, 1]. Horvitz considered likelihood of future instances, and explored policies for proactive allocation of idle time for these potential problems in several settings [6]. Goldman, Musliner, and Krebsbach proposed a greedy deliberation scheduling algorithm and compared its performance with the optimal algorithm using Markov Decision Process models [5, 7]. However, these approaches are either only applicable in restricted subclasses of deliberation scheduling problems (e.g., for limited types of performance profiles for decision procedures), or ignore some aspects of systems (e.g., uncertainty, or deliberation cost).

Our work is directed at addressing these shortcomings. Our formulation of the deliberation scheduling problem as a mathematical program—as a linear program (LP) or mixed-integer linear program (MILP)—makes it much easier to work with a wide variety of performance profiles. It also allows us to explicitly model the costs of deliberation both before mission execution (where more deliberation allows an agent to prepare for more contingencies but delays the start of the mission) and during mission execution (where reconsidering plans can improve performance but can also distract when responding to emergent situations). Our mathematical programming approach is unique among proposed solutions to deliberation scheduling in its ability to work with a richer set of performance profiles, and to simultaneously solve the coupled problems of deciding both when to deliberate given its cost, and which decision procedures to execute during deliberation intervals.

Furthermore, our mathematical programming approach provides a domain-independent framework on which we can easily make simplifying transformations or impose additional constraints. For example, as we will show, a deliberation scheduling problem where the mission quality is the multiplication of individual phase qualities (i.e., a nonlinear ob-

*The primary author of the paper is a student.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

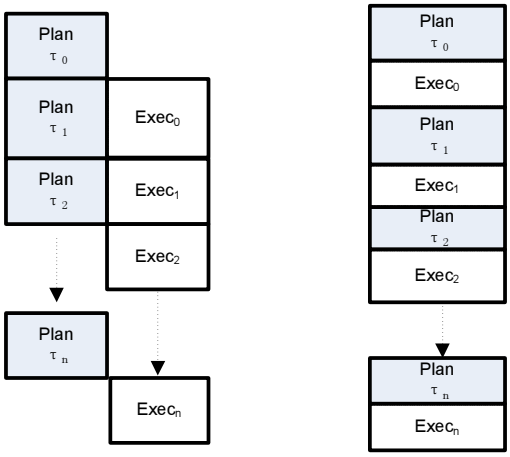


Figure 1: Deterministic phase transitions in two situations: simultaneous planning and execution (left) and interleaved planning and execution (right).

jective function) can be easily transformed into a problem where the mission quality can be represented as the sum of individual phase qualities (i.e., a linear objective function). A mathematical programming formulation also makes it possible to model additional system features. In this paper, we demonstrate this capability by modeling the fact that an autonomous agent should control which phases it will face in the future while scheduling deliberation.

The rest of the paper is organized as follows. Section 2 describes a basic mathematical programming formulation and introduces transformation and approximation techniques. Section 3 extends the formulation to suit domains involving uncertainty, and Section 4 makes further extensions by assuming that phase transitions are controllable by the autonomous agent. We present empirical results in Section 5, where we evaluate the efficiency and optimality of our approach. Finally, Section 6 summarizes the contributions of the work presented and outlines future directions.

2. DETERMINISTIC PHASE TRANSITIONS

We begin our examination of deliberation scheduling techniques by assuming that transitions among phases are deterministic —a mission can be represented as a chain of phases as shown in Figure 1. The left side of the figure depicts the case where deliberation and execution can occur simultaneously, such that deliberation about future phases can be done during execution of the current phase. The right side of the figure shows interleaved planning and execution, where an interval for deliberation is followed by an interval for execution, which in turn is followed by time for deliberation, and so on. In this paper, we describe and illustrate our algorithms on interleaved planning and execution. However, our algorithms, with slight modification, can also be applied for agents that simultaneously plan and execute.

Let τ_0 be the computational time that the autonomous agent initially has when the problem is presented, and τ_i ($i \geq 1$) be additional computational time that the agent can have after finishing the previous phase and before beginning the execution of $phase_i$. Let us temporarily assume that τ_i is specified *a priori*; we will relax this assumption shortly. The deliberation scheduling problem is then to schedule decision procedures within these available computational time intervals so that the expected quality of mission solution is maximized. A straightforward strategy is to allocate τ_i to $decision_i$ (where $decision_i$ represents the decision procedure for $phase_i$), but this myopic approach is usually suboptimal since it might be fruitful to use some of the time to get a head start on decision procedures for future phases.

In deliberation scheduling, a common construct is the expected value of computation (EVC) function (also called performance profile) [1, 2, 5, 6, 7]. An EVC (performance profile) of an anytime algorithm is the expected quality of the solution as a function of the algorithm's run time. In this paper, we use this construct as well.

Let t_i denote the total amount of computational time scheduled for decision procedure $decision_i$, and let $V_i(t)$ denote the EVC function of $decision_i$. The deliberation scheduling problem with deterministic phase transitions can then be represented in the following mathematical formulation:

$$\max f(v_0, v_1, \ldots, v_n) \quad \left| \begin{array}{l} \sum_{i=0}^{k} t_i \leq \sum_{i=0}^{k} \tau_i \\ v_i = V_i(t_i) \\ t_i \geq 0 \end{array} \right. \tag{1}$$

where $\forall k \in \{0, 1, \ldots, n\} : \sum_{i=0}^{k} t_i \leq \sum_{i=0}^{k} \tau_i$ indicates the fact that the amount of scheduled computational time can never exceed the amount of available computational time. v_i is the expected quality of the solution to $phase_i$, and the objective function $f(v_0, v_1, \ldots, v_n)$ specifies how the expected qualities of individual phase solutions determine the overall quality of the mission.

In this paper, we concentrate on the linear objective function $f(v_0, v_1, \ldots, v_n) = \sum_i v_i$, which fits many real-world domains where the quality of a mission is the accumulative quality through all its phases. An intuitive example is that of an autonomous delivery robot making several rounds of deliveries: its total quality is the sum of the qualities of individual package deliveries.

Notice that many nonlinear objective functions can be linearized (but not necessarily as $\sum_i v_i$) through mathematical reformulation, and thus the techniques presented in this paper are also applicable in those domains. We now introduce linearization methods for two typical prototypes of nonlinear objective functions.

- **Objective function** $f(v_0, v_1, \ldots, v_n) = \min v_i$

  A popular objective function in real-world domains is that the quality of a mission is the minimum quality of individual phases. This nonlinear objective function $\min v_i$ can be reformulated as a linear objective function v with additional linear constraints $\forall i : v \leq v_i$. That is, Eq.1 becomes

$$\max v \quad \left| \begin{array}{l} \sum_{i=0}^{k} t_i \leq \sum_{i=0}^{k} \tau_i \\ v_i = V_i(t_i) \\ v \leq v_i \\ t_i \geq 0 \end{array} \right.$$

- **Objective function** $f(v_0, v_1, \ldots, v_n) = \prod_i v_i$

  This objective function is usually used for domains where the probability of successfully completing a mission is the multiplication of the probabilities of successfully completing each phase. By using a logarithmic

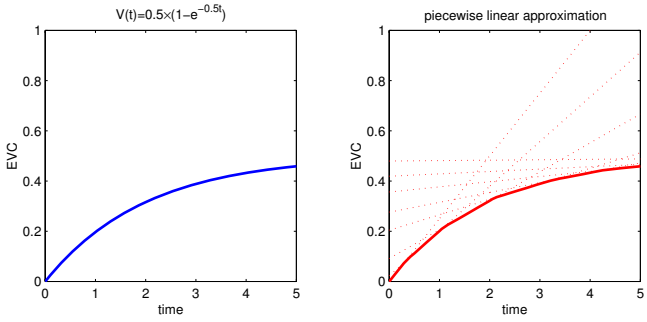


Figure 2: A continuous concave function (left), and its piecewise linear approximation (right).

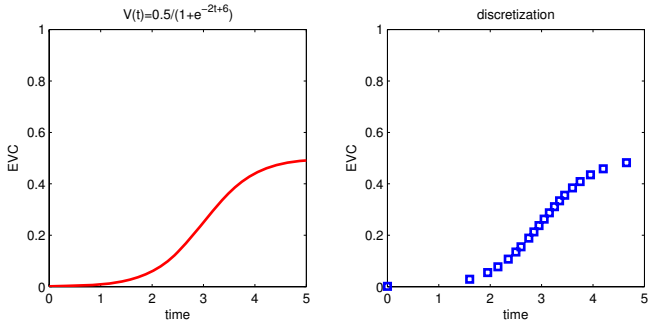


Figure 3: A general nonlinear function (left), and its discrete function (right).

transformation ($\varepsilon_i = \ln v_i$), Eq.1 can be transformed to:

$$\max e^{\sum_i \varepsilon_i} \quad \left| \begin{array}{l} \sum_{i=0}^{k} t_i \le \sum_{i=0}^{k} \tau_i \\ \varepsilon_i = V_i'(t_i) \\ t_i \ge 0 \end{array} \right.$$

where $V_i'(t) \equiv \ln V_i(t)$.

Notice that e^x is a monotonically increasing function, maximizing $e^{\sum_i \varepsilon_i}$ is equivalent to maximizing $\sum_i \varepsilon_i$, and so the objective function is linearized (while nonlinearity is moved to EVC functions).

We have introduced linearization methods for some nonlinear objective functions, but we have not fully addressed the difficulty in solving Eq.1, in which the constraint $v_i = V_i(t_i)$ might be nonlinear. Nonlinear optimization problems are generally intractable; in the following discussion, we present approximation techniques used for linearization in several settings.

- **Linear EVC function**

  If $V_i(t)$ are linear functions of t for any i, then $v_i = V_i(t_i)$ are trivial linear constraints, and deliberation scheduling problems can be formulated as linear programs which are solvable in polynomial time.

- **Continuous concave EVC function**

  For many anytime algorithms, the rate of refinement of the solution slows down with increasing computational activity, i.e., $V_i(t)$ is a continuous concave function. It is well established that a continuous concave function can be approximated as a piecewise linear concave function [8]. Using a sufficiently large number of pieces, such approximation usually performs well. Figure 2 shows an example of approximating function $V(t) = 0.5\times(1-e^{-0.5t})$ with a piecewise linear concave function that is composed of eight linear pieces.

  Considering that deliberation scheduling often occurs online, we adopt a simple but fast algorithm to construct piecewise linear functions, which, at each time step, myopically adds a linear piece that will reduce approximation error most. Our empirical results show that this myopic algorithm can, in general, approximate a function within several milliseconds, and thus is well suited for online applications.

Let $V_{i,j}(t) = a^v_{i,j} \times t + b^v_{i,j}$ be the linear function used to represent the jth segment of the piecewise linear concave curve, then function $V_i(t)$ can be approximated as $V_i(t) = \min_j a^v_{i,j} \times t + b^v_{i,j}$. That is, the constraint $v_i = V_i(t_i)$ in Eq.1 becomes $v_i = \min_j a^v_{i,j} \times t_i + b^v_{i,j}$, and the deliberation scheduling problem can be formulated into the following linear program:

$$\max \sum_i v_i \quad \left| \begin{array}{l} \sum_{i=0}^{k} t_i \le \sum_{i=0}^{k} \tau_i \\ v_i \le a^v_{i,j} \times t_i + b^v_{i,j} \\ t_i \ge 0 \end{array} \right. \qquad (2)$$

A linear program can be solved fast (i.e., in polynomial time), which explains why much previous work focused on continuous concave EVC functions and there exist fast algorithms for such problems [2].

- **General nonlinear EVC function**

  In the previous discussion, we have approximated deliberation scheduling problems with decreasing return rate EVC functions into linear programs. Now, we consider more general nonlinear EVC functions, and use discretization to remove nonlinearity in such functions [8]. Figure 3 shows an example of discretization.

  Let $T^v_{i,j}$ and $V_{i,j}$ represent the jth time point and its corresponding value on the discretized function of $V_i(t)$, and let binary variable $\delta^v_{i,j}$ represent whether time point $T^v_{i,j}$ is selected. The problem can then be formulated as:

$$\max \sum_i v_i \quad \left| \begin{array}{l} \sum_{i=0}^{k} t_i \le \sum_{i=0}^{k} \tau_i \\ t_i = \sum_j T^v_{i,j} \times \delta^v_{i,j} \\ v_i = \sum_j V_{i,j} \times \delta^v_{i,j} \\ \sum_j \delta^v_{i,j} = 1 \\ t_i \ge 0 \\ \delta^v_{i,j} \in \{0,1\} \end{array} \right. \qquad (3)$$

$\sum_j \delta^v_{i,j} = 1$ says that a certain amount of computational time is scheduled for decision procedure $decision_i$. The constraints $t_i = \sum_j T^v_{i,j} \times \delta^v_{i,j}$ and $v_i = \sum_j V_{i,j} \times \delta^v_{i,j}$ model the EVC function $V_i(t)$ through binary variable $\delta^v_{i,j}$.

Recall that for the phases with continuous concave EVC functions, we can approximate them into piecewise linear functions. That is, we only need to discretize non-concave EVC functions, which reduces the

number of binary variables used in mixed integer linear programs and thus improves the computational efficiency.

In the discussion so far, it is assumed that τ_i is known *a priori*, but in many real-world domains, τ_i is associated with a cost function $C_i(\tau_i)$ rather than being pre-specified. An autonomous agent thus also needs to determine the intervals of computational time it should use for improving solutions of its decision procedures.

Using our mathematical programming formulation, it is easy to make this extension. That is,

$$\max \sum_i (v_i - c_i) \quad \left| \begin{array}{l} \sum_{i=0}^{k} t_i \leq \sum_{i=0}^{k} \tau_i \\ v_i = V_i(t_i) \\ c_i = C_i(\tau_i) \\ \tau_i \geq 0 \\ t_i \geq 0 \end{array} \right. \quad (4)$$

Notice that the deliberation cost functions $C_i(\tau)$ are analogous to the EVC functions $V_i(t)$; we can use the approximation techniques described previously to linearize the constraint $c_i = C_i(\tau_i)$ as well. That is, when $C_i(\tau)$ is a continuous convex function (i.e., increasing cost rate), we can approximate it as a piecewise linear convex function. When $C_i(\tau)$ is a general nonlinear function, we can discretize it and build a mixed integer linear program.

We conclude this section by illustrating our algorithm on a simple (randomly generated) example, and more empirical results will be shown in Section 5. In this example, there are four phases $phase_{i\in\{0,1,2,3\}}$. The EVC functions are continuous concave functions:

$$V_0(t) = 3.1319 \times (1 - e^{-0.8233t})$$
$$V_1(t) = 4.0886 \times (1 - e^{-0.3603t})$$
$$V_2(t) = 0.2965 \times (1 - e^{-0.8393t})$$
$$V_3(t) = 2.3293 \times (1 - e^{-0.3057t})$$

The deliberation cost functions are continuous convex functions:

$$\begin{array}{ll} C_0(\tau) = 0.2037 \times (\tau - 1)^{1.5115} & \text{when } \tau \geq 1 \\ C_1(\tau) = 0.4808 \times \tau^{1.1843} & \text{when } \tau \geq 0 \\ C_2(\tau) = 0.4038 \times (\tau - 3)^{1.8348} & \text{when } \tau \geq 3 \\ C_3(\tau) = 0.2129 \times (\tau - 1)^{1.9415} & \text{when } \tau \geq 1 \\ \text{and} \quad C_i(\tau) = 0 & \text{otherwise} \end{array}$$

Approximating each $V_i(t)$ and $C_i(\tau)$ as a piecewise linear function with 20 pieces, this deliberation scheduling problem can then be formulated as a linear program. With the LP solver *cplex* (www.ilog.com), the mission's expected quality is 5.40 and solving the LP takes 0.021 seconds.

The deliberation schedule is shown in Figure 4. In detail, the agent spends 4.068 time units on its decision procedures

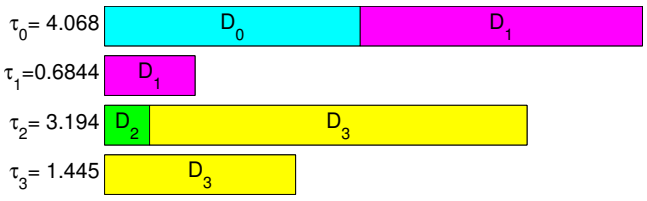


Figure 4: Deliberation schedule where D_i represents the decision procedure for $phase_i$.

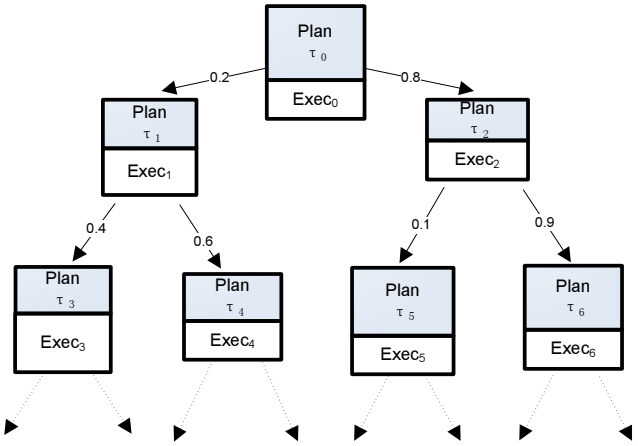


Figure 5: Uncertain phase transitions.

of $phase_0$ and $phase_1$ before it starts to execute the mission. After $phase_0$ is completed, it uses 0.6844 additional time units to improve the solution to $phase_1$. Since $phase_2$ has a much lower expected quality than $phase_3$, most of the available computational time before executing $phase_2$ is used for the decision procedure of $phase_3$. The resulting schedule achieves 20% higher expected quality than using a myopic algorithm that only runs the decision procedure of $phase_i$ right before that phase.

3. UNCERTAIN PHASE TRANSITIONS

We now consider more general deliberation scheduling problems where phase transitions are uncertain. In this paper, we focus on situations where phase transitions can be represented as a tree. Figure 5 shows one such problem. When the agent leaves a phase, it will reach one of the subsequent phases with some probability. The study of uncertain phase transitions is similar to Horvitz's previous work [6], in which future instances are nondeterministic. Our algorithms explore further by explicitly taking into account deliberation costs and more general EVC functions.

Let $p_{i,j}$ represent the transition probability from $phase_i$ to $phase_j$, and then the probability of reaching $phase_j$ in the mission, denoted as P_j, can be computed from $p_{i,j} \times P_i$ where $P_0 = 1$.

Let $\mathcal{A}_i$ denote the set composed of $phase_i$ and its ancestor phases, and let $\mathcal{D}_i$ denote the set composed of $phase_i$ and its descendant phases. Since computational time scheduled for decision procedure $decision_i$ can be from any phase in $\mathcal{A}_i$, $t_i = \sum_{k\in\mathcal{A}_i} \psi_{k,i}$ where $\psi_{k,i}$ is the period of computational time that is located at $phase_k$ and will be used for $phase_i$'s decision procedure. The deliberation scheduling problem with uncertain phase transitions then becomes

$$\max \sum_i P_i \times (v_i - c_i) \quad \left| \begin{array}{l} t_i = \sum_{k\in\mathcal{A}_i} \psi_{k,i} \\ \tau_k \geq \sum_{i\in\mathcal{D}_k} \psi_{k,i} \\ v_i = V_i(t_i) \\ c_i = C_i(\tau_i) \\ \tau_i \geq 0 \\ t_i \geq 0 \\ \psi_{k,i} \geq 0 \end{array} \right. \quad (5)$$

where the constraints $\forall k : \tau_k \geq \sum_{i\in\mathcal{D}_k} \psi_{k,i}$ guarantee that

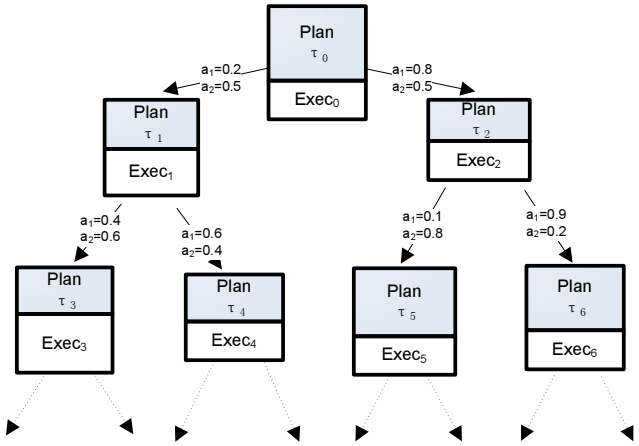


Figure 6: Controllable phase transitions.

the amount of scheduled computational time cannot exceed the amount of available computational time at any point within the mission.

The constraints $t_i = \sum_{k \in \mathcal{A}_i} \psi_{k,i}$ and $\tau_k \geq \sum_{i \in \mathcal{D}_k} \psi_{k,i}$ do not introduce nonlinearities. That is, using the techniques described in Section 2, Eq.5 can be approximated into a linear program whenever, for any $phase_i$, $V_i(t)$ is a continuous concave function and $C_i(\tau)$ is a continuous convex function. Eq.5 can be approximated into a mixed integer linear program whenever some phases have general nonlinear functions $V_i(t)$ or $C_i(\tau)$. Then, it is trivial to derive deliberation schedules from $\psi_{k,i}$ after solving the linear program or mixed integer linear program.

4. CONTROLLABLE PHASE TRANSITIONS

In this section, we further extend our mathematical programming formulation to model another important aspect of deliberation scheduling problems. Recall that in the previous sections it is assumed that transition probabilities (either deterministic or uncertain) among phases are specified *a priori*. However, in many situations, phase transitions might be controllable by agents themselves. For example, an autonomous aircraft needs to determine which route it wants to take when there exist multiple options for reaching a destination. That is, action choices of the agent determine (possibly stochastically) which phases will be reached next. Figure 6 shows one such problem.

With this extension, deliberation scheduling problems become more complex because an agent not only needs to control its reasoning, but also needs to find a policy that maps each phase to its action. A straightforward strategy for solving such problems is to enumerate all possible policies and then, for each policy, use the algorithms presented in the previous sections. However, when the number of policies is large, this simple strategy might become infeasible. In this section, we propose an alternative approach based upon a mathematical programming formulation.

Let $p_{i,a,j}$ represent the probability that the agent reaches $phase_j$ if it executes action a in $phase_i$, let α_i represent the probability that the agent is initially in $phase_i$ (i.e., $\alpha_0 = 1$, $\alpha_{i \geq 1} = 0$), and let $x_{i,a}$ represent the expected number of times that action a is executed at $phase_i$. We can then formulate deliberation scheduling problems with controllable phase transitions into:

$$\max \sum_i x_i \times (v_i - c_i) \quad \left| \begin{array}{l} \sum_a x_{j,a} = \alpha_j + \sum_i \sum_a p_{i,a,j} \times x_{i,a} \\ x_i = \sum_a x_{i,a} \\ t_i = \sum_{k \in \mathcal{A}_i} \psi_{k,i} \\ \tau_k \geq \sum_{i \in \mathcal{D}_k} \psi_{k,i} \\ v_i = V_i(t_i) \\ c_i = C_i(\tau_i) \\ \tau_i \geq 0 \\ t_i \geq 0 \\ \psi_{k,i} \geq 0 \\ x_{i,a} \geq 0 \end{array} \right. \tag{6}$$

where $x_i = \sum_a x_{i,a}$ is the total expected number of times $phase_i$ is visited. In this paper, it is assumed that the structure of phase transitions can be represented as a tree (and thus a phase cannot be visited more than once during the mission), and so x_i also represents the probability of visiting $phase_i$.

The constraint $\sum_a x_{j,a} = \alpha_j + \sum_i \sum_a p_{i,a,j} \times x_{i,a}$ indicates that the expected number of times $phase_j$ is visited must equal the initial probability distribution at $phase_j$ plus the expected number of times $phase_j$ is entered via all possible transitions [4, 10].

The objective function $\sum_i x_i \times (v_i - c_i)$ in Eq.6, which represents the total expected quality, is a quadratic function (since x_i, v_i and c_i are all variables). In general, quadratic optimization problems are difficult to solve. In the rest of this section, we will show how to reformulate Eq.6 into a mixed integer linear program with the discretization approximation technique.

EVC function-related parameters $T^v_{i,j}$, $V_{i,j}$ and $\delta^v_{i,j}$ were defined in Section 2. We now define deliberation cost function-related parameters $T^c_{i,j}$, $C_{i,j}$ and $\delta^c_{i,j}$ similarly, i.e., $T^c_{i,j}$ and $C_{i,j}$ represent the jth time point and its corresponding value on the discretized function of $C_i(\tau)$ respectively, and binary variable $\delta^c_{i,j}$ represents whether time point $T^c_{i,j}$ is selected, and so Eq.6 can be approximated as the following mixed integer linear program.

$$\max \sum_i \sum_j (\chi^v_{i,j} \times V_{i,j} - \chi^c_{i,j} \times C_{i,j}) \quad \left| \begin{array}{l} \sum_a x_{j,a} = \alpha_j + \sum_i \sum_a p_{i,a,j} \times x_{i,a} \\ x_i = \sum_a x_{i,a} \\ t_i = \sum_{k \in \mathcal{A}_i} \psi_{k,i} \\ \tau_k \geq \sum_{i \in \mathcal{D}_k} \psi_{k,i} \\ t_i = \sum_j T^v_{i,j} \times \delta^v_{i,j} \\ \sum_j \delta^v_{i,j} = 1 \\ \sum_j \chi^v_{i,j} = x_i \\ \chi^v_{i,j} \leq \delta^v_{i,j} \\ \tau_i = \sum_j T^c_{i,j} \times \delta^c_{i,j} \\ \sum_j \delta^c_{i,j} = 1 \\ \sum_j \chi^c_{i,j} = x_i \\ \chi^c_{i,j} \leq \delta^c_{i,j} \\ \psi_{k,i} \geq 0 \\ \tau_i \geq 0, \ t_i \geq 0 \\ x_{i,a} \geq 0 \\ \chi^v_{i,j} \geq 0, \ \chi^c_{i,j} \geq 0 \\ \delta^v_{i,j} \in \{0,1\}, \ \delta^c_{i,j} \in \{0,1\} \end{array} \right. \tag{7}$$

The constraints $\sum_j \delta^v_{i,j} = 1$ (where $\delta^v_{i,j}$ are binary vari-

ables) and $\chi^v_{i,j} \leq \delta^v_{i,j}$ ($\chi^v_{i,j} \geq 0$) indicate that, for each $phase_i$, there exists at most one nonzero variable $\chi^v_{i,j}$, and the constraint $\sum_j \chi^v_{i,j} = x_i$ says that this nonzero variable must equal x_i. All these constraints work together to guarantee $\chi^v_{i,j} = x_i \times \delta^v_{i,j}$. In a similar way, we can have $\chi^c_{i,j} = x_i \times \delta^c_{i,j}$.

Now, we can linearize the quadratic objective function in Eq.6. That is,

$$\begin{array}{rl} & \sum_i x_i \times (v_i - c_i) \\ \simeq & \sum_i \sum_j (x_i \times V_{i,j} \times \delta^v_{i,j} - x_i \times C_{i,j} \times \delta^c_{i,j}) \\ = & \sum_i \sum_j (\chi^v_{i,j} \times V_{i,j} - \chi^c_{i,j} \times C_{i,j}) \end{array}$$

which is the linear objective function (where $V_{i,j}$ and $C_{i,j}$ are constants) used in Eq.7.

$\psi_{k,i}$ in the solution of Eq.7 represents the deliberation schedule, and the policy that maps each phase to its action can be derived from $x_{i,a}$: at $phase_i$, action a is executed with probability $\pi_{i,a} = \frac{x_{i,a}}{\sum_a x_{i,a}}$.

5. EMPIRICAL RESULTS

It is important to remember that the run time of the deliberation scheduling algorithm itself consumes computational time that could otherwise be used for deliberation. In other words, if an agent spends too much time scheduling deliberations, then it might have too little time to actually deliberate. In this section, we give a preliminary empirical evaluation of the efficiency and optimality of our algorithms.

In our simulation, we use continuous concave functions $V_i(t) = M \times (1 - e^{-K \times t})$ to evaluate the LP-based algorithm, and use general nonlinear functions $V_i(t) = \frac{Q}{1+e^{-J \times (t-D)}}$ to evaluate the MILP-based algorithm. In both cases, continuous convex functions $C_i(\tau) = C \times \tau^N$ are used as our deliberation cost functions. The results shown in this paper are based upon the following parameters: $M \sim [0.5, 5.0]$, $K \sim [0.05, 0.5]$, $Q \sim [0.5, 5.0]$, $J \sim [1.0, 2.0]$, $D \sim [1.0, 3.0]$, $C \sim [0.05, 0.5]$, and $N \sim [1.3, 1.6]$, where $x \sim [L, U]$ represents that x is uniformly distributed in the range $[L, U]$. These parameter distributions are chosen to avoid overly-simple problems, such as the problem where the deliberation costs are so high that it is obvious that none of deliberations should be done. We run our simulation with *cplex* 9.1 LP/MILP solver (www.ilog.com) on a Pentium IV machine. Each data point is the average value from 100 runs, and curves are smoothed to improve readability.

Figure 7 shows the scalability of our algorithms in solving deliberation scheduling problems with deterministic phase transitions, where m in the figure denotes the number of pieces for each function if using piecewise linear approximation, and denotes the number of points for each function if using discretization. The y-axis of the figure specifies the total amount of runtime, including the time for making piecewise linear approximation or discretization, and the time for constructing and solving an LP/MILP. We can see that, although slower than the myopic algorithm that attempts to maximize $V_i(t) - C_i(t)$ at each individual phase without worrying about future phases, our algorithms compute near-optimal solutions reasonably fast, especially when m is small, and their solutions significantly outperform that of the myopic algorithm as shown in Figure 8.

Not surprisingly, using a small m will reduce the approximation accuracy, and thus impair optimality. In Figure 8, we evaluate the optimality of our algorithms with various

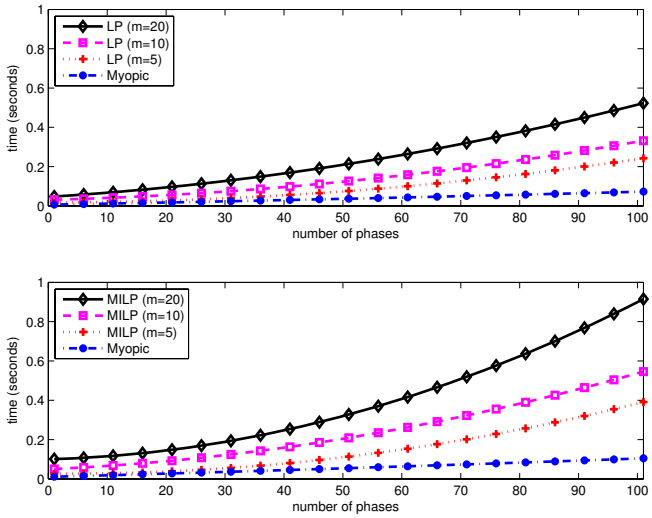


Figure 7: Scalability of the LP-based algorithm (top) and the MILP-based algorithm (bottom) for deterministic phase transitions.

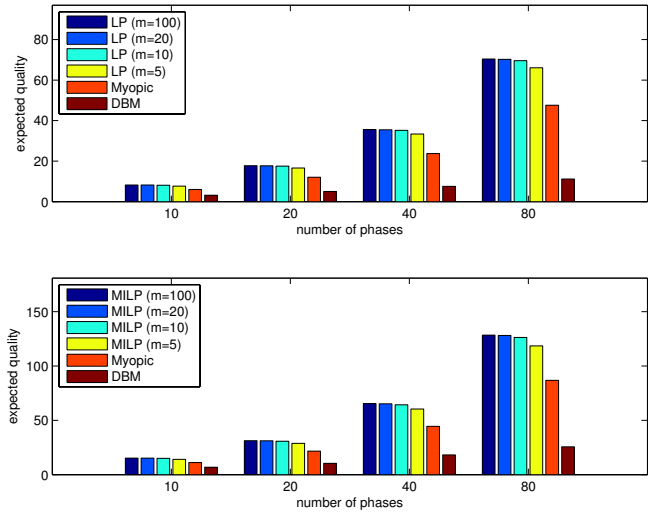


Figure 8: Optimality of the LP-based algorithm (top) and the MILP-based algorithm (bottom) for deterministic phase transitions.

m, the myopic algorithm, and a simple deliberation-before-mission (DBM) algorithm where all deliberations are done before starting the mission (i.e., $\tau_0 \geq 0$ and $\tau_{i \in \{1,2,...\}} \equiv 0$). Since, to the best of our knowledge, there are no existing algorithms that are able to compute optimal deliberation schedules in our experimental settings, we use the solution of our algorithms with a large m (i.e., $m = 100$, which usually makes the approximation error less than 1%) as the baseline. These empirical results show that, with $m = 20$, our algorithms are close to optimal. It is also shown that our algorithms, even with a small m (such as m=5), outperform the myopic algorithm and the DBM algorithm.

When testing our algorithms in problems with uncertain

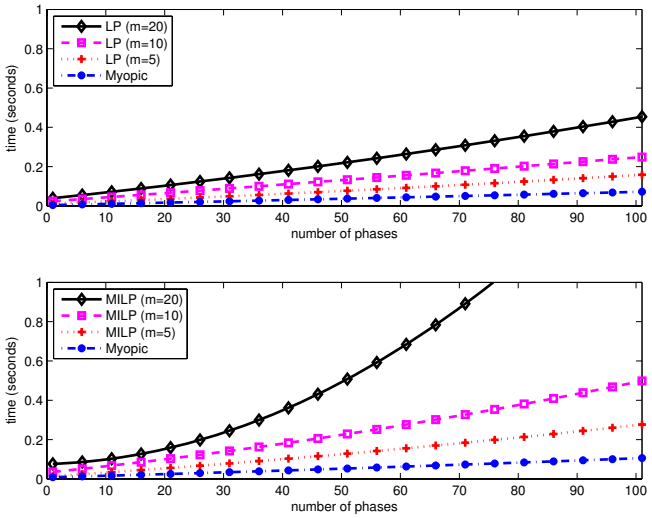


Figure 9: Scalability of the LP-based algorithm (top) and the MILP-based algorithm (bottom) for uncertain phase transitions.

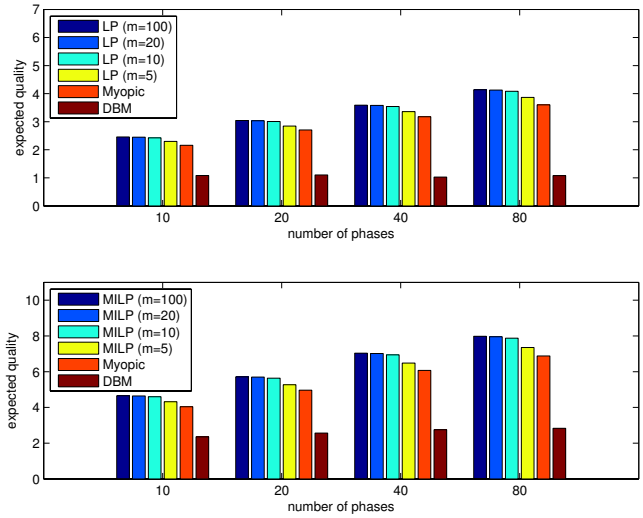


Figure 10: Optimality of the LP-based algorithm (top) and the MILP-based algorithm (bottom) for uncertain phase transitions.

phase transitions, we assume that the phase transitions can be represented as a complete binary tree. When the agent leaves a phase with 2 children, it will reach its left child and its right child with probability ρ and $1-\rho$, respectively, where ρ is uniformly distributed in the range [0, 1.0]. Figure 9 shows the scalability of our algorithms, which is similar to Figure 7.[1]

It might be interesting to see that the myopic algorithm also performs pretty well in Figure 10. This is because, for uncertain phase transitions, the time spent on a future phase is less valuable since it is possible that the agent will never reach that future phase. In other words, in such cases, the agent is more intent to act myopically. However, it is important to emphasize that the myopic algorithm does not provide any quality guarantee. In Figure 11, we can see that the myopic algorithm might return unsatisfactory results sometimes (e.g., less than 60% of the optimal quality). In contrast, the performance of our algorithms is much more stable.

Finally, we evaluate our algorithm in solving problems with controllable phase transitions. It is again assumed that the phase transitions can be represented as a complete binary tree. At each phase with two children, there are two possible actions a_1 and a_2. a_1 moves the agent to the left child with probability 0.9 and to the right child with probability 0.1, and a_2 achieves the opposite effect. Note that problems with controllable phase transitions are in general much more complex than those with deterministic or uncertain phase transitions. When the number of phases is large, the MILP-based algorithm might need a pretty long time to compute an optimal deliberation schedule. However, state-of-the-art MILP solvers (such as cplex) are usually able to return a good solution using much less time. Thus, for online applications, we can adopt a two-step algorithm. That is, we first derive a policy by solving Eq.7 with limited time. With the policy, the problem is reduced to an easier one with deterministic/uncertain phase transitions, and then we can solve the reduced problem again and return a deliberation schedule. As shown in Figure 12, this two-step algorithm is usually able to compute a good schedule within a short time, which makes it applicable in computational-time limited domains.

[1]It should be noted that, for uncertain phase transitions, if neither EVC functions nor deliberation cost functions can be approximated into piecewise linear constraints, it might take a pretty long time to compute an optimal schedule. However, state-of-the-art MILP solvers are usually able to find a good result within limited time.

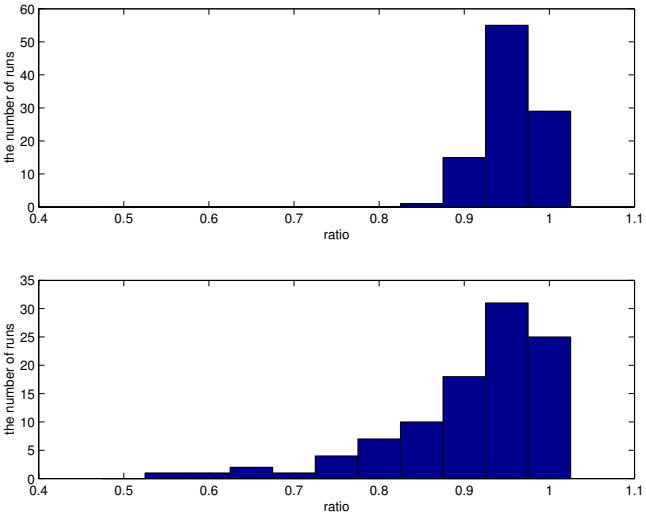


Figure 11: Histograms of ratios of the computed quality to the optimal quality for the LP-based algorithm with $m = 5$ (top) and the myopic algorithm (bottom) in problems with uncertain phase transitions.

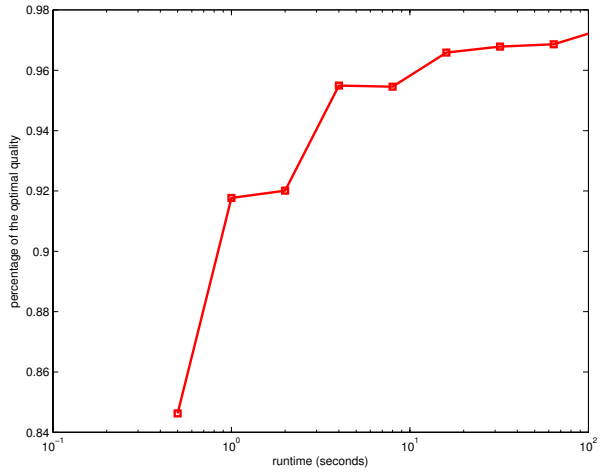


Figure 12: Anytime performance of the two-step algorithm for controllable phase transitions (30 phases and $m = 10$).

6. CONCLUSION AND FUTURE WORK

In this paper, we have presented mathematical programming formulations for deliberation scheduling problems in computational time-limited domains, and have illustrated them through several increasingly complex classes of deliberation scheduling problems. In comparison with previous work, our algorithms are able to solve deliberation scheduling problems in more general settings, and are applicable in complex domains with uncertainty.

The key contribution of this work has been to integrate a variety of established ideas in mathematical programming into a novel approach for solving deliberation scheduling problems. Because most interesting deliberation scheduling problems involve nonlinear constraints, a fundamental technical hurdle that had to be overcome in this work has been to identify transformations that approximate these nonlinear constraints with linear constraints. The deliberation scheduling problems are then solved with existing highly-optimized LP/MILP techniques, rather than building algorithms from scratch that work on the original formulation. By linking deliberation scheduling with LP/MILP, this work lays the foundation for further efforts of exploiting sophisticated techniques in state-of-art LP/MILP solvers [3, 9] for deliberation scheduling problems.

The deliberation scheduling framework studied in this work can be extended to suit more general problems. First, in real-time domains with a large number of phases, a complete mathematical formulation might be too complex to be solvable under time constraints. Considering the fact that probabilities of reaching phases in the far future are often low due to uncertainties, we can restrict the number of phases to be put into the mathematical formulation. As time passes, deliberation scheduling can be repeated if necessary. Second, recall that this work makes a simplifying assumption that the quality of a phase is only determined by the computational time scheduled for it. This is not always true in real-world problems. For example, $phase_i$ might also gain utility when allocating time on another (relevant) $phase_j$ because $phase_i$ and $phase_j$ might have some similar subproblems. Thanks to the mathematical formulation, it is often easy to model such relations (e.g., $v_i = V_i(t_i + t_j)$). Finally, with techniques of limiting look-ahead and modeling inter-influence between phases, we might be able to solve deliberation scheduling problems with loops in phase transitions by *unrolling* the graph to make it acyclic (such that a phase can be repeated in the graph). In the future, we will dig deeper in these directions.

This work, like much previous research, assumes that the agent has required knowledge (such as performance profiles) *a priori*. We will look into the problem where an agent needs to acquire such information in practice, and study how the accuracy of the information affects the system performance. Additionally, in recent years, multi-agent deliberation scheduling problems have been of increasing interest, in which an agent can compute ways to improve solutions to its own problems, or joint problems with other agents. Extending our algorithm to multi-agent domains is thus another promising future direction.

7. ACKNOWLEDGEMENTS

This material is based upon work supported in part by the DARPA/IPTO COORDINATORs program and the Air Force Research Laboratory under Contract No. FA8750–05–C–0030. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

8. REFERENCES

[1] M. S. Boddy and T. Dean. Solving time-dependent planning problems. In *IJCAI*, pages 979–984, 1989.

[2] M. S. Boddy and T. Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artif. Intell.*, 67(2):245–285, 1994.

[3] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley & Sons, New York, 1998.

[4] D. A. Dolgov and E. H. Durfee. Constructing optimal policies for agents with constrained architectures. Technical Report CSE-TR-476-03, University of Michigan, 2003.

[5] R. P. Goldman, D. J. Musliner, and K. D. Krebsbach. Managing online self-adaptation in real-time environments. In *IWSAS*, pages 6–23, 2001.

[6] E. Horvitz. Principles and applications of continual computation. *Artif. Intell.*, 126(1-2):159–196, 2001.

[7] D. J. Musliner, R. P. Goldman, and K. D. Krebsbach. Deliberation scheduling strategies for adaptive mission planning in real-time environments. In M. Anderson and T. Oates, editors, *Metacognition in Computation*, volume SS-05-04 of *AAAI Technical Report*, pages 98–105. AAAI Press, 2005.

[8] M. J. D. Powell. *Approximation Theory and Methods*. Cambridge University Press, Cambridge UK, 1981.

[9] L. A. Wolsey. *Integer Programming*. John Wiley & Sons, New York, 1998.

[10] J. Wu and E. H. Durfee. Automated resource-driven mission phasing techniques for constrained agents. In *AAMAS*, pages 331–338, 2005.

Optimal Decision-Making With Minimal Waste: Strategyproof Redistribution of VCG Payments

Ruggiero Cavallo
DEAS, Harvard University
33 Oxford St.
Cambridge, MA 02138
cavallo@eecs.harvard.edu

ABSTRACT

Mechanisms for coordinating group decision-making among self-interested agents often employ a trusted center, capable of enforcing the prescribed outcome. Typically such mechanisms, including the ubiquitous Vickrey Clarke Groves (VCG), require significant transfer payments from agents to the center. While this is sought after in some settings, it is often an unwanted cost of implementation. We propose a modification of the VCG framework that—by using domain information regarding agent valuation spaces—is often able to achieve redistribution of much of the required transfer payments back among the agents, thus coming closer to budget-balance. The proposed mechanism is strategyproof, ex post individual rational, no-deficit, and leads to an efficient outcome; we prove that among all mechanisms with these qualities and an anonymity property it is *optimally balanced*, in that no mechanism ever yields greater payoff to the agents. We provide a general characterization of when strategyproof redistribution is possible, and demonstrate specifically that substantial redistribution can be achieved in allocation problems.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—Multiagent systems; J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Algorithms, Economics, Theory

Keywords

Social choice, Mechanism design, VCG

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

This paper is concerned with decision-making problems that involve a group of self-interested parties, when each party may be in competition with other members of the group. Instances of this problem abound in everyday life. Consider, to name just a few examples: a group of housemates that jointly own a car and must decide who should get to use it on a Friday night; a set of city neighborhoods competing for a grant to build a new playground; a group of astronomers competing for the rights to use a publicly-owned space telescope; or companies competing for government allocation of wireless spectrum.

How would one formulate a general framework for decision-making in domains such as these? One natural objective is that the decision process lead to the choice that maximizes the total value that is realized. Moreover, and to that end, it is highly desirable that the mechanism not encourage manipulation by dishonest agents—we want *truthfulness* to be in each agent's best-interest; this will allow us to reasonably predict the actual effects of whatever outcome is selected, and will simplify the task of the agents by taking strategic behavior out of the equation that maximizes expected reward.

The well-known Groves family of social choice mechanisms, by requiring specific payments to each agent, reaches outcomes with the desirable properties described above. However, implementing the payments specified in the most basic Groves mechanism requires a large external budget, making it infeasible in typical decision-making scenarios where there is no outside funding.

The Vickrey Clarke Groves (VCG) mechanism counteracts this potential budget imbalance by imposing "charges" on the agents, to be delivered to a "center" capable of enforcing the specified outcome. For typical decision-making problems,[1] VCG results in net transfers being delivered only in the direction of agents to the center. While there are some scenarios in which revenue to a center is sought after (e.g., some auctions), often it is preferable to maintain as much wealth as possible within the group of agents, and the transfer payment can be viewed as an undesirable "cost of implementation." In other words, *budget-balance* is frequently a sought after mechanism property, and though it doesn't run a deficit, the VCG mechanism fails in this regard by producing a surplus of agent charges (hence called the "VCG surplus").

[1]Specifically, when there are no positive externalities.

The ideal social choice mechanism would have the desirable properties of VCG (truthfulness; social welfare maximization; non-negative payoff guarantee), and at the same time run neither a budget surplus nor deficit. As we will see, this *exact* budget-balance is usually not attainable. However, while it has often been stated that no improvement over VCG is possible, we demonstrate that this *is not the case* in a broad class of domains (e.g., allocation problems) where valuations have some basic structure.

We propose a modification of the VCG framework that incorporates redistribution of as much of the VCG surplus as possible back among the agents. We prove that—among all truthful, social welfare maximizing, and no-deficit mechanisms that meet basic anonymity and participation constraints—this "redistribution mechanism" is *optimally balanced*, in that no mechanism ever comes closer to budget-balance. We describe why in the case of completely unconstrained valuation functions no redistribution is possible, and characterize the domains in which it is.

We start by focusing our analysis on the special setting of "all-or-nothing" (AON) domains, those in which each outcome yields non-zero reward for just a single agent (every agent gets either all the reward or none). Here the redistribution mechanism has a particularly simple and elegant form, and the most redistribution is possible. In fact, in AON domains the mechanism is *completely* budget-balanced (i.e., redistributes all surplus) in the limit as the number of agents goes to infinity, for arbitrary valuations in an AON context. We demonstrate the applicability of the mechanism to general allocation problems, and describe empirical results that indicate the extent to which redistribution is possible for different degrees of dependency between agent valuations. Finally, we discuss implementation of the mechanism in the absence of a central enforcer.

2. BACKGROUND AND RELATED WORK

2.1 Notation

- Let I be the set of agents participating in a mechanism.
- Let $n = |I|$, the number of agents.
- Let O be the set of possible outcomes.
- Let $v_i(o)$ denote the *true* value to i of outcome $o \in O$.
- Let $\hat{v}_i(o)$ denote the *reported* value to agent i of outcome o.
- Let o^* denote the socially optimal outcome, according to reported valuations, i.e.:

$$o^* = \arg\max_{o \in O} \sum_{i \in I} \hat{v}_i(o)$$

- Let o^*_{-i} denote the outcome that maximizes payoff among agents other than i, according to reports, i.e.:

$$o^*_{-i} = \arg\max_{o \in O} \sum_{j \in I \setminus \{i\}} \hat{v}_j(o)$$

2.2 Mechanism design background

The field of mechanism design is concerned with reaching outcomes that meet certain system-level criteria, despite the assumption that individuals will pursue only their self-interest. A *mechanism* amounts to a specification of a decision rule, potentially accompanied by a set of requirements (typically monetary transfers) placed on participants. A mechanism that maximizes social welfare is termed *socially optimal* or *efficient*. When no agent can ever benefit from lying, i.e., when being truthful is a dominant strategy, a mechanism is called *strategyproof*. A somewhat weaker property is *incentive-compatibility* (IC); in an IC mechanism truthfulness is a Nash equilibrium: no agent can benefit from being dishonest when all other agents are truthful.

The Groves class of mechanisms is *efficient* and *strategyproof* [Groves, 1973]. A Groves mechanism chooses the outcome that maximizes reported reward across all agents, and makes a transfer payment to each agent equal to the reward received by all other agents. In the most basic version, each agent's payoff is equal to the reward received by the entire system; clearly such a mechanism would require a massive budget to implement. Fortunately, the Groves framework allows for the imposition of a charge on each agent, which does nothing to diminish the mechanism's desirable incentive properties so long as each agent's charge is completely beyond their control. Formally, a Groves mechanism chooses the outcome o^* that maximizes social welfare according to reported rewards, and defines the following transfer payment T_i to each agent i:

$$T_i = \sum_{j \in I \setminus \{i\}} \hat{v}_j(o^*) - C_{-i}$$

where C_{-i} can be *any quantity independent of agent i*.

In order to ensure that all agents want to participate in a mechanism, it is desirable that none ever be worse for having done so. A mechanism that provides this guarantee to every agent, for any possible set of agent valuation functions, is referred to as *ex post individual rational (IR)*. A mechanism that yields non-negative payoff in expectation is termed *ex ante IR*. To obtain ex post IR in the Groves framework, we should ensure that for each i:

$$v_i(o^*) + \sum_{j \in I \setminus \{i\}} \hat{v}_j(o^*) \geq C_{-i}$$

An important special case of the Groves class is the VCG mechanism [Vickrey, 1961; Clarke, 1971; Groves, 1973]. VCG defines C_{-i} above to be $\sum_{j \in I \setminus \{i\}} \hat{v}_j(o^*_{-i})$, the reported reward that all other agents would have received if i were not a participant.[2] Thus VCG yields the following payoff, π_i, for each agent i:

$$\begin{aligned} \pi_i &= v_i(o^*) + T_i \\ &= v_i(o^*) + \sum_{j \in I \setminus \{i\}} \hat{v}_j(o^*) - \sum_{j \in I \setminus \{i\}} \hat{v}_j(o^*_{-i}) \end{aligned}$$

VCG is ex post IR, and is also *no-deficit*: the sum of payments made from agents to the center is never less than 0. But it is often greater—each agent's net payment to the center is always between 0 and the reward it reported for the selected outcome, depending on the reported valuations of the other agents. In the case of AON domains, VCG is known simply as the Vickrey or second-price auction, and there is only one payment made—the winning agent must pay the second highest "price".

Consider the following 4-agent AON example, where outcome o_i corresponds to the decision that yields value for agent i:

[2]Note that since all Groves mechanisms are strategyproof, $\hat{v}_j$ will equal v_j in dominant strategy equilibrium.

	v_1	v_2	v_3	v_4
o_1	10	0	0	0
o_2	0	8	0	0
o_3	0	0	5	0
o_4	0	0	0	4

Table 1: 4-agent AON decision problem.

VCG chooses outcome o_1, as agent 1's value (10) is highest, and the following payoffs result:

$$\pi_i = v_i(o^*) + \sum_{j \in I \setminus \{i\}} \hat{v}_j(o^*) - \sum_{j \in I \setminus \{i\}} \hat{v}_j(o^*_{-i})$$

$$\pi_1 = 10 + 0 - 8 = 2$$

$$\pi_2 = 0 + 10 - 10 = 0$$

$$\pi_3 = 0 + 10 - 10 = 0$$

$$\pi_4 = 0 + 10 - 10 = 0$$

$$\text{total payment to center} = 8$$

In the case of an auction, this rather large payment of 8 may be desirable—it can go to the seller that is providing the item. But for a general social choice or decision problem, it may merely be an unwanted side-effect of choosing to implement VCG, the most famous strategyproof and efficient mechanism. In this work we explore the possibility of returning the surplus to the agents.

DEFINITION 1. **(Redistribution mechanism)** *An efficient social choice procedure that seeks to minimize net transfers from agents to an external body by return of VCG surplus to the agents.*

Our characterization of efforts to achieve minimal surplus in terms of redistribution is motivated by the centrality (and ubiquity) of VCG. In seeking to redistribute as much of the VCG surplus as possible (without violating truthfulness), we are effectively searching among all IR and no-deficit instances of the Groves class of mechanisms for the one that maximizes payoff to the agents. Green & Laffont [1977] showed that for general valuations, the only efficient and strategyproof mechanisms are the members of the Groves class. Holmstrom [1979] strengthened this result significantly, proving that Groves is also unique in any restricted domain in which each agent's space of potential valuations is *smoothly connected*, i.e., when for any two valuations an agent could report, one can be differentiably deformed into the other. As Holmstrom notes, this result shows that "for all practical purposes" one must be content with Groves. Thus, our approach is comprehensive and will yield uniqueness properties for the mechanism we propose.

2.3 Related work

Work in the area of designing budget-balanced mechanisms is relatively scarce, presumably due to the primacy of the goal of social efficiency in combination with the strong negative result of the *Myerson-Satterthwaite impossibility theorem*: that no mechanism is capable of achieving individual rationality, efficiency, and budget balance at the same time for general valuation functions, even if we loosen our solution concept to Bayes-Nash equilibrium [Myerson and Satterthwaite, 1983]. This is an extension of the Hurwicz impossibility theorem [Hurwicz, 1975], which proves the result for dominant-strategy equilibria.

One proposal for achieving budget-balance is the d'AGVA mechanism [d'Aspremont and Gerard-Varet, 2002]. d'AGVA selects the efficient outcome according to reported types, but determines transfer payments based on a model of agent valuations that the center maintains. This mechanism is interesting primarily because it leads to exact budget-balance; however, it is implementable only in Bayes-Nash (rather than dominant-strategy) equilibrium, and if the center's model is produced via iterative execution, serious problems regarding incentive compatibility can arise. Moreover, d'AGVA is only ex ante IR—instances in which an agent is worse off for participating are possible.

Parkes et al. [2001] describe a payment rule that approximates VCG and achieves perfect budget-balance in exchange settings where VCG runs a deficit, though truthfulness (and thus efficiency) is sacrificed. Faltings [2004] formulates the problem more closely to the way we do in this work, though his approach also attains exact budget-balance at the expense of efficiency. His mechanism chooses the outcome that is socially efficient among a subset of agents, and distributes the VCG surplus among agents that are not part of that subset. In a similar vein, Feigenbaum et al. [2001] analyze the Shapley-value mechanism for sharing multicast transmission costs, which comes closest to the efficient outcome among all budget-balanced mechanisms for that domain, though their results suggest its implementation is computationally intractable. Most similar to our proposal is that of Bailey [1997]. He specifies an efficient mechanism that refunds surplus to participants, but he forfeits the no-deficit guarantee in order to achieve a mechanism that is budget-balanced in expectation.

Our approach is significantly different than those described above—we characterize the extent to which budget-balance can be approximated (and sometimes reached) in dominant strategies, without sacrificing individual-rationality, efficiency, or no-deficit guarantees. Rather than giving up one or another of these properties, we are able to achieve budget-balance superiority over VCG *by using domain information that constrains the space of possible valuations.*

3. REDISTRIBUTION IN AON DOMAINS

It is instructive to start with the special case of AON domains, for which our solution is especially simple and elegant, as it provides good intuition for the general case. Formally, we define an AON domain as one in which the number of outcomes is equal to the number of agents, and each agent i's valuation function v_i is known to be constrained in the following way: $v_i(o_j) = 0, \forall j \neq i$. We identify o_i with the unique outcome that may yield positive value to agent i, and use the following short-hand notation:

- Let a_c denote the agent with the cth highest reported value. (So in a welfare-maximizing mechanism a_1 is the "winning" agent—the one that receives positive value from the selected outcome.)
- Let $\mathbb{V}_{a_c}$ denote the *true* value to a_c of the outcome he favors (i.e., $v_{a_c}(o_{a_c})$), and $\hat{\mathbb{V}}_{a_c}$ the *reported* value (i.e., $\hat{v}_{a_c}(o_{a_c})$).

We propose the following redistribution mechanism for AON domains that maintains much of the VCG surplus among the agents:

DEFINITION 2. **(Redistribution mechanism RM)**

1. Each agent i communicates to the center $\mathbb{v}_i$, the value he places on the outcome under which he obtains non-zero reward.
2. The center implements the outcome that maximizes social welfare.
3. The winner (a_1) pays each agent (including himself) a VCG surplus redistribution of Z_i, and pays the center $\hat{\mathbb{v}}_{a_2} - \sum_{i \in I} Z_i$, where

$$Z_i = \begin{cases} \frac{\hat{\mathbb{v}}_{a_3}}{n} & \text{for } i = a_1, a_2 \\ \frac{\hat{\mathbb{v}}_{a_2}}{n} & \text{for } i = a_3, ..., a_n \end{cases}$$

THEOREM 1. **RM** *is ex post individual-rational in AON domains.*

PROOF. The payoff to each agent for the implemented outcome in **RM** is:

$$\pi_{a_1} = \mathbb{v}_{a_1} - \hat{\mathbb{v}}_{a_2} + Z_{a_1}$$
$$\pi_{j \neq a_1} = Z_j$$

The payoff to any "non-winner" j is always the non-negative Z_j payment term. When the mechanism is followed, $\mathbb{v}_{a_1} = \hat{\mathbb{v}}_{a_1}$; then since $\hat{\mathbb{v}}_{a_1} \geq \hat{\mathbb{v}}_{a_2}$ by definition, and since $Z_{a_i} \geq 0$, a_1's payoff is also always non-negative. □

THEOREM 2. **RM** *is strategyproof in AON domains.*

PROOF. **RM** differs from the VCG mechanism only in the Z payment term—each agent receives the standard VCG payoff plus a non-negative Z payment. Since we know VCG is strategyproof, it suffices to show that no agent i can increase its Z_i payment term by misreporting its reward. a_k below denotes the agent with the kth highest bid under truthful reporting.

Agent a_1 receives redistribution payment (from himself) $\hat{\mathbb{v}}_{a_3}/n$. Over-reporting $\hat{\mathbb{v}}_{a_1}$ changes nothing. Under-reporting could put a_1 in the second or third position (beyond the third it's obvious that nothing could change). In the second position, he would still receive $\hat{\mathbb{v}}_{a_3}/n$. In the third he would receive $\hat{\mathbb{v}}_{a_3}/n$ as well, since the second position would then be held by the actual a_3.

Agent a_2 receives redistribution payment $\hat{\mathbb{v}}_{a_3}/n$. Over-reporting could move him to the first position, in which case his payment would be the same. Under-reporting could put him in the third position or beyond, but then the second position would be held by the actual a_3, so a_2's payoff would be the same.

Agent a_3 receives redistribution payment $\hat{\mathbb{v}}_{a_2}/n$. Under-reporting changes nothing. Over-reporting could put him in the first or second position. In both cases he receives the same Z payment, since the third position would then be held by the actual a_2. The same holds for all $a_{j>3}$. □

Theorems 1 and 2 together show that **RM** for AON domains always achieves the efficient outcome in dominant strategy equilibrium.

THEOREM 3. *As the number of participating agents n goes to ∞, the amount of extracted wealth that cannot be redistributed among the agents under* **RM** *goes to 0. That is,* **RM** *is asymptotically budget-balanced for AON domains.*

PROOF. The amount of wealth redistributed among the agents is:

$$\sum_{i \in I} Z_i = \frac{n-2}{n} \cdot \hat{\mathbb{v}}_{a_2} + \frac{2}{n} \cdot \hat{\mathbb{v}}_{a_3}$$

The amount not redistributed is the payment to the center:

$$\begin{aligned} Z_c &= \hat{\mathbb{v}}_{a_2} - \sum_{i \in I} Z_i \\ &= \hat{\mathbb{v}}_{a_2} - \frac{n-2}{n} \cdot \hat{\mathbb{v}}_{a_2} - \frac{2}{n} \cdot \hat{\mathbb{v}}_{a_3} \\ &= \frac{2}{n} \cdot (\hat{\mathbb{v}}_{a_2} - \hat{\mathbb{v}}_{a_3}) \end{aligned}$$

We assume existence of some finite bound on valuations, and thus on $\hat{\mathbb{v}}_{a_2}$. □

As n increases, we may expect Z_c to be pushed down by a convergence of $\mathbb{v}_{a_2}$ and $\mathbb{v}_{a_3}$, but regardless of this **RM** achieves perfect budget balance in the limit. *VCG will always lose $\hat{\mathbb{v}}_{a_2}$, no matter the number of agents.*

Consider once more the example illustrated in Table 1. The following payoffs are obtained under **RM**:

$$\pi_1 = 10 - 8 + \frac{5}{4} = \frac{13}{4}$$
$$\pi_2 = \frac{5}{4}\,, \quad \pi_3 = \frac{8}{4}\,, \quad \pi_4 = \frac{8}{4}$$
$$Z_c = 8 - \left(\frac{5}{4} + \frac{5}{4} + \frac{8}{4} + \frac{8}{4}\right) = \frac{3}{2}$$

Even in this example with just 4 agents, the vast majority (81%) of the VCG surplus has been redistributed. If $\mathbb{v}_{a_3}$ were 8 rather than 5, 100% would be redistributed.

3.1 Optimality of RM

We will now characterize the relationship between **RM** and all other possible redistribution mechanisms through a series of incremental observations. We seek the optimal way of redistributing the VCG payment back among agents in AON domains. More specifically, we wish to achieve the following:

DEFINITION 3. **(Optimally balanced mechanism)** *A mechanism is optimally balanced for domain D if—among all efficient, no-deficit, and ex post IR mechanisms that can be implemented in dominant strategy equilibrium—it maintains the most wealth within the set of agents, for every instance $d \in D$.*

A domain D specifies a space of possible valuations for every agent, and an instance $d \in D$ may be any realization of actual valuations consistent with D.

LEMMA 1. *In any strategyproof redistribution mechanism for smoothly connected valuation spaces, for every agent i, the amount of surplus redistributed to i, Z_i, must be independent of the valuation function i reports.*

PROOF. The Lemma follows from uniqueness of the Groves class. For intuition, consider an agent i with arbitrary valuation $v_i = \{v_i(o_1), ..., v_i(o_n)\}$ yielding payment Z_i. If there is some outcome o_k and $\epsilon \in \Re$ such that reporting $\hat{v}_i(o_k) = v_i(o_k) + \epsilon$ yields $Z_i + \gamma$ for some $\gamma > 0$ without changing the selected outcome, agent i will misreport to maximize payment. □

To achieve efficiency in dominant strategy equilibrium, a redistribution mechanism must be an instantiation of the Groves class, so the redistribution can be considered part of the necessarily agent-independent Groves "charge" function.

Given Lemma 1, it will be useful now to consider each agent's redistribution payment as a function of the reported rewards of all other agents in the system. So we have, $\forall_{i \in I}$:

$$Z_i = f_i(\hat{\mathbb{V}}_0, \hat{\mathbb{V}}_1, ..., \hat{\mathbb{V}}_{i-1}, \hat{\mathbb{V}}_{i+1}, ..., \hat{\mathbb{V}}_n)$$

for some f_i. The task of constructing a redistribution mechanism can then be viewed as the specification of n functions, each of which is defined over a distinct set of $n-1$ of the valuation reports. It's important to note that the above characterization is comprehensive, as each f_i is defined over all variables in the system save i's report, which it can't depend on by Lemma 1.

But it is often preferred (or even required) that we do not discriminate amongst agents based on personal identity; we often want to implement mechanisms that are *anonymous*:

DEFINITION 4. **(Anonymous mechanism)** *A mechanism that chooses an outcome and defines transfer payments (including redistribution) according to a single deterministic function that is invariant to domain information that does not apply identically to every agent.*

In other words, anonymity requires that transfer payments be computed independent of "personal identity". To construct an anonymous redistribution mechanism, then, in fact we define only a single function f that is used universally, but applied to a distinct set of $n-1$ parameters for each agent. We know that an *optimally balanced* anonymous redistribution mechanism exists, since to define it amounts to specifying the value of f, for every possible set of $n-1$ values, that is greatest while still ensuring that no deficit could result for any possible instantiation of the excluded n^{th} value.

Let $p_i^{(k)}$ denote the kth largest parameter to the function that defines Z_i. $p_i^{(2)}$ is then the second highest bid among all agents other than i, and is also the greatest lower-bound (*guarantee*) on the VCG surplus that will exist, independent of the valuation i reports.

LEMMA 2. *The optimally balanced anonymous redistribution mechanism for AON domains must define redistribution payments Z_i, $\forall_{i \in I}$, that are monotonically increasing in $p_i^{(2)}$.*

PROOF. Suppose the Lemma didn't hold. Then there would be reported reward profiles A and $B \in \Re^n$ with $\hat{\mathbb{V}}_{a_2}(A) \leq \hat{\mathbb{V}}_{a_2}(B)$, yet with $Z(A) > Z(B)$. In such a case the mechanism could not be optimal, as implementing $Z(A)$ in case B would keep more wealth among the agents. □

THEOREM 4. **RM** *is the optimally balanced anonymous mechanism for AON domains.*

PROOF. Any anonymous, efficient, no-deficit, and ex post IR redistribution mechanism defines payments $Z_i \leq \frac{p_i^{(2)}}{n}$, $\forall_{i \in I}$. Consider the parameters of f_i that would lead to Z_i being $> \frac{p_i^{(2)}}{n}$. Simply note that there exists a problem instance such that all $p_j^{(2)}$, $\forall_{j \in I}$ are identical. This, when viewed in light of Lemma 2, shows that if Z_i were $> \frac{p_i^{(2)}}{n}$, the mechanism would not be guaranteed IR and no-deficit, as $\sum_j Z_j$ could then be $> \hat{\mathbb{V}}_{a_2}$.

Now observe that another way of describing **RM** is as follows: redistribute a portion of $\hat{\mathbb{V}}_{a_2}$ to each agent i equal to $\frac{1}{n}$ times the value of the second highest reported reward of agents other than i. □

4. REDISTRIBUTION IN THE GENERAL CASE

We have thus far only considered redistribution of VCG payments in AON domains. In this section we explore the general conditions under which strategyproof redistribution is possible, and characterize how to redistribute surplus optimally.

We first show that in the case of domains in which value functions are completely unrestricted, redistributing any of the transfer payments while maintaining the desirable properties of VCG is impossible. One way of understanding this is to consider the problem of finding a lower-bound on the total amount of VCG surplus that is guaranteed to exist, independent of any given agent's reported valuation.

LEMMA 3. *In any efficient, strategyproof, ex post IR and no-deficit mechanism, an upper-bound on the redistribution payment to any agent i is the minimum VCG surplus that could be realized, taken over all possible instantiations of $\hat{v}_i$.*

PROOF. Among all efficient, strategyproof, and ex post IR mechanisms, VCG yields the greatest surplus [Krishna and Perry, 1998]. Thus a mechanism with these properties that redistributes a quantity greater than the VCG surplus will run a deficit, since charging any agent more would violate ex post IR.

Let $\underline{\hat{v}}_i$ be the instantiation of $\hat{v}_i$ that results in minimum VCG surplus, S_i. Suppose for contradiction that i receives redistribution payment $Z_i > S_i$. By Lemma 1, Z_i must be the same regardless of the valuation i reports; thus when i reports $\underline{\hat{v}}_i$ he will have been redistributed a quantity greater than the total VCG surplus, and the system will have run a deficit. □

Lemma 3 points to a way of generally characterizing when redistribution is and is not possible. Whether or not a VCG surplus will result is intimately connected to the *similarity* of agents' valuations. In the extreme, all agents may favor the same outcome, in which case there will be no surplus. In some domains, e.g., AON, we know a priori that agents' valuations are distinct; such distinctions are in fact what enables redistribution. We now formalize these intuitions.

DEFINITION 5. **(Potential for Universal Relevance Nullification [PURN])** *An agent i has the potential for universal relevance nullification, given the valuation reports of the other agents, if i could report a valuation $\hat{v}_i$ that would yield $o^* = o^*_{-j}$, $\forall_{j \in I}$ (including i).*

The *PURN* property indicates that an agent could potentially report a value that renders null the influence on the outcome of any single agent's report alone, even his own. *PURN* is often dependent on the values other agents have reported, and is then a property of one agent's valuation domain combined with the other agents' actual reports.

THEOREM 5. *No efficient, strategyproof, ex-post IR, and no-deficit mechanism redistributes any surplus to an agent in an instance in which* PURN *holds for that agent.*

PROOF. The total VCG surplus is:

$$\sum_{j\in I}\Big(\sum_{k\in I\setminus\{j\}}\hat{v}_k(o^*_{-j})-\sum_{k\in I\setminus\{j\}}\hat{v}_k(o^*)\Big)$$

Here we sum over all agents $j \in I$ the VCG surplus j creates: the difference between j's VCG "charge" delivered to the center and the Groves payment he receives. By definition, any agent i with *PURN* may report a $\hat{v}_i$ that would lead to this quantity being zero. The theorem follows directly from this and Lemma 3. □

LEMMA 4. *Any agent with an unconstrained valuation always has PURN, regardless of the other agents' valuations.*

PROOF. Let i be an agent with an unconstrained valuation. Consider the outcome o^*_{-i} that maximizes utility among the $n-1$ other agents. Let i's valuation be as follows: $\hat{v}_i(o^*_{-i}) = \max_{j\in I\setminus\{i\}}\hat{v}_j(o^*_{-i})$, and $\hat{v}_i(o) = 0,\ \forall_{o\neq o^*_{-i}}$. Considering this full set of n valuations, the overall efficient outcome o^* will be identical to $o^*_{-j},\ \forall_{j\in I}$; this is precisely the condition for *PURN*. □

PROPOSITION 1. *The VCG mechanism with no redistribution is optimally balanced for unconstrained valuations.*

PROOF. When valuations are unconstrained, by Lemma 4 each agent has *PURN*. By Theorem 5, then, no agent can receive any redistribution payment. □

While this is a strong negative result, its applicability is hardly universal; in practice no valuation function is *entirely* unconstrained. Specifically, in determining the payment Z_i for any agent i, v_i need not be treated as a complete unknown. The AON case, for instance, is not caught by Proposition 1 because we need only consider valuations that are non-zero for just a single outcome.

In any given domain there may be several ways in which valuations are known to be restricted. Even the mere existence (and knowledge) of distinct finite bounds on each agent's valuation for any outcome can be enough to enable redistribution. *PURN* can be seen as one extreme in a spectrum of possible relationships between agent valuation spaces—rather than having the capacity to *completely* nullify the relevance of other agents in determining the outcome, an agent may have the potential to diminish others' impact only partially, or perhaps not at all. This capacity is of interest to us here because of the bearing it has on the amount of VCG surplus that will result. *Our ability to redistribute VCG surplus to an agent is directly tied to the extent to which we can know surplus will exist independent of that agent's reported valuation.*

DEFINITION 6. **(Surplus-guarantee S_i)** *The lower-bound on VCG surplus computed over all possible instantiations of $\hat{v}_i$, i.e., the greatest surplus guarantee that is independent of agent i.*

$$S_i = \min_{\hat{v}_i}\Big[\sum_{j\in I}\Big(\max_{o'\in O}\Big\{\sum_{k\in I\setminus\{j\}}\hat{v}_k(o')\Big\}-\sum_{k\in I\setminus\{j\}}\hat{v}_k(o^\dagger)\Big)\Big] \quad (1)$$

$$= \min_{\hat{v}_i}\Big[\sum_{j\in I}\Big(\max_{o'\in O}\sum_{k\in I\setminus\{j\}}\hat{v}_k(o')\Big)-(n-1)\Big(\sum_{j\in I}\hat{v}_j(o^\dagger)\Big)\Big]$$

subject to the constraint that the $o^\dagger$ and $\hat{v}_i$ in the above minimization must satisfy the following:

$$o^\dagger = \arg\max_{o\in O}\sum_{j\in I}\hat{v}_j(o) \quad (2)$$

Breaking down equation (1), within brackets is an expression representing the net payment—summed over all agents—to the center under VCG; this is the surplus we'll have to redistribute. For each agent i, we take the minimum of this expression over all possible valuations i could report.

Now note that a payment based on S_i will not always be *completely* anonymous, since we're using information about valuation constraints that are potentially unique to each agent. Specifically, it will not be anonymous in domains that are not *symmetric*.[3] But interestingly, it turns out that a generalization of mechanism **RM** is optimal not only across the space of completely anonymous mechanisms, but also among mechanisms that meet just the following weaker requirement:

DEFINITION 7. **(Surplus-anonymous redistribution mechanism)** *A mechanism that chooses an outcome and maps agent-specific VCG surplus lower-bounds (S_i) to redistribution payments (Z_i) according to a single deterministic function that is invariant to domain information that does not apply identically to every agent.*

In a *surplus-anonymous* redistribution mechanism, then, for any two agents i and j: if $S_i = S_j$, then $Z_i = Z_j$.

THEOREM 6. *For all domains with smoothly connected valuation spaces, the redistribution mechanism* $\mathbf{RM_g}$ *that modifies the VCG mechanism by paying each agent i a quantity equal to $\frac{S_i}{n}$ is strategyproof, and is optimally balanced across all surplus-anonymous mechanisms.*

PROOF SKETCH. Strategyproofness follows immediately from strategyproofness of VCG plus the fact that S_i is independent of i's reported valuation. The argument for optimality is omitted here for lack of space. □

Any *completely-anonymous* mechanism is also *surplus-anonymous*, and the conditions for *complete-anonymity* and *surplus-anonymity* coincide in symmetric domains (e.g., AON). $\mathbf{RM_g}$ is thus *completely-anonymous* when applied to symmetric domains, and *surplus-anonymous* otherwise. It is easy to verify that the AON mechanism **RM** is a special case of $\mathbf{RM_g}$—S_i in an AON domain is equal to the second highest of the bids placed by agents other than i.

It can be shown that no mechanism, including $\mathbf{RM_g}$, is *optimally-balanced* when anonymity considerations are completely dropped. For instance, a mechanism that redistributes S_i to a randomly selected agent i, and none to other agents, will in some cases redistribute more surplus than $\mathbf{RM_g}$; however, such a mechanism is clearly not *optimally-balanced*.

4.1 Redistribution in Allocation Problems

We've been discussing redistribution in the context of general social choice or decision-making problems. An important subclass of social choice problems consists of those in which a decision must be made on how to allocate goods amongst a group of competing agents. The most basic allocation problem is that in which a single item is available; this limited case (usually) falls within the AON class.[4] Generally, an allocation problem consists of a number of sellers

[3]A *symmetric domain* is one in which the space of possible valuation reports is identical for each agent.

[4]The exception is when there are externalities, e.g., when agent i is happy if his friend agent j is allocated the item.

who bring goods to the system, and a number of interested buyers, each with preferences over the goods that are potentially combinatorial in nature.

Sometimes in such domains allocation mechanisms are selected to maximize revenue to the seller(s), but there are many significant cases in which it is preferable to keep as much wealth as possible in the hands of the agents. For example, consider the allocation of job time on a publicly owned super-computer among a community of researchers. Resources like this are often established with the mandate of maximizing benefit to the public good (efficiency); if a VCG-based allocation were implemented a surplus would result, fair redistribution of which would go further toward satisfying this mandate.

In the vast majority of allocation problems, the following are generally accepted to hold: agents receive no value if they aren't allocated anything (normalization); agents' values monotonically increase as they receive more goods (free disposal); and agents have no preferences over allocations to other agents (no externalities). These intrinsic elements of the allocation domain map directly to constraints on valuation functions that can allow for significant strategyproof redistribution of VCG surplus. Formally, let G be the set of goods to be allocated, and for any bundle of goods $b \subset G$ let $v_i(b)$ be agent i's value for obtaining b. Each agent i's valuation conforms to the following:

$$v_i(\emptyset) = 0$$
$$v_i(b) \leq v_i(b \cup \{g\}), \quad \forall_{b \subset G,\, g \in G}$$

Consider the following valuations of three agents in an allocation problem with two goods, A and B, where $\{X_1, X_2, X_3\}$ represents the outcome in which bundles X_1, X_2, and X_3 are allocated to agents 1, 2, and 3 respectively.

	v_1	v_2	v_3
$\{AB, \emptyset, \emptyset\}$	12	0	0
$\{\emptyset, AB, \emptyset\}$	0	10	0
$\{\emptyset, \emptyset, AB\}$	0	0	11
$\{A, B, \emptyset\}$	4	6	0
$\{A, \emptyset, B\}$	4	0	5
$\{B, A, \emptyset\}$	5	7	0
$\{B, \emptyset, A\}$	5	0	7
$\{\emptyset, A, B\}$	0	7	5
$\{\emptyset, B, A\}$	0	6	7

Table 2: 2-good, 3-agent allocation problem.

The efficient outcome is to allocate A to agent 3 and B to agent 2. Applying $\mathbf{RM_g}$, agent payoffs are as follows:

$$\pi_i = v_i(o^*) + \sum_{j \in I \setminus \{i\}} \hat{v}_j(o^*) - \sum_{j \in I \setminus \{i\}} \hat{v}_j(o^*_{-i}) + Z_i$$
$$\pi_1 = 0 + 13 - 13 + \frac{8}{3} = \frac{8}{3}$$
$$\pi_2 = 6 + 7 - 12 + \frac{11}{3} = \frac{14}{3}$$
$$\pi_3 = 7 + 6 - 12 + \frac{10}{3} = \frac{13}{3}$$

Using VCG with no redistribution, total social utility is 2 and payment to the center is 11. 9.67 of this (88% of surplus) can be redistributed to the agents under $\mathbf{RM_g}$, yielding a social utility of 11.67, a nearly 5-fold improvement.

5. SIMULATIONS

5.1 Computing redistribution payments

Determining redistribution payments under $\mathbf{RM_g}$ amounts to computing surplus guarantee S_i for each agent i, and then merely dividing by n. In a broad range of domains, a simple algorithm for computing S_i exists. For example, in AON and in fact any typical[5] allocation domain, S_i can be computed as follows: consider v_i to be as high as possible for outcome o^*, and 0 for all other outcomes; then compute the VCG surplus with this v_i and all other agents' reported valuations. This simple algorithm does not hold in general (i.e., not for *all* possible sets of value constraints); however, we can *always* compute S_i through a mixed-integer programming (MIP) specification of equation (1). We briefly outline the formulation here.

The objective in the MIP for agent i's payment is to minimize VCG surplus, and the primary variables are $v_i(o)$ for each $o \in O$. Representing the program constraints is relatively straightforward, but some care must be taken in handling equation (2) and the inner maximization in (1). In both cases there are non-linearities. In (2), for instance, we must specify $c_o \cdot v_i(o)$ for each outcome o, where c_o is a boolean variable representing whether or not o is chosen as the outcome that maximizes social welfare. We can get around this issue by representing $c_o \cdot v_i(o)$ with a new variable $v'_i(o)$, and including the following constraints:

$$v'_i(o) \leq v_i(o)$$
$$v'_i(o) \leq c_o \cdot M$$

where M is a value larger than the maximum possible value an agent could have for any outcome. $v'_i(o)$ will then be $v_i(o)$ if $o^* = o$, and 0 otherwise, as desired.

While solving a mixed-integer program has exponential worst-case running time, in practice we were able to quickly find solutions to very large problems. Determining redistribution payments in a 100 agent, 100 outcome problem took 24 seconds for each agent.[6] Note that the MIPs for calculating agent payments (one for each agent) are independent of each other, and thus all can be solved in parallel.

5.2 Numerical results

In order to understand how much redistribution can be achieved for valuations under different levels of mutual constraint, we performed an empirical analysis on large sets of randomly generated problem instances (sets of valuations), each with the same number of outcomes as agents. We generated valuations according to the following process, where e is an "exclusivity" (between agent valuations) parameter representing the extent to which a domain has "all-or-nothing properties." We chose a maximum value for each agent i's valuation function uniformly at random between 0 and 100. We then chose each $v_i(o_i)$ uniformly at random between 0 and $maxval_i$, and $v_i(o_{j \neq i})$ uniformly at random between 0 and $(1 - e) \cdot maxval_i$. So when $e = 1$, we have a completely AON domain; when $e = 0$ there are no exclusivity constraints between agent valuations. The graph

[5]One without valuation space constraints beyond the standard normalization, free disposal, and no externalities assumptions.

[6]Solutions were obtained using the commercial solver CPLEX, run on a 1.6 GHz Pentium 4 PC.

below plots the percentage of VCG surplus that could be redistributed as a function of the parameter e, for problems with various numbers of agents. For each number of agents, 100 samples were computed for each value of e between 0 and 1 in increments of 0.05.

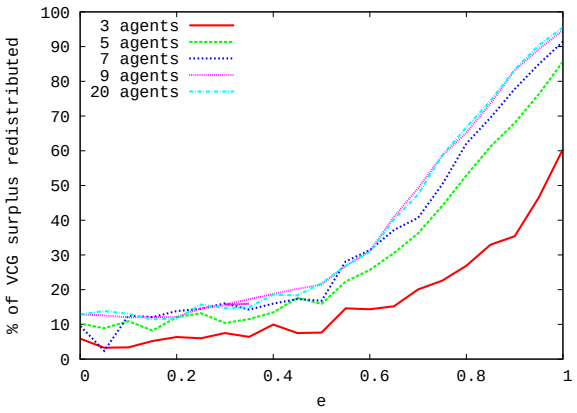


Figure 1: Percent of surplus redistributed as a function of mutual "exclusivity" between agent valuations.

Notably, redistribution remains nearly constant for values of e between 0 and 0.5, and then increases roughly linearly with e from 0.5 to 1. As expected, the possibility for redistribution grows with the number of agents.

6. A CENTER-LESS IMPLEMENTATION

Consider the problem of building a decision-making mechanism that doesn't require a center. This is intuitively desirable, as it's easy to imagine scenarios in which a group of agents must determine an allocation or decide on an action to take without the benefit of a trusted central authority.[7]

THEOREM 7. **RM$_\mathbf{g}$** *can be implemented in Nash equilibrium without the facilitation of a center when the following conditions hold:*

i) no agent acting alone has the power to obstruct realization of a specified outcome or transfer payments, while $n-1$ agents acting together do.
ii) agents can simultaneously "broadcast" valuations to all other agents, and can compute and perform transfers.
iii) agents have the capacity to publicly destroy money.

PROOF SKETCH. The first condition brings abiding by the mechanism into Nash equilibrium—unilaterally deviating cannot be beneficial. The second and third conditions ensure that agents will be able to execute what the mechanism prescribes. With (ii), all agents can recognize the efficient outcome, and deliver the appropriate redistribution payment to each "receiving" agent. (iii) allows the "paying" agents (just a_1 in AON domains) to demonstrably receive the appropriate payoff. The quantity Z_c that cannot stay within the group of agents would normally be transfered to a center, but its destruction is a satisfactory substitute. □

In AON domains, at least, where communicating a valuation amounts merely to a public announcement of a single (value, outcome) pair, satisfaction of these conditions is quite plausible. Then using **RM**, it is possible in equilibrium for a large group of self-interested agents to independently reach the socially optimal outcome, and jointly reap nearly all fruits of the chosen action via redistribution.

[7]See [Shneidman and Parkes, 2004; Petcu *et al.*, 2006] for further discussion of distributed implementations that seek to minimize the role of a center.

7. CONCLUSION

In this paper we argued for the desirability of redistributing VCG payments back among participants in a mechanism, and showed it is feasible in a broad range of settings, including allocation problems. We presented the *optimally balanced* redistribution mechanism, that which always redistributes the maximal amount that any strategyproof, efficient, IR, and no-deficit mechanism could. In "all-or-nothing" domains, a class which encompasses all typical single-item allocation problems, the mechanism is perfectly budget-balanced in the limit.

The VCG mechanism maximizes transfers *from* the group of agents, which is sometimes appealing because transfers amount to revenue for a center. But in many decision problems the transfers are, in a sense, actually waste; in this paper we specified the mechanism that minimizes that waste.

Acknowledgments

I'd like to thank David C. Parkes for inspiration, as well as very helpful suggestions and comments on a draft.

8. REFERENCES

[Bailey, 1997] Martin J. Bailey. The demand revealing process: To distribute the surplus. *Public Choice*, 91:107–126, 1997.

[Clarke, 1971] Edward Clarke. Multipart pricing of public goods. *Public Choice*, 8:19–33, 1971.

[d'Aspremont and Gerard-Varet, 2002] Claude d'Aspremont and Louis-Andre Gerard-Varet. Incentives and incomplete information. In *Game Theory in the Tradition of Bob Wilson*, 2002.

[Faltings, 2004] Boi Faltings. A budget-balanced, incentive-compatible scheme for social choice. In *Agent-mediated E-commerce AMEC*, 2004.

[Feigenbaum *et al.*, 2001] Joan Feigenbaum, Christos H. Papadimitriou, and Scott Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63(1):21–41, 2001.

[Green and Laffont, 1977] Jerry Green and Jean-Jacques Laffont. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica*, 45:427–438, 1977.

[Groves, 1973] Theodore Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[Holmstrom, 1979] Bengt Holmstrom. Groves' scheme on restricted domains. *Econometrica*, 47(5):1137–1144, 1979.

[Hurwicz, 1975] Leonid Hurwicz. On the existence of allocation systems whose manipulative nash equilibria are pareto optimal. *(presented at the 3rd World Congress of the Econometric Society)*, 1975.

[Krishna and Perry, 1998] Vijay Krishna and Motty Perry. Efficient mechanism design. Game theory and information, Economics Working Paper Archive at WUSTL, December 1998.

[Myerson and Satterthwaite, 1983] Roger Myerson and Mark A Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 28:265–281, 1983.

[Parkes *et al.*, 2001] David C. Parkes, J. R. Kalagnanam, and M. Eso. Achieving budget-balance with vickrey-based payment schemes in exchanges. In *Proc. 17th Int. Joint Conf. on Artificial Intelligence*, pages 1161–1168, 2001.

[Petcu *et al.*, 2006] Adrian Petcu, Boi Faltings, and David C. Parkes. MDPOP: Faithful distributed implementation of efficient social choice problems. In *Proc. 5th Int. Joint Conf. on Autonomous Agents and Multi Agent Systems*, 2006.

[Shneidman and Parkes, 2004] Jeffrey Shneidman and David C. Parkes. Specification faithfulness in networks with rational nodes. In *Proc. 23rd ACM Symp. on Principles of Distributed Computing*, 2004.

[Vickrey, 1961] William Vickrey. Counterspeculations, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

Distributed Lagrangean Relaxation Protocol for the Generalized Mutual Assignment Problem

Katsutoshi Hirayama
Kobe University
5-1-1 Fukaeminami-machi, Higashinada-ku
Kobe 658-0022, Japan
hirayama@maritime.kobe-u.ac.jp

ABSTRACT

We present a new formulation of distributed task assignment, *Generalized Mutual Assignment Problem* (GMAP), which is derived from a typical NP-hard combinatorial optimization problem that has been studied for many years mainly in the operations research community. Then, to solve the GMAP, we introduce a novel distributed solution protocol using *Lagrangean decomposition* and *distributed constraint satisfaction*, where the agents solve their individual optimization problems and coordinate their locally optimized solutions through a distributed constraint satisfaction technique. Next, to produce quick agreement between the agents on a feasible solution with reasonably good quality, we provide a parameter that controls the range of "noise" mixed with increment/decrement in a Lagrangean multiplier and report our experimental results indicating that the parameter may allow us to control tradeoffs between the quality of a solution and the cost of finding it.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Coherence and coordination*

General Terms

Algorithms

Keywords

distributed combinatorial optimization

1. INTRODUCTION

The distributed task assignment problem, which concerns assigning a set of jobs to multiple agents, has been one of the main research issues from the early days of distributed problem solving [9]. Recently, several studies have been reported in which they formalize the problem as the optimization problem to be solved by some general solution protocol. [3, 6, 7].

In this paper, we present a new formulation of distributed task assignment which is derived from the *Generalized Assignment Problem* (GAP). The GAP is a typical NP-hard combinatorial optimization problem that has been studied for many years mainly in the operations research community [1, 8]. The goal of the GAP is to find an optimal assignment of jobs to agents such that a job is assigned to exactly one agent and the assignment satisfies all of the resource constraints imposed on individual agents.

To adapt the GAP for a distributed context, we first introduce the *Generalized Mutual Assignment Problem* (GMAP). The GMAP is the distributed problem in which each agent in the GAP has a set of jobs and tries to achieve the goal of the GAP. To put it another way, the agents themselves try to generate an optimal solution of the GAP from a non-optimal (and sometimes infeasible) solution of it. Then, to solve the GMAP, we introduce a novel distributed solution protocol using *Lagrangean decomposition* [4] and *distributed constraint satisfaction* [10]. In this protocol, the agents solve their individual optimization problems and coordinate their locally optimized solutions through a distributed constraint satisfaction technique. One important feature of this protocol is that, unlike other optimization protocols for distributed task assignment, it is a pure distributed protocol where neither the server nor the coordinator exists.

2. FORMALIZATION

A GMAP instance consists of multiple agents each having a finite set of jobs that are to be assigned. The agents as a whole solves the following *integer programming problem* denoted as $\mathcal{GAP}$.

$$
\begin{aligned}
\mathcal{GAP} \quad & (\text{decide } x_{kj},\ \forall k \in A,\ \forall j \in J) : \\
\text{max.} \quad & \sum_{k \in A} \sum_{j \in J} p_{kj} x_{kj} \\
\text{s. t.} \quad & \sum_{k \in A} x_{kj} = 1, \ \ \forall j \in J, && (1) \\
& \sum_{j \in J} w_{kj} x_{kj} \leq c_k, \ \ \forall k \in A, && (2) \\
& x_{kj} \in \{0, 1\},\ \forall k \in A,\ \forall j \in J, && (3)
\end{aligned}
$$

where $A = \{1, ..., m\}$ is a set of agents; $J = \{1, ..., n\}$ is a set of jobs; p_{kj} and w_{kj} are a profit and an amount of resource required, respectively, when agent k selects job j;

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

c_k is a capacity, an amount of available resource, of agent k. Also, x_{kj} is a decision variable whose value is set to 1 when agent k selects job j and 0 otherwise. A goal of the agents is to find a job assignment that maximizes a total sum of profits such that (1) each job is assigned to exactly one agent (denoted as *assignment constraints*), (2) a total amount of resource required for each agent does not exceed its capacity (denoted as *knapsack constraints*), and (3) each job is assigned or not assigned to an agent (denoted as *01 constraints*). We refer to the maximal total sum of profits as the *optimal value* and the assignment that provides the optimal value as an *optimal solution*.

We can obtain the following Lagrangean relaxation problem $\mathcal{LGAP}(\mu)$ by dualizing assignment constraints of $\mathcal{GAP}$ [2].

$$\begin{aligned}
\mathcal{LGAP}(\mu) \quad & (\text{decide } x_{kj},\ \forall k \in A,\ \forall j \in J) : \\
\text{max.} \quad & \sum_{k\in A}\sum_{j\in J} p_{kj}x_{kj} + \sum_{j\in J}\mu_j\left(1 - \sum_{k\in A} x_{kj}\right) \\
\text{s. t.} \quad & \sum_{j\in J} w_{kj}x_{kj} \le c_k,\ \forall k \in A, \\
& x_{kj} \in \{0,1\},\ \forall k \in A,\ \forall j \in J,
\end{aligned}$$

where μ_j is a real-valued parameter and generally called a *Lagrangean multiplier* for (the assignment constraint of) job j. Note that a vector $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$ is called a *Lagrangean multiplier vector*. Based on the idea of *Lagrangean decomposition* [4], we can divide this problem to produce a set of subproblems $\{\mathcal{LGMP}_k(\mu) | k \in A\}$, where each subproblem $\mathcal{LGMP}_k(\mu)$ of agent k is as follows.

$$\begin{aligned}
\mathcal{LGMP}_k(\mu) \quad & (\text{decide } x_{kj}, \forall j \in R_k) : \\
\text{max.} \quad & \sum_{j\in R_k} p_{kj}x_{kj} + \sum_{j\in R_k}\mu_j\left(\frac{1}{|S_j|} - x_{kj}\right) \\
\text{s. t.} \quad & \sum_{j\in R_k} w_{kj}x_{kj} \le c_k, \\
& x_{kj} \in \{0,1\},\ \forall j \in R_k,
\end{aligned}$$

where R_k is a set of jobs that may be assigned to agent k and S_j is a set of agents to whom job j may be assigned. We can assume $S_j \neq \emptyset$ because the job with an empty S_j does not have to be considered in the problem. When we consider that agent k decides the value of x_{kj} for $\forall j \in R_k$ (in other words, the agent that may be assigned job j has a right to decide whether it will undertake the job or not), agent k is able to solve $\mathcal{LGMP}_k(\mu)$ completely by itself since $\mathcal{LGMP}_k(\mu)$ includes only k's decision variables.

On the relation between the entire problem and a set of subproblems, there exist the following properties which are very useful in designing a distributed solution protocol for the GMAP.

PROPERTY 1. *For any values for μ, a total sum of the optimal values of $\{\mathcal{LGMP}_k(\mu)|k \in A\}$ provides an upper bound of the optimal value of $\mathcal{GAP}$.*

PROPERTY 2. *For any values for μ, if all of the optimal solutions to $\{\mathcal{LGMP}_k(\mu)|k \in A\}$ satisfy the assignment constraints, $\sum_{i\in A} x_{ij} = 1$, $\forall j \in J$, then these optimal solutions constitute an optimal solution to $\mathcal{GAP}$.*

3. PROTOCOL

Property 2 prompted the development of a distributed solution protocol, which is called the *distributed Lagrangean relaxation protocol*, where the agents start with $t = 0$ and $\mu^{(0)} = (0, \ldots, 0)$; then, they alternate the following series of actions (called a *round*) in parallel until all of the assignment constraints are satisfied.

1. Each agent k finds an optimal solution to $\mathcal{LGMP}_k(\mu^{(t)})$ (solves the *primal problem*).
2. Agents exchange these solutions with their neighboring agents.
3. Each agent k updates the Lagrangean multiplier vector from $\mu^{(t)}$ to $\mu^{(t+1)}$ (solves the *dual problem*) and increases t by one.

It must be noted that the overall behavior of this protocol is similar to the *distributed breakout algorithm* for solving the DisCSP [5]. We describe the key ideas of the protocol in this section.

3.1 Neighborhood and Communication Model

In order to solve primal and dual problems, agent k needs, for each job j in R_k, the values of Lagrangean multiplier $\mu_j^{(t)}$, the size of set S_j, and the decision variables of the related agents (i.e., $\{x_{ij} \,|\, i \in S_j, i \neq k\}$). The value of $\mu_j^{(t)}$ is locally computed as we will explain later, and the size of set S_j is given to agent k as prior knowledge. On the other hand, the values of the decision variables are obtained through communication with a set of agents denoted as $\bigcup_{j\in R_k} S_j \setminus \{k\}$. We refer to this set of agents as agent k's *neighbors* and allow an agent to communicate only with its neighbors. Thus, the protocol assumes a peer-to-peer message exchange model, in which a message is never lost and, for any pair of agents, messages are received in the order in which they were sent.

3.2 Primal Problem

Once agent k determines the values of the associated Lagrangean multipliers, agent k searches for an optimal solution to $\mathcal{LGMP}_k(\mu^{(t)})$ to determine the values of its own decision variables $\{x_{kj} \,|\, j \in R_k\}$. This search problem is equivalent to the *knapsack problem* whose goal is to select the most profitable subset of R_k such that the total resource requirement of the subset does not exceed c_k. In the problem, for each job j in R_k, the profit is $p_{kj} - \mu_j^{(t)}$, and the resource requirement is w_{kj}.

After finding an optimal solution, agent k sends the solution to its neighbors via an *assign* message.

3.3 Dual Problem

After receiving all of the current solutions of neighbors, agent k solves the dual problem by updating the related Lagrangean multipliers. In this work we use a *subgradient optimization method*, a well-known technique for systematically updating a Lagrangean multiplier vector [2]. In the method, under the current values of the related decision variables, agent k first computes *subgradient* g_j for an assignment constraint of each job $j \in R_k$ as follows:

$$g_j = 1 - \sum_{i\in S_j} x_{ij}.$$

Then, using $|S_j|$ and *step length* $l^{(t)}$, which may decay at rate r $(0 < r \leq 1)$ as the number of rounds t proceeds (i.e., $l^{(t+1)} \leftarrow r l^{(t)}$), agent k updates the multiplier of job j as

$$\mu_j^{(t+1)} \leftarrow \mu_j^{(t)} - l^{(t)} \frac{g_j}{|S_j|}. \quad (4)$$

Notice that since multiplier μ_j is attached to job j all of the agents in S_j must agree on a common value for it. If the agents in S_j assign different values to μ_j, then neither property 1 nor 2 holds any more. To set such a common value, we give all of the agents a common initial value for μ, a common value for initial step length $l^{(0)}$, and a common value for decay rate r as their prior knowledge, and prohibit each agent from working at round $t+1$ until it receives all of the assign messages issued from its neighbors at round t. By doing this, instead of implementing explicit communication among S_j, we can force agents to automatically set a common value to a Lagrangean multiplier.

3.4 Convergence Detection

The protocol should be terminated when all of the optimal solutions to $\{\mathcal{LGMP}_k(\mu^{(t)}) \,|\, k \in A\}$ satisfy all of the assignment constraints, but it is not so simplistic to detect this fact because no agent knows the entire solving processes. In this protocol we use a termination detection procedure of the distributed breakout algorithm [5] to detect this fact.

3.5 Convergence to Feasible Solution

Unfortunately, the protocol may be trapped in an infinite loop depending on the parameter values in rule (4). If that case, the protocol must be forced to terminate after a certain number of rounds, but it has not yet discovered any feasible solution on the way to an optimal solution. Therefore, we make rule (4) stochastic so that the protocol can break an infinite loop and produce quick agreement between the agents on a feasible solution with reasonably good quality. More specifically, we let the agents in S_j assign slightly different values to μ_j by introducing a parameter δ $(0 \leq \delta \leq 1)$ that controls the range of "noise" mixed with increment/decrement in a Lagrangean multiplier. The noise, denoted as N_δ, is a random variable whose value is uniformly distributed over $[-\delta, \delta]$. Multiplier updating rule (4) is thus replaced by

$$\mu_j^{(t+1)} \leftarrow \mu_j^{(t)} - (1 + N_\delta) l^{(t)} \frac{g_j}{|S_j|}. \quad (5)$$

This rule diversifies agents' views on the value of μ_j and thus the protocol can break an infinite loop. On the other hand, since properties 1 and 2 do not hold anymore under this rule, the protocol may converge to a non-optimal (but feasible) solution. Note that rule (5) is equal to rule (4) if δ is set to zero.

4. EXPERIMENTS

Through experiments, we observed the actual performance of the protocol when varying the values of δ and the *assignment topologies*. We adopted two problem suites, one is hand-made but has various assignment topologies and the other is from the GAP benchmark instances of OR-Library but has a specific assignment topology.

Consequently, we observed that the performance of the protocol dramatically changed when δ was set to one of the non-zero values. The results showed that the cost got lower while the solution quality was kept at a reasonable level, suggesting that by using the protocol with those settings, the agents can quickly agree on a feasible solution with reasonably good quality.

It is also true that the protocol with those settings may fail to find an optimal solution. In the experiments, it failed to find an optimal solution at every non-zero value of δ for 3 complete topology instances of the hand-made problem suite and 5 instances of the benchmark problem suite.

For almost all of the instances, we observed that when δ increased both the cost and the solution quality were reduced. This means that increasing δ may generally influence the agents to rush to reach a compromise of lower quality. This indicates that parameter δ may allow us to control tradeoffs between the quality of a solution and the cost of finding it.

5. CONCLUSION

We presented a new formulation of distributed task assignment and a novel distributed solution protocol, where the agents solve their individual optimization problems and coordinate their locally optimized solutions. Furthermore, we introduce a parameter of the protocol that may allow us to control tradeoffs between the quality of a solution and the cost of finding it.

6. REFERENCES

[1] D. G. Cattrysse and L. N. V. Wassenhove. A survey of algorithms for the generalized assignment problem. *European J. of Operational Research*, 60:260–272, 1992.

[2] M. L. Fisher. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

[3] B. P. Gerkey and M. J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *Intl. J. of Robotics Research*, 23(9):939–954, 2004.

[4] M. Guignard and S. Kim. Lagrangean decomposition: A model yielding stronger Lagrangean bounds. *Mathematical Programming*, 39:215–228, 1987.

[5] K. Hirayama and M. Yokoo. The distributed breakout algorithms. *Artificial Intelligence*, 161(1–2):89–115, 2005.

[6] E. Kutanoglu and S. D. Wu. On combinatorial auction and Lagrangian relaxation for distributed resource scheduling. *IIE Transactions*, 31(9):813–826, 1999.

[7] T. Nishi, M. Konishi, and S. Hasebe. An autonomous decentralized supply chain planning system for multi-stage production processes. *J. of Intelligent Manufacturing*, 16:259–275, 2005.

[8] I. H. Osman. Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches. *OR Spektrum*, 17:211–225, 1995.

[9] R. G. Smith. The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Trans. on Computers*, 29(2):1104–1113, 1990.

[10] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: formalization and algorithms. *IEEE Trans. on Knowledge and Data Engineering*, 10(5):673–685, 1998.

Auctioning Transformable Goods

Andrea Giovannucci
Juan A. Rodríguez-Aguilar
IIIA-CSIC, Campus UAB, 08193 Bellaterra, Spain
{andrea, jar}@iiia.csic.es

Jesús Cerquides
Dept. de Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Gran Via, 585, 08007, Barcelona, Spain
cerquide@maia.ub.es

ABSTRACT

In this paper we explore whether an auctioneer/buyer may benefit from introducing his transformability relationships (some goods can be transformed into others at a transformation cost) into multi-unit combinatorial reverse auctions. Thus, we quantitatively assess the potential savings the auctioneer/buyer may obtain with respect to combinatorial reverse auctions that do not consider tranformability relationships.

1. INTRODUCTION

Consider a company devoted to sell manufactured goods. It can either buy raw goods from providers, transform them into some other goods via some manufacturing process, and sell them to customers; or it can buy already-transformed products and resell them to customers. Thus, either the company buys raw goods to transform via an in-house process at a certain cost, or it buys already-transformed goods. Figure 1 graphically represents an example of a company's inner manufacturing process, more formally *Transformability Network Structure* (TNS), fully described in [1]. This graphical description largely borrows from the representation of Place/Transition Nets (PTN), a particular type of Petri Net [2]. Each circle (corresponding to a PTN *place*) represents a good. Horizontal bars connecting goods represent manufacturing operations, likewise *transitions* in a PTN. Manufacturing operations are labeled with a numbered *t*, and shall be referred to as *transformation relationships* (*t-relationships* henceforth). An arc connecting a good to a transformation indicates that the good is an *input* to the transformation, whereas an arc connecting a transformation to a good indicates that the good is an *output* from the transformation. In our example, g_2 is an *input good* to t_2, whereas g_6, g_7, and g_8 are *output goods* of t_2. Thus, t_2 represents the way g_2 is transformed. The labels on the arcs connecting *input goods* to transitions, and the labels on the arcs connecting *output goods* to transitions indicate the units required of each *input good* to perform a transformation and the units generated per *output good* respectively. In figure 1, the labels on the arcs connected to t_2 indicate that 1 unit of g_6, 7 units of g_7, and 1 unit of g_8 are obtained after processing 1 unit of g_2. Each transformation has an associated cost every time it is carried out. In our example, transformation t_2 costs €7.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

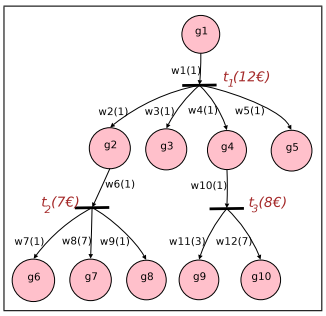


Figure 1: Example of a Transformability Network Structure.

Say that a buying agent requires to purchase a certain amount of goods g_3, g_5, g_6, g_7, g_8, g_9, and g_{10}. For this purpose, it may opt for running a combinatorial reverse auction with qualified providers. But before that, a buying agent may realise that he faces a decision problem: shall he buy g_1 and transform it via an in-house process, or buy already-transformed goods, or opt for a *mixed-purchase* solution and buy some already-transformed goods and some to transform in-house? This concern is reasonable since the cost of g_1 plus transformation costs may eventually be higher than the cost of already-transformed goods.

The work in [1] addresses the possibility of expressing transformability relationships among the different assets to sell/buy on the bid-taker side in a multi-unit combinatorial reverse auction. The new type of combinatorial reverse auction (the Multi-Unit Combinatorial Reverse Auction with Transformability Relationships among Goods (MUCRAtR)) provides to buying agents: (a) a language to express required goods along with the relationships that hold among them; and (b) a winner determination problem (WDP) solver that not only assesses what goods to buy and to whom, but also the transformations to apply to such goods in order to obtain the initially required ones. It is shown that, if the TNS representing the relationships among goods is acyclic, the associated WDP is modeled by the following integer program:

$$min[\sum_{j=1}^{m} x_j p_j + \sum_{k=1}^{r} q_k c(t_k)] \quad (1)$$

$$\forall\, 1 \leq i \leq n \;\; \sum_{j=1}^{m} a_j^i x_j + \sum_{k=1}^{r} q_k m_k^i \geq u_i \quad (2)$$

where $x_j \in \{0,1\}\ \forall\ 1 \leq j \leq m$ stands for whether bid b_j is selected or not, p_j is the price associated to bid b_j, q_k is a decision variable taking into account how many times transformation t_k is fired, a_j^i is the number of units of good i offered in bid b_j, u_i is the number of required unit of good i, m_k^i is obtained from the *incidence matrix* [2] of the place-transition net within a TNS,

$c(t_k)$ stands for the cost associated to transformation t_k, m is the number of bids, n is the number of different negotiated goods, and r is the number of *t-relationships*. Expression (1) minimises the sum of the costs of the selected bids plus the cost of the transformations to apply, and equation (2) enforces that the selected bids plus the transformations applied at least fulfill a buyer's requirements. We will assume a finite production capacity, that is $q_k \in \{0, 1, \ldots, max_k\}, 1 \leq q_k \leq r$.

Notice that the integer program above can be clearly regarded as an extension of the integer program associated to a Multi Unit Combinatorial Reverse Auction (MUCRA) WDP as formalised in [3]. Thus, the second component of expression (1) changes the overall cost as transformations are applied, whereas the second component of expression (2) makes sure that the units of the selected bids fulfill a buyer's requirements, taking into account the units consumed and produced by transformations.

The purpose of this paper to quantitatively assess the potential savings the auctioneer/buyer may obtain with respect to combinatorial reverse auctions that do not consider tranformability relationships.

2. EMPIRICAL EVALUATION

Our experiments artificially generate different data sets. Each data set shall be composed of: (1) a TNS; (2) a Request for Quotations (RFQ) detailing the number of required units per good; and (3) a set of combinatorial bids. Then, we solve the WDP for each auction problem regarding and disregarding *t-relationships*. This is done to quantitatively assess the potential savings that a buyer/auctioneer may obtain thanks to *t-relationships*. Thus, the WDP for a MUCRA will only consider the last two components of the data set, whereas the WDP for a MUCRAtR will consider them all. In order to solve the WDP for a MUCRA we exploit its equivalence with the multi-dimensional knapsack problem [3]. The WDP for a MUCRAtR is modeled by the integer program represented by expressions (1) and (2). In what follows we describe the way to artificially generate such data set.

2.1 Data Set Generation

In order to create a data set, the most delicate task is concerned with the generation of a collection of combinatorial bids. Unfortunately, we cannot benefit from any previous methods for artificially generating auction data sets in the literature since they do not take into account the novel notion of t-relationship.

TNS generation. Firstly, we consider the creation of a TNS. As explained in the introduction, if we restrict to the case of an acyclic TNS, then the WDP for a MUCRAtR can be formulated as an integer program. Thus, we shall focus on generating acyclic TNSs for our data sets. For this purpose, we create TNSs fulfilling the following requirements: (a) each transition receives a single input arc; (b) each place can have no more than one input and one output arc; and (c) there exists a place, called *root place*, that can only have output arcs. Figure 1 depicts an example of a TNS that satisfies such requirements. A distinguishing feature of our algorithm is that, since we aim at empirically assessing the potential savings when considering *t-relationships* independently of TNSs' shapes, it is capable of constructing acyclic TNSs that may largely differ in their shapes, and in the combination of weights assigned to arcs. Our generator randomly constructs TNSs receiving as inputs: (1) a number of places n (the number of goods); (2) a number of *t-relationships* r; (3) the minimum/maximum arc weight w_{min}/w_{max} (each arc weight is chosen from a uniform discrete distribution $U[w_{min}, w_{max}]$); and (4) the minimum/maximum transformation cost c_{min}/c_{max} (a transformation cost for each *t-relationship* is drawn from a uniform distribution $U[c_{min}, c_{max}]$).

RFQ generation. An RFQ is represented as a set $U = \{u_1, \ldots, u_n\}$ where u_i stands for the number of units requested of good g_i. We generate each $u_i \in U$ from a uniform discrete distribution $U[u_{min}, u_{max}]$, where u_{min} and u_{max} stand for the minimum and maximum number of units required per item respectively.

Bid generation. Finally, we complete the artificial generation of a data set by generating a set of plausible bids. Each bid $b_j \in B$ can be represented as a pair $\langle p_j, [a_j^1, \ldots, a_j^n] \rangle$ where p_j stands for the bid price and $[a_j^1, \ldots, a_j^n]$ for the units offered per good. For each bid b_j our generator firstly obtains the number of jointly offered goods from a binomial distribution with parameters ($p_{offered\_goods}$,n), (say z goods); then it randomly selects z goods in G (the set of required goods). For each one of the z selected good g_i, the number of offered units is obtained from another binomial distribution parameterised by ($p_{offered\_units}, u_i$). We employ binomial distributions since our aim is to maintain a proportionality relationships among: (1) the number of negotiated goods and the cardinality of offers; and (2) the number of required units and the number of offered units per good. This is done since we would like to analyse separately the effects of such parameters on savings, and we want to avoid inter-dependency effects. For instance, employing a geometric distribution to describe the number of offered units would implicitly create a dependency effect among the number of required units and the number of offered units, since increasing the number of required units would have the equivalent effect of lowering the number of offered units. Instead, a binomial distribution allows to analise, *ceteris paribus*, the effect of increasing the number of required units.

After generating the units to offer per good for all bids, we must assess all bid prices. This process is rather delicate when considering *t-relationships* if we want to guarantee the generation of plausible bids. In general, it is unrealistic to think of a market scenario wherein raw goods are more expensive than transformed goods. Hence, we assume that all providing agents produce goods in a *similar* manner (they share similar TNSs). However, goods' prices and transformation costs differ from provider to provider. In practice, our providing agents use the same TNS as the buying agent, though each one has his own transformation costs. Thus, for each provider we compute the unitary price for each good in the TNS. Thereafter, for each provider, we use his unitary prices to construct his bids.

Next, we describe how to calculate the unitary prices for each good for a given provider. We depart from the value of the p_{root} parameter, standing for the average unitary price of the root good (e.g. the root good in figure 1 is g_1). The first step of our pricing algorithm calculates the unitary price of the root good for each provider under the assumption that all providers have similar values for such good. Thus, for each provider P_j, his unitary price for the root good is assessed as $\pi_{root,j} = p_{root} \cdot \lambda$, where λ is sampled from a normal distribution $N(\mu_{root\_price}, \sigma_{root\_price})$. After that, our pricing algorithm recursively proceeds as follows. Given a provider and a good whose unitary price has been already computed, this is *propagated down* the provider's TNS through the transition it is linked to towards its output goods. We compute the value to propagate by weighting the unitary price by the value labelling the arc connecting the input good to the transition, and adding the provider's particular transformation cost of the transition. The resulting value is unevenly distributed among the output goods according to a *share factor* randomly assigned to each output good. For instance, consider the TNS in figure 1 and a provider P_j such that its unitary cost for g_2 is $\pi_{g_2,j} = €50$, *his* transformation cost (different from the buying agent's one) for t_2 is €10, and $w_6 = 1$.

Table 1: Parameters characterising our experimental scenario

Parameter	*Explanation*	*Value*
n	The number of items	20
r	The number of transitions	8
u_{min}, u_{max}	The minimum/maximum number of units required per item	10/10
w_{min}, w_{max}	Minimum/Maximum arc weight	1/5
c_{min}, c_{max}	Minimum/Maximum Transformation cost	10/10
m	The number of bids to generate	1000
$p_{offered\_goods}$	Statistically sets the number of items simultaneously present in a bid	0.2 - 0.3 0.4 - 0.5
$p_{offered\_units}$	Statistically sets the number of unit offered per item	0.2 - 0.3 0.4 - 0.5
p_{root}	Average price of the $root$ good	1000
μ_{root_price} σ_{root_price}	Parameters of a Gaussian distribution weighting the $root$ price p_{root}	1 0.01
$\mu_{production\_cost}$ $\sigma_{production\_cost}$	Parameters of a Gaussian distribution setting the production costs difference between buyer and providers	0.8:0.1:1.8 0.1

In such a case, the value to split down through t_2 towards g_6, g_7, and g_8 would be $50 \cdot 1 + 10 = €60$. Say that g_7 is assigned 0.2 as share factor. Thus, $60 \cdot 0.2 = €12$ would be allocated to g_7. Finally, that amount should be split further to obtain g_7 unitary price since $w_8 = 7$. Then, the final unitary price for g_7 is $€1.7142 = \frac{12}{7}$. Hence, we can provide a general way of calculating the unitary price for any good for a given provider. Let P_j be a provider and g a good such that $a_j^g \neq 0$. Let t be a transition such that g is one of its output goods, and $father(g)$ is its single input good. Besides, we note as G' the set of output goods of t. Then, we obtain $\pi_{g,j}$, the unitary price for good g as follows:

$$\pi_{g,j} = \frac{\pi_{father(g),k} \cdot |M[father(g),t]| + c(t) \cdot \nu}{M[g,t]} \omega_g \qquad (3)$$

where $\pi_{father(g),k}$ is the unitary price for good $father(g)$ for a provider $P_k \neq P_j$; $|M[father(g),t]|$ indicates the units of good $father(g)$ that are input to transition t; ν is a value obtained from a normal distribution $N(\mu_{production\_cost}, \sigma_{production\_cost})$ that weighs transformation cost $c(t)$; $M[g,t]$ indicates the number of units of good g that are output by transition t; and ω_g is the share factor for good g. Notice that after applying our pricing algorithm we obtain Π, an $n \times m$ matrix storing all unitary prices.

Several remarks apply to equation 3. Firstly, the share factors for output goods must satisfy $\sum_{g' \in G'} \omega_{g'} = 1$. Secondly, it may surprise the reader to realise that the value to propagate down the TNS ($\pi_{father(g),k}$) is collected from a different provider. We enforce this *crossover* operation among unitary prices of different providers to avoid undesirable *cascading effects* that occur when we start out calculating unitary prices departing from either high or low unitary root prices. In this way we avoid to produce non-competitive and extremely competitive bids respectively that could be in some sense regarded as noise that could eventually lead to diverting results. Finally, from equation 3 we can readily obtain the bid price for a bid $b_j \in B$ as $p_j = \sum_{i=1}^{n} a_j^i \cdot \pi_{i,j}$.

After generating a complete data set, in a MUCRA scenario the Winner Determination Algorithm (WDA) shall solely focus on finding an optimal allocation for the required goods, whereas in a MUCRAtR scenario, the WDA shall assess whether an optimal allocation that considers the buying agent's *t-relationships* can be obtained. Therefore, the difference is that a MUCRAtR WDA does consider and exploit both the buying agent's *t-relationships* along with the implicit transformation cost within each bid, while a MUCRA WDA does not.

To summarise, the parameters we must set to create an auction data set are reported in table 1.

2.2 Experimental Settings and Results

In order to measure the benefits provided by the introduction of t-relationships among goods we compute the cost of the optimal outcome, that is, the cost of the winning bid set for MUCRA (C^{MUCRA}) and the cost of the winning bid set plus transformations for MUCRAtR ($C^{MUCRAtR}$). We define the *saving index* (SI) as: $SI = 100 \cdot \frac{C^{MUCRA} - C^{MUCRAtR}}{C^{MUCRA}}$. The larger the index, the higher the benefits that a buyer can expect to obtain by using a MUCRAtR instead of a MUCRA. Our experimental hypothesis is that the SI index will increase as the buyer's transformation costs increase wrt the providers' ones. Therefore, we will analyse how the difference in production costs between the buyer and the providers affects SI. The differences among the transformation costs of the buyer and the average transformation costs of the providers are controlled by the $\mu_{production\_cost}$ parameter. We expect that, as the average transformation costs of the providers increases with respect to the buyer's ones, so do the benefits of using a MUCRAtR instead of a MUCRA. In fact, the experimental results do strongly agree with our hypothesis. Figure 2 depicts the results when varying $\mu_{production\_cost}$ from 0.8 to 1.8. The legend reports the value of the $p_{offered\_goods}(=)p_{offered\_units}$ parameter. As $\mu_{production\_cost}$ increases, so do savings.

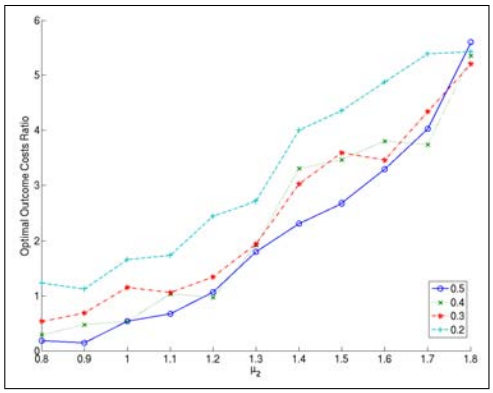


Figure 2: Varying the $\mu_{production\_cost}$ parameter

3. CONCLUSIONS AND FUTURE WORK

We have performed an experiment to empirically evaluate under which market conditions it is convenient to employ the new auction defined in [1]. We found that the best benefits in terms of savings are obtained when: (1) the manufacturing costs of a buyer/auctioneer are higher than the providers' ones; and (2) a buyer cannot access the raw materials at a price lower or equal than the providers.

Acknowledgements. This work was partially funded by the Spanish Science and Technology Ministry as part of the Web-i-2 project (TIC-2003-08763-C02-00). Giovannucci Andrea enjoys the BEC.09.01.04/05-164 CSIC scholarship.

4. REFERENCES

[1] A. Giovannucci, J. A. Rodrguez-Aguilar, and J. Cerquides. Multi-unit combinatorial reverse auctions with transformability relationships among goods. In *WINE 2005*, volume 3828 of *LNCS*, Hong Kong, China, December 2005. www.iiia.csic.es/∼andrea/papers/WINE318a.pdf.

[2] T. Murata. Petri nets: Properties, analysis and applications. In *IEEE*, volume 77, pages 541–580, 1989.

[3] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 69–76, Bologna, Italy, July 2002.

Ontologies and web services

ANEMONE: An Effective Minimal Ontology Negotiation Environment

Jurriaan van Diggelen, Robbert-Jan Beun, Frank Dignum, Rogier M. van Eijk, John-Jules Meyer
Institute of Information and Computing Sciences
Utrecht University, the Netherlands
{jurriaan, rj, dignum, rogier, jj}@cs.uu.nl

ABSTRACT

Communication in open heterogeneous multi agent systems is hampered by lack of shared ontologies. To overcome these problems, we propose a layered communication protocol which incorporates techniques for ontology exchange. Using this protocol, the agents gradually build towards a semantically integrated system by establishing minimal and effective shared ontologies. We tested our approach, called ANEMONE, on a number of heterogeneous news agents. We show how these agents successfully exchange information on news articles, despite initial difficulties raised by heterogeneous ontologies.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems, coherence and coordination*; D.2.12 [**Software Engineering**]: Interoperability—*data mapping*

General Terms

Languages, Theory

Keywords

Ontology negotiation, Semantic interoperability, Ontology alignment

1. INTRODUCTION

A fundamental communication problem in open multi agent systems is caused by the heterogeneity of the agent's knowledge sources, or more specifically of the underlying *ontologies*. Although ontologies are often advocated as a complete solution for knowledge sharing between agents, this is only true when all agents have knowledge about each others' ontology. The most straightforward way to establish this would be to develop one common ontology which is used by all agents. However, this scenario would be very unlikely in open multi agent systems, as those on the internet, because it would require all involved system developers to reach consensus on which ontology to use. Moreover, a common ontology forces an agent to abandon its own world view and adopt one that is not specifically designed for its task [7]. This may result in a suboptimal situation.

In an open multi agent system, communication problems that arise from heterogeneous ontologies should be *solved*, rather than avoided. In this paper, we present a suitable solution, called ANEMONE: AN Effective Minimal Ontology Negotiation Environment. We illustrate our approach with a case study that involves some semantic integration problems that are typical for open systems. We consider an open community of agents that periodically download news articles on different topics from RSS news feeds[1]. Different news providers are represented by different agents; the taxonomy of news topics supplied by a provider forms the agent's ontology. The fact that different news providers categorize their news differently gives rise to a proliferation of heterogeneous ontologies. The resulting semantic integration problems crop up once the agents start to exchange news articles with each other. For example, consider the agent Ag-M that represents the news provider *Moreover* [5], and the agent Ag-Y that represents the news provider *Yahoo* [3]. Driven by a user's request to Ag-M for more articles on the topic *Basketball*, Ag-M decides to ask Ag-Y for *Basketball* articles. Although Ag-Y has relevant articles on this topic, in Ag-Y's ontology they are classified under *NBA*[2]. Because Ag-Y does not know *Basketball* and Ag-M does not know *NBA* their communication fails.

Traditional approaches, such as standardization and *Ontology alignment* [16], are not suited for these kinds of semantic integration problems. Standardization efforts in this domain are almost doomed to fail, because the news providers deliberately distinguish themselves from others by using different ontologies. Ontology alignment has been proposed as a technique that enables agents to keep their individual ontologies by making use of *mappings* between the different ontologies. Although this is a step in the right direction, it assumes that the mappings can be pre-defined before the agents start interacting. In our case, it is not known beforehand which ontology mappings are needed due to the openness of the system and the fact that ontologies change from time to time. For example, one of Yahoo's news topics, *Asia Tsunami Disaster*, is clearly a temporary topic.

[1]RSS is a popular XML format for the syndication of news content on the internet
[2]"NBA" is the National Basketball Association

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

In ANEMONE, the agents exchange ontological information on an as-need basis. We hereby adhere to the emerging paradigm of ontology negotiation [6]. Agents first try to cope with the situation as is; when communication fails to be effective, the agents seek a minimal solution which solves their communication problem at hand. Drawing on related work on ontology negotiation and on our experiences with the news agent case, we adopt a standpoint on issues as what kind of shared ontology should be built up, and when and where this should occur. These standpoints are incorporated in ANEMONE and make it a novel approach for ontology negotiation. By implementing ANEMONE and applying it to the news agent case, we show that ANEMONE provides a solution for some semantic integration problems which, to the best of our knowledge, cannot be solved using existing techniques.

In the following section we compare our approach with related work. The third section provides a broad outline of our approach. The communication mechanisms of ANEMONE are discussed in section four. Section five describes implementation details and experimental results. We conclude and give directions for future research in section six.

2. RELATED WORK

Recently, researchers have turned their attention to more flexible solutions for semantic integration problems. An ontology agent [1] has been proposed to facilitate agent communication by registering ontologies and performing services such as translating between ontologies. This would allow agents to reconcile heterogeneous ontologies at interaction time, *on the condition* that every agent knows and trusts the ontology agent and that the ontology agent knows a mapping between every ontology in the system. In our case, such a centralized service is unsuitable as it is not known beforehand which inter-ontology mappings are needed.

Most recently, a few approaches appeared in the literature which tackled the problem in a fully *decentralized* way. W. Truszkowski and S. Bailin have coined the term *Ontology Negotiation* to refer to such approaches [6]. In their paper, the authors present a communication mechanism which enables agents to exchange parts of their ontology in a pattern of successive clarifications. The DOGGIE approach [20] makes similar assumptions, but focuses mainly on the machine learning aspects of ontology exchange, namely the problem of teaching the meaning of a concept to another agent. In this context, we also mention the approach by Soh and Chen [17], where agents exchange ontological knowledge when they believe it would improve operational efficiency. Ontology negotiation can be considered as the most ambitious approach for semantic integration [19]. Because this area is also relatively new, much work remains to be done.

The main differences between ANEMONE and the approaches mentioned above are:

- Contrary to [6], [20] and [17],ANEMONE is not restricted to equivalence mappings (mappings between two equivalent concepts in two ontologies), but deals with arbitrary mappings, such as subset mappings and disjointness mappings. For example, a mapping might state that one agent's concept *NBA* is a subset of another agent's concept *Basketball*. This has some repercussions in the overall communication mechanism regarding the prevention of overgeneralization.

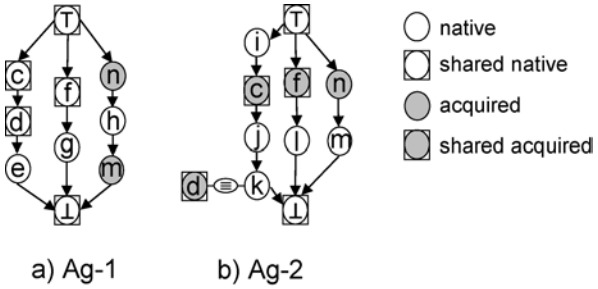


Figure 1: Example ontologies

- Before the agents use resource consuming machine learning techniques, they try to exchange a meaning of a concept by stating the relations with other concepts. This is done by defining the unknown concept in terms of other concepts. The receiver decides whether this definition sufficiently explains the concept.
- Whereas the approach by Soh and Chen uses improved operational efficiency as a main criterium to start ontology exchange, we use a formal, and thus more precise, notion of lossless communication [10]. As lossless communication depends on the ontologies of both agents, both agents are involved in assessing this property.

3. APPROACH

3.1 Ontologies

An ontology consists of a set of concepts and a set of concept relations. We avoid naming conflicts by ensuring that two concepts in different ontologies with the same name also have the same meaning. This can be easily achieved by using namespaces. A concept relation is one of the following: $\sqsubset$ (strict subconcept relation), $\sqsupset$ (strict superconcept relation), $\equiv$ (equivalence), $\perp$ (disjointness), $\oplus$ (overlap). The symbol $\perp$ is also used to denote the bottom concept. Figure 1 presents a graphical representation of two example ontologies. An arrow between two concepts represents a subconcept relation (and against the flow a superconcept relation). Two concepts in two different branches in the ontology are disjoint. Concepts that are equivalent or overlap are connected with a line with the $\equiv$ or $\oplus$ symbol in it. For readability, we have left out concept relations that are derivable from other relations.

Besides concepts and their relations, some other aspects of ontologies are important in the context of ontology negotiation. Firstly, there is a difference between a *native* concept and an *acquired* concept. In Figure 1, the former is represented as non-shaded and the latter as shaded. An agent that has not yet exchanged any ontological knowledge with other agents, has only native concepts in its ontology. The native ontology of an agent is implemented by its system developer to enable it to store and reason with operational knowledge (knowledge that is used by the agent to carry out its task). Because the operational knowledge base is defined in terms of the native ontology, this ontology is constrained to be static. During ontology negotiation, the agents occasionally teach each other new concepts. These are the agent's acquired concepts and are defined additionally to its native concepts. Acquired concepts are not used for storing knowledge but only serve as vehicles for communication.

This allows the acquired ontology to be dynamic.

Another important issue in ontology negotiation is keeping track of which concepts are shared with other agents. In Figure 1, a concept that is shared between Ag-1 and Ag-2 is represented by a box around the concept. When we speak of a concept that is shared between two agents, we actually mean that the concept is common knowledge between the agents. In ontology negotiation, a concept may also be *unknowingly shared*. In this case, both agents know the concept, but do not know this of each other. Examples of unknowingly shared concepts are n and m in Figure 1. Such a situation might arise when two agents learn the same concept from a third agent.

The following terminology is useful for discussing the communication mechanisms of ANEMONE:

- γ_1 is a *particularization* w.r.t. γ_2 iff $\gamma_1 \sqsubset \gamma_2$ or $\gamma_1 \oplus \gamma_2$
- γ_1 is an *implied concept* w.r.t. γ_2 iff $\gamma_1 \sqsupset \gamma_2$ or $\gamma_1 \equiv \gamma_2$

3.2 Communication

ANEMONE does not make any initial demands on agents with regard to their amount of shared concepts. We will discuss an initial situation which is the most difficult, i.e. the agents do not share any native concepts besides the $\top$ concept. At first, these agents lack any means to cooperate. As they participate in conversations, they build up something which enables them to reach a desired level of coordination. An overview of the communication protocol that provides for this is presented in Figure 2. The upper layer of the protocol (NCP) deals with normal agent communication, i.e. the kind of social interaction which agents normally exhibit when no ontology problems exist in the system. To deal with ontology problems, two layers are added to the protocol: a protocol for exchanging concept definitions (CDP), and a protocol for teaching concepts to each other using machine learning techniques (CEP).

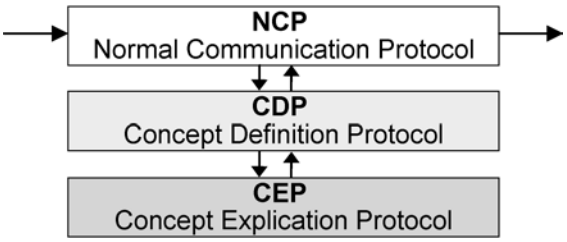


Figure 2: Layered communication protocol

This modular structure of ANEMONE's communication mechanism enables us to make alterations in one layer of the protocol without affecting the other layers. For example, in our news agent domain we have implemented CEP using text classification techniques. In another domain, which deals with individuals other than text documents, other concept learning techniques may be used in CEP.

Our focus in this section is on the domain independent design choices of the protocol, for example, on how the agents traverse through the protocol layers and on general techniques that can be used in the individual layers. These design choices are motivated by a number of standpoints on issues as: *what* should be built up during ontology negotiation and *where* and *when* is this built up?

- What: *Minimal & Effective* Agents should build up a minimal and effective shared ontology. An ontology that is minimal in size is not bulky and slowly processable due to superfluous concepts. An ontology that is minimal in use enables agents to keep their conversations short. An effective ontology is expressive enough to convey sufficient information in a sound manner. Particularly, communication should be *lossless*, meaning that no information is lost in communication that could have been avoided by spending more effort on learning each others' ontologies. A formal definition of lossless communication is presented in [10].

- When: *Lazy* Agents should build up a shared ontology only when strictly necessary. Ontology exchange is not a goal in itself; it is a means of enabling communication. The agents leave NCP when communication fails to be effective and try to return as soon as possible. Defining a concept in CDP is less resource consuming than learning a concept in CEP. Therefore an agent explicates instead of defines a concept only when strictly necessary. In other words, the agents try to stay as high up as possible in the layered communication protocol.

- Where: *Decentralized* In line with the ontology negotiation paradigm, no central location exists where a shared ontology is built up. Every agent increments its own ontology with the necessary concepts. In this way, they collectively address the semantic integration problem.

4. COMMUNICATION MECHANISM

In the following three sections we discuss and motivate the communication mechanisms that underly NCP, CDP and CEP. An overview of the messages that are exchanged while following the protocol is presented in figure 6.

4.1 Normal Communication Protocol

Normal communication protocols have been extensively studied in the agent communication literature. The focus of this section is on the adjustments that are required to deal with the partially shared ontologies described in section 3.1, and the criteria upon which the agents base their decision to resort to CDP. This confronts us with the task of finding a proper way to deal with *message composition*, *message interpretation*, and the decision of when to *switch to CDP*. To discuss these issues, we assume a simple setting in which a speaker intends to convey one of its native concepts to a hearer. *Message composition* is the speaker's task of translating the native concept it intends to convey to a concept it will actually speak to the hearer. *Message interpretation* is the hearer's task of translating the concept in the message to one of its native concepts to store the information in its knowledge base. Either the speaker or the hearer may decide to *switch to CDP* when it believes that more concepts should be shared between them. We begin our discussion with a very simple way to implement message composition, interpretation, and a switch to CDP. As we evaluate this protocol against the quality criteria of minimality, effectivity, and laziness (discussed in section 3.2), we detect shortcomings and adjust the protocol accordingly. Subsequent evaluations and refinements eventually give rise

to a communication mechanism of considerable complexity which is described in section 4.1.5.

4.1.1 Equivalence mappings

Our first proposal for NCP is a simple communication mechanism that exploits some features of partially shared ontologies introduced in section 3.1, namely the distinction between native and acquired concepts, equivalence mappings, and the notion of shared concepts. These features can be incorporated in a relatively straightforward manner. In this proposal, message composition involves translating the native concept the speaker intends to convey to an *equivalent shared concept*, and to use this in a message. Message interpretation involves translating the concept used in the message to an *equivalent native concept*. The speaker switches to CDP when no equivalent shared concept is available in its ontology. Some examples w.r.t. Figure 1: Ag-1 intends to convey d, speaks d which Ag-2 interprets as k; Ag-1 intends to convey e, can not do so, and switches to CEP to teach the concept to Ag-2. Note that in this scenario, a shared concept enables communication in *both* directions. For example, when Ag-2 intends to convey k, it may speak d which Ag-1 interprets as d. The speech act that allows the speaker to translate what it intends to convey to an equivalent shared concept is called *ExactInform*. It is clear that this communication mechanism does not give rise to information loss, i.e. it enables lossless communication.

4.1.2 Beyond equivalence mappings

The previous proposal for NCP does not describe the handling of acquired concepts that are mapped in an agent's native ontology as sub- and superconcepts. For example, Ag-2's acquired concept c is mapped between its native concepts j and i. One measure we must take to overcome this shortcoming, is to make message interpretation less strict; from now on, message interpretation involves translating the concept used in the message to *the most specific implied native concept* (instead of to an equivalent native concept). For example, Ag-1 intends to convey c, exactinforms c which Ag-2 interprets as i. Note that although this version of ExactInform may introduce some information loss, no information is lost in the communication process that could have been prevented by sharing extra concepts. Therefore, ExactInform still guarantees lossless communication.

Besides this refinement to message interpretation, the message composition part in the protocol also requires revision. This is motivated in the following example. If message composition is defined as strictly as in the previous proposal, Ag-2 would never use concept c in its message because none of Ag-2's native concepts is equivalent to c. To comply with minimality, shared concepts should allow for communication in both directions. For example, Ag-2 should be allowed to communicate j using the shared concept c. For this reason, we loosen the conditions for message composition such that the speaker can translate the native concept it intends to convey to a shared implied concept. The speech act that is associated with this version of message composition is called *Inform*. For example: Ag-2 intends to convey j, informs c which Ag-1 interprets as c; Ag-1 intends to convey e, informs d, which Ag-2 interprets as k.

4.1.3 Towards effectivity

The previous proposal for NCP threatens the effectiveness of communication by allowing the agents to become overly general. In the extreme case, an agent that intends to convey one of its native concepts, speaks the shared concept $\top$. Apparently, inform does not guarantee lossless communication. We prevent overgeneralization by enabling the hearer to recognize when communication has been lossless and when not. When the hearer is not certain that communication was lossless, it requests specification. When the speaker cannot convey more specific information due to the lack of shared concepts, the agent switches to CDP. Stated in this way, the decision to switch to CDP has to do with recognizing overgeneralization.

The hearer recognizes overgeneralization by reasoning as follows. Upon hearing that an individual belongs to some concept, it knows that the individual is a member of every implied concept and that the individual is not a member of all concepts that are disjoint with the concept. This knowledge cannot be a symptom of overgeneralization. However, the hearer remains ignorant about the particularizations of the concept. This ignorance may be a symptom of overgeneralization. The rule for recognizing overgeneralization we propose here is: if there are native particularizations of the concept used in the message, communication may not have been lossless. For example: Ag-2 intends to convey k, informs c, Ag-1 requests specification (because of concept d), Ag-2 exactinforms d and the conversation finishes; Ag-1 intends to convey e, informs d, Ag-2 interprets this as k, recognizes lossless communication and the conversation finishes; Ag-2 intends to convey l, informs f, after which Ag-1 requests specification (because of g), after which Ag-2 switches to CEP to teach l to Ag-1.

4.1.4 Towards efficiency

The proposal for NCP described in the previous section eventually guarantees lossless communication. However, the communication mechanism is not minimal in use. For example, every time that Ag-2 informs Ag-1 about j, Ag-1 requests specification (due to the presence of d), upon which Ag-2 answers that it does not know whether d holds. Especially when larger ontologies are involved, every time the speaker intends to convey a general concept, a cumbersome dialogue follows where the hearer requests for specification many times. To make communication more efficient, we make the following adjustments to the *Inform* speech act. In message composition, the agent is obliged to translate to the *most specific* shared implied concept. This not only prevents the speaker from becoming more general than necessary, it also enables the hearer to form a belief about what the speaker intended to convey and what it did not. In philosophy of language, such a derivation is known as a *conversational implicature* [12]. In ANEMONE, it works as follows: upon receiving a concept, the hearer knows that the speaker did *not* intend to convey the shared subconcepts of that concept, otherwise it would have spoken differently. It therefore considers it useless to request these particularizations. For example, when Ag-2 speaks c (with the intention to convey j), Ag-1 does not request whether d holds, because it knows that Ag-2 could not have intended to convey d. But the hearer can form an even stronger belief than that. The hearer also knows that the speaker did *not* intend to convey any non-shared subconcepts of the shared subconcept of the spoken concept. For example, when Ag-2 informs

c (with the intention to convey j), Ag-1 does not request whether e holds, because it knows that Ag-2 could not have intended to convey e, otherwise it would have informed d. This communication mechanism has been proven to yield lossless communication [8].

4.1.5 Dealing with unknown concepts

Until now, we have discussed how agents should deal with shared concepts that are common knowledge. As argued before, ontology negotiation also gives rise to unknowingly shared concepts. A minimal ontology negotiation protocol should exploit this property. Therefore, in message composition, we also allow an agent to translate to a non-shared concept (which may turn out to be an unknowingly shared concept). The rule is that an agent may still translate what it intends to convey to the most specific shared implied concept, but if there are more specific non-shared implied concepts present, it may also choose one of these. Once an agent has used an unknowingly shared concept in the message, this concept becomes shared. Because an agent is not allowed to speak non-shared implied concepts that are more general than a shared implied concept, the efficiency measures that make use of conversational implicatures are still applicable. The choice to translate to a shared concept with the risk of being too general, or to translate to a more specific non-shared concept, with the risk of being not understood, is left to the agent. Communication strategies that resolve non-determinacy in the communication protocol are investigated in [9].

The final version of NCP is summarized in Figure 3. The next example shows that it may be beneficial to speak an unknowingly shared concept rather than a more general shared concept. Ag-2 intends to convey m, exactinforms the non-shared concept m which is understood by Ag-1 as h. Ag-2 could also have informed the shared concept $\top$ with the intention to convey m, which would have been answered by Ag-1 with a request for specificity. The next example shows that it may be beneficial to speak a knowingly shared concept, rather than a more specific non-shared concept. Suppose that Ag-1 intends to convey e, exactinforms e which does not get understood by Ag-2. If Ag-1 would have informed d instead, Ag-2 would have understood the message and would have recognized lossless communication.

Composition
- The agent translates what it intends to convey to:
  - the most specific shared implied concept, or
  - an implied concept that is more specific than the most specific shared implied concept

Interpretation
- The agent translates the concept in the message to:
  - the most specific implied native concept

Switch to CDP
- The hearer switches to CDP when the speaker cannot convey what it intends to convey in a message that gets understood by the hearer and which is not overgeneralized. The hearer considers an inform message which mentions a concept γ overgeneralized if native particularizations of γ exist that are not
  - a shared subconcept of γ, or
  - a subconcept of a shared subconcept of γ

Figure 3: Normal Communication Protocol

4.2 Concept Definition Protocol

In the concept definition protocol, the speaker tries to convey the meaning of a concept by stating the relations with other concepts, i.e. it speaks a number of concept definitions. If these definitions enable the hearer to derive the *complete meaning* of the concept, the hearer switches back to NCP. An agent considers the meaning of an acquired concept complete, if it knows the relation with every other concept in its ontology. If there are not sufficient shared concepts available to convey the complete meaning, the agents switch to CEP. For example, Ag-2 intends to define k, speaks "$k \equiv d$" after which Ag-1 knows the complete meaning of k and switches back to NCP. Suppose Ag-1 intends to define h and speaks "$m \sqsubset h \sqsubset n$". This definition enables Ag-2 to derive that h is disjoint with f, l, i, c, j, k, d, that h is subset of $\top$ and n, and that h is a superset $\bot$ and m. It therefore regards the meaning of h complete and switches back to NCP. Suppose Ag-1 intends to convey the meaning of g and speaks "$\bot \sqsubset g \sqsubset f$". This definition leaves Ag-2 ignorant about the relation between g and l. It does not consider this meaning complete, and switches to CEP.

Composition
- The agent states the relations of what it intends to define with every shared concept.

Interpretation
- The agent adds the defined concept as an acquired concept to its ontology.
- The agent adds the relations in the message to its ontology, and derives relations that follow from this.

Switch to NCP
- When the hearer knows the relation of the defined concept with every concept in its ontology.

Switch to CEP
- When the speaker does not know any shared concepts to define concept relations.
- When the hearer does not know the relation of the defined concept with every concept in its ontology.

Figure 4: Concept Definition Protocol

4.3 Concept Explication Protocol

The purpose of CEP is to convey the meaning of a concept when no satisfactory definition of the concept in terms of other concepts can be given. A commonly used technique for generating a concept mapping is to invoke a dictionary or a thesaurus, such as WordNet [11]. However, this turns out to have its limitations in our domain. Firstly, the technique is incomplete, e.g. Yahoo's concept *NBA* is not defined in WordNet. Secondly, the technique may be incorrect. For example, the *International* concept of the American company Reuters is really a superset of the *UK* concept of the (British) BBC. From the perspective of the BBC agent, WordNet would suggest differently. For these reasons, we do not take the name of a concept into account while finding out its meaning.

Following Luc Steels [18], we assume that the meaning of a concept can be conveyed to another agent by pointing to instances. A basic requirement for this approach is a *classi-*

fier for each concept in an agent's ontology. The speaking agent, upon explicating a concept, communicates a number of positive and a number of negative examples of the concept. The hearer classifies these examples using the concept classifiers from its own ontology. We use the following terms that are common in machine learning [15]:

- TPR: True positive rate: the number of positively classified positive examples divided by the total number of positive examples.
- TNR: True negative rate: the number of negatively classified negative examples divided by the total number of negative examples.

For each concept, the agent calculates TPR and TNR and uses this information to assess the concept relation. In the ideal situation, when the agent's classifiers are perfect and every element in the domain of discourse is used as an example, TPR and TNR offer strict criteria for determining the correct concept relation. Consider an ideal situation, where TPR and TNR result from applying a classifier of concept γ_1 to examples of concept γ_2. The relation between concept γ_1 and γ_2 can be determined as follows: $\gamma_1 \equiv \gamma_2$ iff $\text{TPR} = 1 \wedge \text{TNR} = 1$; $\gamma_1 \bot \gamma_2$ iff $\text{TPR} = 0$; $\gamma_1 \sqsubset \gamma_2$ iff $0 < \text{TPR} < 1 \wedge \text{TNR} = 1$; $\gamma_1 \sqsupset \gamma_2$ iff $\text{TPR} = 1 \wedge \text{TNR} < 1$; $\gamma_1 \oplus \gamma_2$ iff $0 < \text{TPR} < 1 \wedge \text{TNR} < 1$. In most practical situations, the agents have noisy classifiers and the domain of discourse may be infinite (as in our case study). We deal with this by loosening the conditions for determining concept relations. There are actually many ways to do this. We follow a simple approach by introducing some threshold parameters θ_1, θ_2 and θ_3. The way the concept relation is assessed is described in Figure 5.

Composition

- The agent sends positive and negative examples. The positive examples are individuals belonging to the concept to be explicated. The negative examples are individuals that belong to concepts that are disjoint with the concept to be explicated.

Interpretation

- For every native concept γ_1 in its ontology, the agent computes TPR and TNR using the examples in the message.
- The agent assesses the relation of the explicated concept γ_2 with every native concept γ_1 as follows:
  - $\gamma_1 \equiv \gamma_2$ if $\text{TPR} \geq \theta_1 \wedge \text{TNR} \geq \theta_2$
  - $\gamma_1 \bot \gamma_2$ if $\text{TPR} \leq \theta_3$
  - $\gamma_1 \sqsubset \gamma_2$ if $\text{TPR} < \theta_1 \wedge \text{TNR} \geq \theta_2$
  - $\gamma_1 \sqsupset \gamma_2$ if $\text{TPR} \geq \theta_1 \wedge \text{TNR} < \theta_2$
  - $\gamma_1 \oplus \gamma_2$ if none of the previous conditions holds

Switch to CDP

- After the hearer has processed the examples.

Figure 5: Concept Explication Protocol

5. RESULTS

5.1 Implementation

We have implemented ANEMONE as a multi-threaded Delphi application. A single news agent corresponds to an RSS reader. It periodically checks the internet for new news articles that belong to one of the concepts in its ontology. If so, it downloads these articles and uses an XML-parser to extract the title, summary, date, and URL of the article. It stores this information in a local database. The agents actively exchange information with each other about new news articles and their topics. NCP provides for this communication. A query for news articles of a topic, is interpreted as a request to the other agent for an inform speech act, after which NCP deals with the subsequent communication. The non-determinacy in message-composition is resolved using the following simple communication strategy. If the speaker intends to convey γ and the only translation of γ to a shared concept is $\top$, the speaker prefers to speak the unshared concept γ; in all other cases, the speaker translates to the most specific shared implied concept of γ.

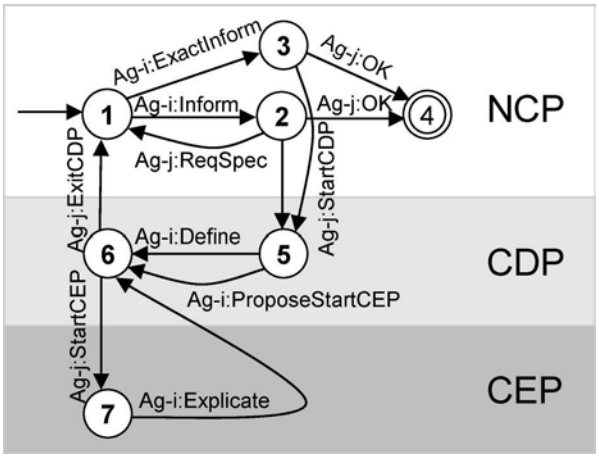


Figure 6: Message protocol of ANEMONE

The concept explication protocol incorporates some application specific design choices. Because our case deals with text documents, the classifiers are implemented using text classification techniques. We use a support vector machine [14] to classify the *description* fields of the news articles of a topic. The texts are preprocessed using common techniques in natural language processing [13], such as *stemming* and *stopword removal*; subsequently, the texts are transformed in term frequency vectors. These vectors are used for training the agent's concept classifiers. When the agent has sufficiently trained its concept classifiers on its own news articles, it is capable of learning concepts from other agents by following CEP. In CEP, the teacher sends news articles belonging to the concept to be explicated as positive examples, and news articles belonging to the concepts that are disjoint with the concept to be explicated as negative examples. As described in the previous section, the hearing agent computes TPR and TNR values for each concept in its ontology and derives the concept relations. Because in our application, a small amount of misclassifications is acceptable, we used relatively tolerant criteria to assess concept relations, i.e. $\theta_1 = 0.75$, $\theta_2 = 0.75$, $\theta_3 = 0.4$.

5.2 Case study

We demonstrate the system using the four relatively simple agents depicted in Figure 7. The agents represent news publishers BBC [2], Moreover [5], Reuters [4] and Yahoo [3]. The agent's ontologies consist of subsets of the news feeds provided by the news publishers. After the agents have col-

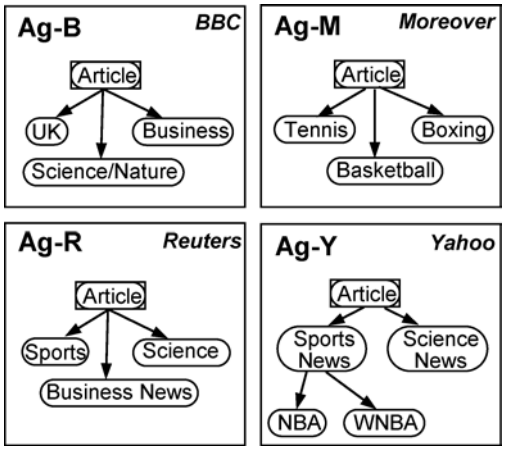


Figure 7: Initial situation

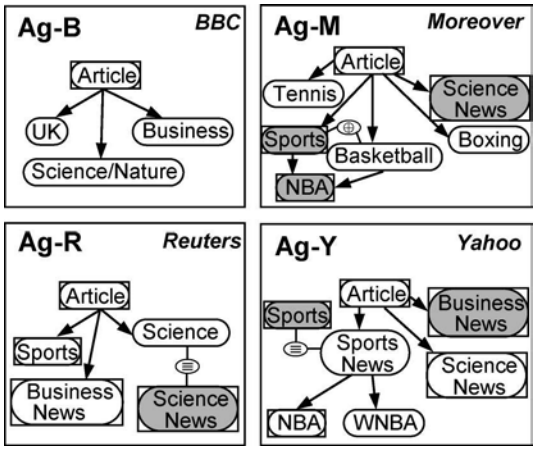


Figure 8: Situation after 9 conversations

lected news articles for a period of two months, their knowledge bases were filled with approximately 200 news articles per topic. This enabled them to train their classifiers and participate in ontology negotiation. The following examples present a series of successive dialogues which were generated by the agents. For demonstration purposes, the agents make some of their internal workings public; these are preceded by a *. Due to space limitations, we abbreviated long path names of news articles with `[]`, and summarized some of the conversations. The agents begin in a situation where they do not share any concepts with each other (Figure 7).

Ag-Y intends to inform Ag-R about Science News.

```
1 Ag-Y:<Ag-R,<ExactInform,Science News,[]>>
2 Ag-R:<Ag-Y,<StartCDP>>
3 Ag-Y:<Ag-R,<ProposeStartCEP>>
4 Ag-R:<Ag-Y,<StartCEP>>
5 Ag-Y:<Ag-R,<Explicate,Science News,[],[]>>
* Ag-R Business News: TPR=0.15 TNR=0.86
* Ag-R Science: TPR=0.76 TNR=1
* Ag-R Sports: TPR=0 TNR=0.24
* Ag-R equivalent(Science News,Science)
6 Ag-R:<Ag-Y,<ExitCDP>>
7 Ag-Y:<Ag-R,<ExactInform,Science News,[]>>
* Ag-R translates to Science
```

Ag-R intends to inform Ag-Y about Science

```
8 Ag-R:<Ag-Y,<ExactInform,Science News,[]>>
```

Ag-Y intends to inform Ag-M about Science News

```
9 Ag-Y:<Ag-M,<ExactInform,Science News,[]>>
10 Ag-M:<Ag-Y,<StartCDP>>
11 Ag-Y:<Ag-M,<ProposeStartCEP>>
12 Ag-M:<Ag-Y,<StartCEP>>
13 Ag-Y:<Ag-M,<Explicate,Science News,[],[]>>
* Ag-M Basketball: TPR=0.29 TNR=0.32
* Ag-M Tennis: TPR=0.21 TNR=0.84
* Ag-M Boxing: TPR=0.26 TNR=1
* Ag-M disjoint(Science News,Basketball;Tennis;Boxing)
14 Ag-M:<Ag-Y,<ExitCDP>>
15 Ag-Y:<Ag-M,<ExactInform,Science News,[]>>
* Ag-M translates to Article
```

Ag-R intends to inform Ag-M about Science

```
16 Ag-R:<Ag-M,<ExactInform,Science News,[]>>
* Ag-R translates to article
```

Ag-R intends to inform Ag-Y about Sports

```
17 Ag-R:<Ag-Y,<ExactInform,Sports,[]>>
18 Ag-Y:<Ag-R,<StartCDP>>
19 Ag-R:<Ag-Y,<Define Sports,
       <disjoint Science News>>>
20 Ag-Y:<Ag-R,<StartCEP>>
```

21-27 *In CEP, Ag-Y learns that Sports is equivalent with Sports News*

Ag-Y intends to inform Ag-R about NBA

```
28 Ag-Y:<Ag-R,<Inform,Sports,[]>>
29 Ag-R:<Ag-Y,<OK>>
```

Ag-R intends to inform Ag-Y about Business News

```
30 Ag-R:<Ag-Y,<ExactInform,Business News,[]>>
31 Ag-Y:<Ag-R,<StartCDP>>
32 Ag-R:<Ag-Y,<Define,Business News,
   <disjoint Sports,disjoint Science News>>>
33 Ag-Y:<Ag-R,<ExitCDP>>
34 Ag-R:<Ag-Y,<ExactInform,Business News,[]>>
* Ag-Y translates to Article
```

Ag-Y intends to inform Ag-M about Sports News

35-41 *In CEP, Ag-M derives that Sports overlaps with basketball. Ag-M translates Sports to Article*

Ag-Y intends to inform Ag-M about NBA

```
42 Ag-Y:<Ag-M,<Inform,Sports,[]>>
*  Ag-M translates to Article
43 Ag-M:<Ag-Y,<ReqSpec>>
44 Ag-Y:<Ag-M,<ExactInform,NBA>>
45 Ag-M:<Ag-Y,<StartCDP>>
```

46-51 *In CEP Ag-M learns that NBA is a subset of basketball and translates to basketball*

After this conversation, the agents have built up knowledge about each others' ontologies, as shown in Figure 7. The agents have built up a shared ontology that enabled them to convey what they intended to convey (effectivity), but have not made the shared ontology larger than required (minimality). The acquired concepts are not only mapped to equivalent native concepts but also to native superconcepts (e.g. Ag-M's *NBA* and *Basketball*) and to disjoint native concepts (e.g. Ag-M's *Science News*). Because the approaches mentioned in section 2 only deal with equivalence mappings, these approaches would have failed to solve

the semantic integration problems of the news agents.

The conversation shows that many of the ontology negotiation techniques we have discussed in this paper are useful in a realistic domain. We mention the following: the use of mappings other than equivalence mappings (9-15, 42-51), the use of generalization in message composition (28-29), detecting information loss (42-43), the use of speaking in *unknowingly shared* concepts (16), using acquired concepts for communication in *both* directions (8,28), the use of communicating concept definitions (30-34), detecting inadequate concept definitions (19-20).

In this case, it took around 10 seconds (on a Pentium 4, 2.5 GHz) for the agents to finish concept explication. Although the CEP layer worked well enough for this relatively simple case, in a scenario with more complex ontologies or where the correctness of concept mappings is of critical importance, the CEP layer needs improvements. Most likely, these improved versions will be even more time consuming. This strengthens our motivation for lazy ontology negotiation to reduce the occurrence of concept explication to a minimum.

6. CONCLUSION

In this paper, we have presented the ANEMONE approach for solving semantic integration problems. Instead of trying to solve all ontology problems at one stretch at design time, ANEMONE provides agents with the tools to overcome ontology problems at agent interaction time. The layered communication mechanism tackles semantic integration problems when needed and only when needed.

We have applied ANEMONE to the domain of news articles. Despite the fact that agents used different ontologies to classify news articles, they were able to overcome their communication problems and successfully exchanged news articles with each other. The performance of ANEMONE can be improved by incorporating more sophisticated machine learning techniques in the lower layer of the protocol. For our application, where the agents use relatively simple ontologies, the concept explication layer worked sufficiently well. The focus in this paper, however, has been on the communication mechanisms which serve to embed the machine learning techniques in the overall communication architecture, i.e. the upper two layers in the protocol. Even in our relatively simple case, these communication mechanisms proved to be useful. The agents refrain from applying resource consuming machine learning techniques as much as possible. When necessary, they gradually contribute to a shared ontology which is minimal in size and enables them to convey sufficient information. There is no need for a central coordinating agent; the agents find out by themselves if their communication is not satisfactory and solve the problem between themselves. These features of ANEMONE established that the news article agents, after having participated in only a few conversations, became part of a reasonably semantically integrated system.

In the future, we plan to apply ANEMONE to more complex domains. This may require us to replace the current concept explication protocol with one that is better suited for that domain. Furthermore, the ontologies in that domain may be more complex than the concept hierarchies we have focussed on in this paper. This requires an extension of the other layers in the communication protocol to deal with, for example, the alignment of concept attributes.

7. REFERENCES

[1] FIPA Ontology Service Specification. http://www.fipa.org/specs/fipa00086/.

[2] http://news.bbc.co.uk/1/hi/help/3223484.stm.

[3] http://news.yahoo.com/rss.

[4] http://today.reuters.com/rss/newsrss.aspx.

[5] http://w.moreover.com/site/other/categories_rss.html?

[6] S. Bailin and W. Truszkowski. Ontology negotiation between intelligent information agents. *Knowledge Engineering Review*, 17(1):7–19, 2002.

[7] T. Bylander and B. Chandrasekaran. Generic tasks for knowledge-based reasoning: the right level of abstraction for knowledge acquisition. *Int. J. Man-Mach. Stud.*, 26(2):231–243, 1987.

[8] J. v. Diggelen, R. Beun, F. Dignum, R. v. Eijk, and J.-J. Meyer. Combining normal communication with ontology alignment. *Proceedings of the International Workshop on Agent Communication (AC'05).*

[9] J. v. Diggelen, R. Beun, F. Dignum, R. v. Eijk, and J.-J. Meyer. A decentralized approach for establishing a shared communication vocabulary. *Proceedings of the International Workshop on Agent Mediated Knowledge Management (AMKM'05) held with AAMAS 2005.*

[10] J. v. Diggelen, R. Beun, F. Dignum, R. v. Eijk, and J.-J. Meyer. Optimal communication vocabularies and heterogeneous ontologies. In *Developments in Agent Communication*, LNAI 3396. Springer Verlag, 2004.

[11] C. Fellbaum. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.

[12] H. P. Grice. Logic and conversation. *Cole, P., and J.L. Morgan, eds. Speech Acts. New York: Academic Press, 4158*, 1975.

[13] P. Jackson and I. Moulinier. *Natural Language Processing for Online Applications: Text retrieval, extraction, and categorization.* John Benjamins Publishing, 2002.

[14] T. Joachims. Making large-scale svm learning practical. *Advances in Kernel Methods - Support Vector Learning*, 1999.

[15] R. Kohavi and F. Provost. *Machine Learning*, (30):271–274, 1998.

[16] N. F. Noy and M. A. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. *In Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2000.

[17] L.-K. Soh and C. Chen. Balancing ontological and operational factors in refining multiagent neigborhoods. *Proceedings of the Fourth International Conference on Autonomous Agents and Multi-Agent Systems*, 2005.

[18] L. Steels. Language as a complex adaptive system. In *Proceedings of PPSN VI*, Lecture Notes in Computer Science, Berlin, September 2000. Springer-Verlag.

[19] M. Uschold and M. Gruninger. Creating semantically integrated communities on the world wide web. *Semantic Web Workshop Co-located with WWW 2002 Honolulu*, 2002.

[20] A. Williams. Learning to share meaning in a multi-agent system. *Autonomous Agents and Multi-Agent Systems*, 8(2):165–193, 2004.

Agents for e-Business Applications

A.Negri, A. Poggi, M. Tomaiuolo, P. Turci
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma
Viale delle Scienze, 181A – 43100 – Parma
Tel. +39 0521 905708, Fax +39 0521 905723
{negri, poggi, tomamic, turci}@ce.unipr.it

ABSTRACT

Web services are increasingly utilized by organizations that want to improve responsiveness and efficiency. While they may be used in an isolated way, the need of integrating them as part of workflow processes is more and more felt. However the creation of applications composed of dynamically selected basic services entails facing two essential issues: how to efficiently discover Web services and how to allow and facilitate their composition.

In this paper, we propose an agent-based framework representing an attempt of giving an answer to such problems. Its peculiar characteristic and strength is the integration of the agent technology with other key emerging technologies, that is semantic Web, Web service, rule engine and workflow technologies. The multiagent system, which constitutes the backbone of the framework, represents the "glue" that holds these pieces together and makes them perform properly. The framework has been experimented and evaluated in the realization of a simple, but realistic, prototype of an e-travelling system. The results, though still preliminary, are quite encouraging.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *multiagent systems*.

General Terms

Management, Experimentation, Security.

Keywords

Agent-mediated e-business, ontology, Web services, workflow, security, service discovery and composition.

1. INTRODUCTION

The rapid growth of the Internet, networking systems and Web technologies is stimulating an ever-increasing number of companies to use internet not only to provide e-commerce worldwide, but also to improve internal communication, to help manage supply chains, to conduct technical and market research, and to locate potential partners.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

Agent technology is considered one of the most interesting technologies to successfully support these activities, which are gathered under the name of e-business. In fact, besides being an ideal mechanism for implementing complex systems, agent technology is well-suited to applications, like e-business ones, that are communication-centric, based on distributed computational and information systems, and requiring autonomous components readily adaptable to changes. In particular, the use of agent technology for realizing e-business systems provides conceptual simplicity, enhances scalability, and makes interactions in a large collection of information sources become tractable.

Considering their peculiar features and the role they should play in an e-business scenario, it is evident that agents do not necessarily offer new functionalities. To be successful, it is crucial to appropriately engineer and integrate agent technology with other technologies that have found and will find a purpose within enterprise computing. Data-mining, workflows, rule engines, just to mention a few, belong to the former group.

New interesting and promising technologies are represented by the semantic Web and Web services. The vision which is making its way into the research community is to encapsulate the organization's functionalities within appropriate interfaces and advertise them as one or more Web services, which could be integrated, when brought into play, in workflows. This innovative idea brings with it new outstanding opportunities but also new great issues, related mainly to the ability of automatically discovering and composing Web services. An answer to these problems could come from the semantic Web technology.

Recently, we have seen an explosion of interest in ontologies as artefacts to represent human knowledge and as a critical component in several applications; among these the e-business applications. Moreover the "marriage" between agents and ontologies seems to be the kind of technology that can significantly change the face of enterprise software.

On the one hand, ontologies should accomplish the task of giving a precious support to solve two tricky problems: how to efficiently discover Web services and how to make possible the interoperability of heterogeneous Web services. In order to facilitate the resolution of such a structural and semantic heterogeneity, Web services, which play the role of workflow components, will have their interfaces semantically described by ontological concepts.

On the other hand, ontologies enable agents to communicate in a semantic way, exchanging messages which convey information according to explicit domain ontologies.

In this scenario agents represent the "glue" that could hold these pieces together and make them perform properly.

Assuming to adopt an agent-based approach, a typical scenario of an e-business application would be characterized mainly by three actors: service providers, brokers and users, playing roles which would be allocated to different concrete agents. The system architecture would likely be organized in communities constituted by different kinds of agents: service providers, personal assistants and middle agents (e.g. service brokers, user profile managers, workflow managers, etc). In order to achieve their goals (semantic matching, service contracting and so on) these autonomous agents should be able to perform their tasks in cooperation or competition with other agents and to interoperate with external entities (e.g., legacy software systems). Moreover they should show reasoning capabilities and should have a support for dynamic behaviour modification based on business rules. Finally they should be able to build workflows, compose the external Web services and monitor their execution. The entire process should be supported by a distributed trust management.

Clearly the researchers are well aware that such a scenario is quite ambitious and the outlined objectives difficult to achieve in a short period. Indeed there are several overlooked technical issues and the present technology presents significant limitations. Nevertheless the realizations of prototype systems centred on the underlying infrastructure can be of great help in order to raise awareness of these issues and to delineate possible solutions.

Bearing in mind what said above, the aim of the present paper is to introduce a framework, under development at the University of Parma, for the realization of agent-based e-business applications.

In the next section we discuss the related work in the fields of the emergent and more established technologies which we aim at integrating with agent technology. Section 4 describes the framework aiming at being the basis for the realization of successful and innovative agent-based e-business applications. In this section we focus mainly on its architecture, the ontological support, the integration with a rule engine and our proposal for a distributed trust management. Section 5 shortly presents a prototype of an e-travelling system realized by using JADE and the aforementioned framework. Finally, we give some concluding remarks and present our future research directions.

2. RELATED WORK

There is evidence from several research studies [2],[33] that agents represent one of the most suitable technologies which can be used to meet the performance needs for e-business applications. Indeed, agents provide both an appropriate level of abstraction in modelling e-business applications and a natural merging of object orientation and knowledge-based technologies that can facilitate the incorporation of reasoning, learning and high-level dialogue capabilities to realize intelligent and adaptive applications. In particular the current interest in using agents for developing e-business applications is rising mostly because different works have shown how agent technology can be leveraged if used together with technologies exploited in the Internet, that is, semantic Web, Web services and workflows [8],[10],[17],[21],[28],[32],[33].

Semantic Web technologies appear to be the right means to provide the semantic integration between data and processes across systems that can be owned by different enterprises [9]. This technology is not completely mature yet; some major activities related to the definition of languages for expressing the semantics of the Web are still in progress [22],[11]. Nevertheless different works have shown how the powerful synergism between agents and semantic Web could be very promising [28],[40] and some efforts have been made in order to define ontology models and develop tools suitable for agents aiming at being truly semantic aware agents. The research community contributions have been mainly devoted to cope with three different issues:

- The formal definition of a standard language for expressing semantics on the web which has led to the Web Ontology Language,
- The development of integral software infrastructures, for writing semantic web applications, offering a variety of tools to engineer ontologies. These supports for the construction of ontology-oriented and ontology-based applications are mainly thought for applications written by using the Java language,
- The development of ontological supports specifically thought for multiagent systems.

The second and third points are strictly connected to the first one since OWL is considered the reference language; therefore the work carried out, starting from OWL, has developed tools more suitable for different contexts.

As far as the second point is concerned, an interesting approach is characterized by the definition of a meta-model that closely reflects the OWL syntax and semantics. This is the case of the modelling APIs of Jena [25], which is the most famous and widely used tool in the sphere of the semantic web (and recently also in the context of multiagent systems), and the OWL API [4] framework. The latter consists of a high-level programmatic interface for accessing and manipulating OWL ontologies. To date, it is shipped with a reference implementation which is main memory-based and not optimized.

Considering the third point, the focus is on the specific needs of multiagent systems, and the objective is to provide a communication support enabling agent to perform the proper semantic checks on a given content expression. A significant example of the efforts made in this direction is represented by the ontological support of JADE [24], designed to represent, using Java objects, a taxonomy of concepts. Such semantically aware agents should then be able to discover, invoke, compose and monitor those Web resources that provide services. In order to make agents able to use a service, they need a computer interpretable description of the service itself and furthermore to know the means by which it is accessible. To that purpose, a community of researchers is developing an ontology of services, called OWL-S, with the aim of providing a semantic orientation to the description of Web services.

To enable software systems for e-business applications, security issues have to be carefully analysed and sound solutions have to be deployed. A number of different solutions for the problems of authentication and authorization in open systems have been proposed in the scientific literature, and some standards have emerged through the years. Most of them are based on some kind of PKI and signed certificates issued by a Certification Authority.

In particular, this is the case of X.509, which is the best known and adopted standard for authentication and authorization. However, its weaknesses have been clearly demonstrated in a number of works [20], above all related to its (mainly politically motivated) effort to create a global directory of unique names. In any case, relying on an external entity as root of all certifications represents an additional, not directly controllable, point of failure for the whole system.

In contrast, different approaches have been proposed, based on local names. Both SDSI [1] and PetName Markup Language [29] allow local names to be used in a global scale by prefixing them with the public key of the principal defining them, in the form of `(key, name)` couples. This way, name conflicts are solved thanks to the uniqueness of the public keys. In [44], authors show that local names and YURLs are more robust than global names to phishing attacks, arguing the root for these attacks lie in the global namespace itself. Moreover, in [30],[31], authors shows that local names and a subset of the SDSI/SPKI standard [14] can be used to implement a distributed RBAC infrastructure, in which local names are interpreted as distributed roles, whose name is localized to their defining principal (key). In [14], delegation certificates are defined as s-expressions, to link separate namespaces and to delegate access rights among principals.

Local names and delegation certificates are the key to build systems adhering *trust management principles* [27]. These systems are completely distributed as they avoid any centralized authority. This way they can easily scale to large peer-to-peer networks, where each node is in charge of protecting its own resources and to show proper credentials when accessing resources of other nodes.

To conclude just a few remarks on JADE since it is considered the reference implementation of the FIPA specifications and one of the most used and promising agent development framework. The present release of JADE tries to provide agent developers with a support integrating almost all these technologies, even if in our opinion only partially. As a matter of fact, JADE agents can exploit an ontological model of the application domain to improve their interactions, are able to interact with external Web services [17] and finally different works have shown how the integration of a JADE agent with the Jess rule engine is feasible. But this simply represents a first step towards an effective integration with the aim of supporting real e-business applications. In fact a detailed analysis shows its weaknesses.

The idea which mostly inspired the design of the JADE content language and ontological support was to define an ontology independent abstract model of the content language that could be subsequently bound to any domain ontology representation expressed using an object-oriented data model. This ontological support has been conceived when the Semantic Web was on its very early stage of research and development and OWL was not already established as a standard. Consequently its expressive power is clearly limited with respect to OWL and basically allows expressing taxonomy of concepts, predicate and actions and therefore it is not able to represent completely the different application domains where JADE agent may be used.

JADE provides the integration with the JESS rule engine [26], which is probably the most known Java rule engine implementing the Rete algorithm [16]. This integration is realized through a so-called JessBehaviour that allows the encapsulation of a JESS rule engine inside a JADE agent and has the duty of storing and retrieving information in/from the rule engine. The main limits of this solution are: i) the rule engine is completely hidden to the other agents of the system and there is not any support for the cooperation among different rule-based agents (i.e., agents encapsulating a JESS rule engine) and ii) JESS is a commercial software and so we have additional costs if we plan to realize commercial applications by using JADE together with JESS.

At the beginning of the last year a new release of JADE was announced providing a service allowing the integration of agents with Web services. This service, called WSIGS (Web Services Integration Gateway Service), is a stand alone, encapsulated application that provides transparent, bidirectional transformations between JADE agent services and Web services [17]. To date, the WSIG supports only simple WSDL description of Web services, without taking into account emerging technologies related to the semantic Web, like OWL-S specifications. Another limit of this integration service is that it does not provide any means that agents can use to automatically compose Web services.

Finally, JADE also provides some basic facilities, to sign and/or encrypt ACL messages. But, to have a complete and usable security infrastructure, these facilities must be augmented with tools enabling the distributed management of trust relationships.

3. EXTENDING JADE FOR E-BUSINESS APPLICATIONS

To overcome the limits of the present release of JADE, we have realized GAIN (Grid Agent INfrastructure), an agent based framework that allows the automatic composition of Grid tasks and Web services through the use of workflow technologies [32]. Its architecture is based on a society of agents, mostly composed of two kinds of agents: component managers and workflow managers.

Each component manager is associated to one or more Web services and is responsible for the interaction with them. Through the use of the WSIG JADE add-on, the component managers are able to invoke a Web service, converting ACL messages into WSDL descriptions and vice versa. Moreover, a component manager allows a flexible provision of services defining "on the fly" the features of the services (prize, timing, etc.) through a set of business rules managed by a rule engine and modifiable by the operators of the service provider through a Web interface.

Workflow managers have the goal of supporting users in the process of building the workflows, composing external Web services and monitoring their execution. To accomplish this complex activity the workflow managers provide the users with two alternative automatic procedures:

i) Predefined workflow; the workflow is extracted from a repository of standard and common templates, e.g. templates used in previous computations. In this case the duty of the workflow manager is to support the user in the selection of the most appropriate Web services for the execution of the different workflow tasks. The workflow manager is able to select a matching service thanks to the exploitation of a shared ontology that gives a common knowledge background

to all the agents in the system.

ii) Dynamic workflow; the workflow manager, according to the user's requirements, creates a new workflow, composing the atomic services available in the system. This is done by applying a planner (we have realized extending the SGP planner [39]) that works on the operators extracted from the OWL-S descriptions of the Web services, provided by component managers. After the composition of the final workflow, the workflow manager is able to update it and possibly replace those Web services that are failed or no more available or cannot satisfy the execution time constraints.

Moreover, GAIN offers to the users the possibility of manually building workflows. In this case, a personal assistant (i.e. an agent, associated with each user active in the system, responsible for the interaction between the user and the other parts of the system) helps its user presenting her/him the tasks (Web services) that can be composed and possibly informing her/him when the realized workflow does not satisfy the composition rules, coming from the related OWL-S descriptions. When a complete workflow is realized, the user can ask its personal assistant to delegate the workflow execution to a workflow manager. The enactment is clearly a problematic phase. When a workflow is going to be executed, a Web service could be no more available due to the expiration of a timeout, a failure of a resource or other unpredictable problems. In this case the workflow manager helps the user finding a new solution, creating a new contract phase with all the component managers that are able to satisfy the task and suggesting to the user the replacement of the failed service with the new one.

So far we have given a concise description of the system architecture and the responsibilities of the major system components, intentionally leaving out the treatment of the issues connected to the tools needed by the agents, in order to carried out their activities. In the following subsections, we will go into details, illustrating our proposals and the implemented tools.

3.1 Ontological Support

As highlighted several times within this paper, the multi-agent systems "powered" with an ontological support seem to be the key to the success of the e-business applications. JADE already makes available an ontological support but as mentioned above it is quite limited. In order to provide a JADE agent with an adequate expressive power (i.e., equivalent to the one offered by OWL DL), it is necessary either to replace or to integrate the JADE ontological support. In the attempt of finding a suitable solution to this problem one has to choose among the proposals described above and others, each characterized by different domain knowledge modelling techniques and answering different needs. The majority of the research work in this field is thought for the semantic Web. But while in the vision of the semantic Web the increasing interest in ontologies is driven by the large volumes of information available and by the need of automating many information retrieval activities, in the agent context the focal point is slightly different and it is mainly on communicative acts - communications which implies actions. Agents would use ontologies to perform the proper semantic checks on a given content expression, and therefore ontologies should include concepts (objects of the domain of the discourse) but also predicates (assertions on properties of concepts) and actions (that agents can perform in the domain). Moreover a peculiar characteristic of the agent community is the heterogeneity of resources available and the roles played by different agents of a system. This leads us to claim that a one-level approach, based on a single ontology model and the corresponding tool, with the aim of being omni comprehensive seems to be seldom feasible. In our opinion a good compromise is represented by choosing different approaches in different contexts. Our solution was to realize a compound tool, called OWLBeans [8], that allows the use of ontologies described by using OWL DL [3]. These ontologies can be used by agents for performing their tasks in cooperation with other agents, for interoperating with external entities (e.g., legacy software systems) and for performing a semantic matching of Web services, described by using OWL-S and having inputs and outputs associated with concepts belonging to a domain ontology.

OWLBeans is based on a two-level approach with the aim of coping with both the issues of managing complex ontologies and of providing ontology management support to lightweight agents, which seldom need to deal with the whole complexity of a OWL DL ontology. Therefore, lightweight agents maintain the simple JADE ontology support whereas one or more dedicated agents, acting as ontology servers, are able to use and manage complete OWL DL ontologies and provide the service to the agents that need it.

The main functionality of OWLBeans is to extract JADE ontologies from OWL DL ontologies realizing a set of ontologies usable by JADE agents, with the obvious shortcoming that not all the information maintained in the original OWL ontologies are taken into account. Therefore, for all those systems that need a complete support for OWL DL ontologies, OWLBeans offers a set of ontology server agents implemented as JADE agents, providing a common knowledge base and reasoning facilities. These ontology servers use the Jena toolkit to load, maintain and reasoning about OWL ontologies. The other agents of the system do not need to know anything about the Jena toolkit given that these ontology servers provide them with a set of simple actions for querying and manipulating the ontologies. Furthermore, ontology servers take into account proper authorization mechanisms. In particular, the underlying trust management support (discussed in the following subsection) has been leveraged to implement a certificate-based access control. Only authenticated and authorized principals will be granted access to managed ontologies. A delegation mechanism allows the creation of communities of trusted entities, which can share a common ontology, centrally managed by the ontology server.

Finally, despite the fact that the JADE ontological support is quite simple, it could still be complex for some devices with limited resources such as smart phones. This is the reason why we have decided to improve OWLBeans adding a further feature which allows agents to import taxonomies and classifications from OWL ontologies, in the form of a hierarchy of Java classes with the purpose of providing very simple artefacts to access structured information. Given its modular architecture, based on an intermediate ontology model, OWLBeans also provides further functionalities, e.g., saving a JADE ontology into an OWL file, or generating a package of JavaBeans from the description provided by a JADE ontology.

3.2 Production Rule Management

As for the ontological support, in order to cope with the limits of the current JADE support for rule engines, we realized a software library, called D4J (Drools4JADE) [6], that integrates JADE agents with the Drools rule engine [13]. Drools is a well known, freeware implementation of the so-called Rete-OO algorithm. Apart of its open-source availability, one of the main advantages of Drools is exactly the fact that it is not just a literal implementation of the Rete algorithm, but rather an adaptation for the object-oriented world. This greatly eases the burden of integrating the rule engine and the application rules with the existing external objects. In Drools, asserted facts are simple Java objects, that can be modified through their public methods and properties. Where Jess requires hundreds of lines of code, for example to simply access an ACL message mapped into a Java object, Drools rules can obtain the same result in a dozen of easy-reading code lines.

D4J guarantees both the advantages of full rule-based agents, i.e., agent whose behaviour and/or knowledge is expressed by means of rules [23],[38], and the advantages of rule-enhanced agents, i.e., agents whose behaviour is not normally expressed by means of rules, but that use a rule engine as additional component to perform specific reasoning, learning or knowledge acquisition tasks [19]. In facts, in D4J, the Drools rule-engine is integrated into an agent as a JADE behaviour, but it also provides an API for interacting with it through ACL messages allowing both remote storing and retrieval of knowledge and the cooperation among different rule-based agents. Moreover, this API allows rules mobility, i.e., a rule-based agent can move a rule to another rule-based agent.

Given their nature, business rules often refer to domain specific concepts and, especially when dealing with data on the semantic Web, these concepts are part of a domain ontology. To better support this scenario, rule-enhanced JADE agents should be augmented with a tool for the automatic transformation of concepts, relations and individuals of an OWL ontology [5] to java classes, properties, and instances. The D4J framework can be integrated with OWLBeans, which enables the extraction of JavaBeans from an OWL ontology. The JavaBeans can then be directly asserted as facts into the working memory of Drools.

3.3 Distributed Trust Management

Out of the box e-business applications are not certainly possible if security problems are not analyzed and addressed. Our framework supports the implementation and deployment of secure systems, adhering trust management principles. For this purpose, local names, interpreted as distributed roles, and delegation certificates are made available, to build peer-to-peer networks of trusted entities.

The accurate release of authorizations is often the most critical point of security systems. Ideally, systems should respect the principle of *least privilege*, but this often contrasts with other requirements, as easiness of understanding, scalability and manageability. In this respect, the RBAC model [37] has proven to be a good abstraction to manage large and complex systems, up to corporate and virtual-organizations environments.

Following the RBAC model, each resource manager of our system (i.e. each node in the peer-to-peer network) has to deal with three main concepts: principals (i.e. authenticable entities which act as users of resources and services), permissions (i.e. rights to access resources or use services) and roles [12].

A many to many relationship binds principals and the roles they are assigned to. In the same way, a many to many relationship binds permissions and the roles they are granted to, thus creating a level of indirection between a principal and his access rights. This also leads to a better separation of duties (between the assignment of principals to roles and the definition of role permissions), to implement privilege inheritance schemes among superior and subordinate roles and to permit temporary delegations of some of the assigned roles towards other principals. The fundamental principle here is that each node is in charge of defining its own roles, and of assigning principals to them.

Dynamic delegation of access rights is made possible through the use of delegation certificates, whose structure is based on the theory of [14]. But we have avoided s-expressions, preferring XML to them, as it provides a better ground to exploit and integrate standard technologies. In particular, in recent times SAML [36] have emerged as a language to express properties of authenticable principals, encoded in the generic form of signed security assertions. SAML assertions can easily bind public keys to local names, and certify the links between two different local namespaces, as it happens in SDSI/SPKI certificates.

Both providers and consumers of services exploit this infrastructure to build a distributed P2P network of trust relationships, embodied and quantified in terms of signed delegation certificates. These certificates, attached to signed request messages, list the set of roles the requester is assigned to, and they are signed by trusted authorities according to customizable policies. The delegation certificates owned by the agents can also be used to delegate roles, and thus access rights, to other agents, to allow them to complete the requested tasks or to achieve delegated goals [7].

Delegation is particularly important to deal with the activation of intermediate agents, acting between the human user and the concrete service providers, i.e. personal agents, workflow managers and agents providing composite services. In this case, privileges must be forwarded in the form of delegation certificates from the user toward each agent in the chain, up to the final service provider, which will check them for consistency with local security policies. These policies are stored and managed locally as XACML documents [43].

It is worth noting that similar solutions are made possible in GRID environments by issuing *proxy certificates* [41]. These are essentially locally signed X.509 certificates, allowing not only to specify the identity of a subject, but also to grant only a limited set of privileges to it. But, being an extension to the X.509 standard, they end up with neither being fully compliant with existing infrastructures, nor avoiding the fundamental problem of centralized authorities. Besides, they do not even enjoy the advantages of XML-based solutions, as human readability, extensibility and interoperability.

To cope with the security problems coming from the remote utilization of the rule engine and from the mobility of rules, also D4J exploits this security layer to implement: i) the authentication of the agents, ii) the checking of permissions owned by the agents,

and iii) the delegation from an agent to another agents of its permissions. Therefore, only authenticated and authorized agents can successfully ask another to store and retrieve knowledge or to accept new rules. In D4J, the infrastructure is used to enforce security policies at two different levels: proper authorization is necessary to modify the working memory and the rule set of an agent; moreover, each rule is associated with a specific protection domain, limiting the resources made accessible when it is scheduled for execution.

4. AN E-TRAVELLING APPLICATION

The framework has been experimented in the realization of a prototype of an e-travelling system which provides both atomic services (e.g., the purchase of a flight ticket) and composite services (e.g., the reservation of various hotels and restaurants and the purchase of flight tickets for a business or holiday trip). The behaviour of this system can be described introducing three different kinds of actors: service providers, service brokers and users.

Service providers provide atomic services and make them available through a set of Web services.

Service brokers help other agents to find services and to compose them. All brokers share an ontological model to describe the provided services. Service providers that like to register to one of these brokers need to provide the mapping of their Web services to such an ontological model. Travel operators have also the opportunity of directly delegating the complete management of their services to one or more service brokers. In this case, travel operators and service brokers need to share the service status (i.e., the free/busy places of a flight, the free/busy rooms of a hotel, etc.) and travel operators need to provide their service brokers with the business rules in order to make them able to compute the prices of the services. Business rules are not static, but need to be dynamically updated by the travel operators. For this reason, service brokers provide a Web interface that allows a travel operator to list its business rules, add and remove new rules and identify dependencies among them.

A user interacts with a service broker through a Web interface too. She/he can either ask for an atomic service, receiving from the broker an ordered list of the feasible solutions that should satisfy her/his requirements, or a composite service, receiving, for example, information concerning the bookings and tickets needed for a trip or the different prices of the services necessary to organize a conference. In particular, in the case of a composed service, the Web interface assists her/him in defining the composed service through a graphical construction of a workflow in which each step corresponds to a component of the service. This is an iterative process that, for instance, initially allows a user to fix the ticket for flights and the booking for hotels, and then, after all the terms have been settled (dates, places, etc.), to add booking for special restaurants, purchasing of tickets for theatres plays, etc.

Moreover, service brokers maintain a profile of registered users and use such profiles to send them information about special offers that can be of their interest.

The core of the realized system is a multi-agent system that is deployed on the servers of the service brokers. Service providers have only the duty of maintaining the servers managing their services (the service management can also be delegated to the service brokers). Users need simply a Web browser and an Internet connection.

The multi-agent system is based on five types of agents: service providers, workflow managers, personal assistants, user profile managers and service brokers. Figure 1 shows the interactions between the different types of agents and between the agents and the software systems used by the providers to manage the services they offer.

Service providers are agents that combine the behaviour of a GAIN component manager agent powered by the D4J framework. These agents have the following responsibilities: i) to provide an interface (compliant to the ontology shared by service brokers) for the Web services offered by the providers' management systems, ii) to advertise the offered services at the service brokers (a service provider can offer services through different service brokers), and iii) to manage the business rules used to compute the features of booking and selling contracts.

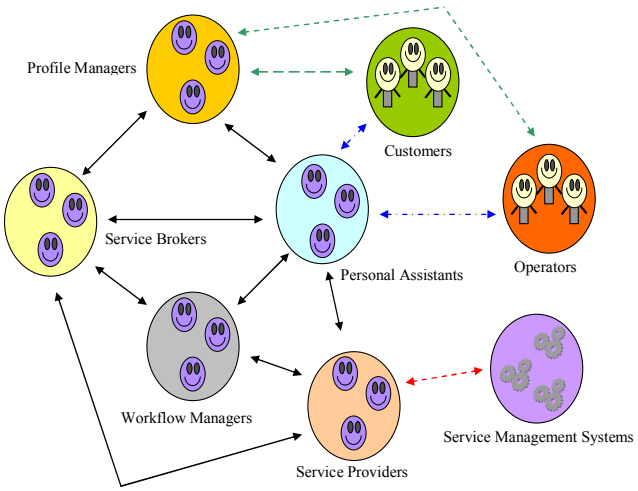


Figure 1 – Interactions between the different types of agents and between agents and service management systems.

Workflow managers are instances of the GAIN workflow manager agent and so their responsibilities concern mostly the management of workflows, strictly related to the composition of services. In particular, they are responsible for: i) checking the workflow submitted by the users, ii) building and/or completing the workflows that satisfy the requests of the users, iii) starting and monitoring the execution of the workflows, and possibly iv) updating the workflow structure if a component fails during its execution. They receive an explicit delegation from the user's personal assistant, which is responsible for providing them with all privileges needed to accomplish the task. These privileges can be further delegated to service providers, in order to enable access to crucial resources.

Personal assistants are agents that are in charge of helping service provider operators and customers of the system in their actions. In particular, they help service provider operators in the maintenance of business rules and customers in both the searching and execution of atomic services and in the construction and execution of composite services. Moreover, they have the duty of

building and updating the profile of the users of the system. Users interact with the personal assistants through Web servers. On the one hand, these Web servers provide users with the appropriate Web interfaces. On the other hand, they allow the personal assistants to push information to the users.

Personal assistants are respectively created and destroyed at the login and logoff of the users. Therefore, personal assistants need to delegate the maintenance of the user profiles to persistent agents, called profile managers. These agents are mainly responsible for exchanging the user profiles with the personal assistants when they are created and destroyed. Moreover, they have the duty of exploiting user profiles to look for new and advantageous offers (e.g., airplane tickets with reduced prize) that may be of interest and possibly to send users information about them through emails. Personal assistants are also responsible for the management of user's credentials, in the form of identity certificates submitted directly by the user at login, and other delegation certificates received by other peers in the course of business interactions.

Service brokers are the agents managing information about the services offered by the connected service providers. In fact, on the one hand, service providers register their services by the service brokers sending them an OWL-S description of their services. On the other hand, both personal assistants and workflow managers can send queries to the service brokers to look for services satisfying user requirements.

Up to now, we have not experimented the system with real users and real services, instead we have tested and evaluated the system functionalities implementing some "artificial" services and involving a group of students, acting either as service provider operators or as customers. Some of the information used by the service providers, implemented just for the experimentation of the system, comes from the Web site of some real service providers (e.g., flight companies, hotel brokers, etc.). With the view to doing it, we have realized an application able to get information from a Web site and then to make available this information through a Web service. This application is independent from the structure of the Web site because it builds queries and extracts information from the responses on the basis of an XML file containing the patterns of both queries and answers. Therefore, we are able to simulate a set of different service providers by simply defining the patterns for querying the Web site of a service.

5. CONCLUSION

In this paper, we have presented an agent-based framework for realizing e-business systems that integrates agent technology with other technologies that have found, and will find, a purpose within enterprise computing: web services, workflows, ontologies and rule engines.

We are well aware that the current multi-agent solutions, aiming at facilitating and supporting the realization of e-business applications, need to be improved since the technologies used are still not completely mature. However a lot of researchers and software developers are really interested in giving a significant contribution in this direction, driven by the motivation of providing a strengthening of the related standards and new methodologies, algorithms and implementations to realize real flexible, adaptive intelligent e-business systems [2],[15].

Our future activities will be oriented towards the aforementioned goal. In particular, we will continue working on the JADE software environment in order to both improve the integration of the JADE agents with the most interesting knowledge and internet-oriented technologies and realize real adaptive agents that will be the basis of next and future e-business applications. At present, we are working in two main directions: i) to provide a full OWL DL support through a home-made framework supplying ontology management and reasoning functionalities, with the main purpose of reducing the amount of computational resources and time required (with respect to the Jena engine), ii) to enhance the agent-based and graphical support, that users can currently use when they define their workflows or modify the set of business rules controlling their services.

6. REFERENCES

[1] Abadi, M. On SDSI's Linked Local Name Spaces. Journal of Computer Security 6(1-2):3-22.1998.

[2] AgentLink III. Agent Technology Roadmap. Available from http://www.swsi.org/resources/wsmf-paper.pdf.

[3] Akkermans, H. Intelligent E-Business - From Technology to Value. IEEE Intelligent Systems, 16(4):8-10, 2001.

[4] Bechhofer, R. Volz, and P. Lord. Cooking the semantic web with the OWL API. In Proc. Int Semantic Web Conference, pp. 659-675, Sanibel Island, FL, 2003.

[5] Bechhofer, S., van Harmelen, F., Hendler, J. Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., & Stein, L.A. OWL Web Ontology Language Reference. 2004. Available from http://www.w3.org/TR/owl-ref/.

[6] Beneventi, A., Poggi, A., Tomaiuolo, M., & Turci, P. Integrating Rule and Agent-Based Programming to Realize Complex Systems. WSEAS Trans. on Information Science and Applications, 1(1):422-427, 2004.

[7] Bergenti, F., Rimassa, G., Somacher, M., Botelho, L.M. A FIPA Compliant Goal Delegation Protocol. Communication in Multiagent Systems, Vol. 2650, pp. 223-238. 2003. Springer.

[8] Bergenti, F., Poggi, A., Tomaiuolo, M., Turci, P. An Ontology Support for Semantic Aware Agents. In Proc. Seventh International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2005 @ AAMAS), Utrecht, The Netherlands, 2005.

[9] Berners-Lee, T., Hendler, J., Lassila O. The Semantic Web - A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. 284(5):34-43, 2001.

[10] Buhler P.A., Vidal, J.M. Towards Adaptive Workflow Enactment Using Multiagent Systems. Information Technology and Management, 6(1):61-87, 2005.

[11] de Bruijn, J., Polleres, A., Lara, R., Fensel, D. OWL DL vs. OWL Flight: Conceptual Modeling and Reasoning for the Semantic Web. In Proc. of the 14th Int. World Wide Web Conference (WWW2005), pp. 623-632, Chiba, Japan, 2005.

[12] Chadwick, D.W., Otenko, A. RBAC Policies in XML for X.509 Based Privilege Management. In Proc. of the 17th Int. Conf. on Information Security (SEC2002), pp. 39-54, 2002.

[13] Drools software and documentation. Available from http://drools.org.

[14] Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T. SPKI Certificate Theory. RFC 2693, 1999.

[15] Fensel, D., Bussler, C. The Web Service Modeling Framework WSMF. Electronic Commerce Research and Applications 1(2): 113-137, 2002.

[16] Forgy, C. Rete: A Fast Algorithm for the Many Patterns/Many Objects Match Problem. Artificial Intelligence, 19(1):17-37, 1982.

[17] Greenwood, D., Callisti, M. Engineering Web Service-Agent Integration. In IEEE Conference of Systems, Man and Cybernetics, 2004. Available from http://www.whitestein.com/resources/papers/ieeesmc04.pdf.

[18] Gibbins, N., Harris, S., Shadbolt, N. Agent-based semantic web services. In Proc of the 12th International World Wide Web Conference (WWW2003), Budapest, Hungary, 2003.

[19] Gutknecht, O., Ferber, J., Michel, F. Integrating tools and infrastructures for generic multi-agent systems. In Proc. of the 5th International Conference on Autonomous Agents. Montreal, Canada, 2001.

[20] Gutmann, P. (2000). X.509 Style Guide. Available from http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt

[21] Hendler, J: Agents and the semantic web, IEEE Intelligent Systems, 16(2): 30-37, 2001.

[22] Horrocks, I. and Patel-Schneider, P. F. A proposal for an OWL rules language. In Proc. of the Thirteenth International World Wide Web Conference (WWW 2004), pp. 723-731, 2004.

[23] Hindriks, K.V., de Boer, F.S., van der Hoek, & W., Meyer, J.C. Control Structures of Rule-Based Agent Languages. In Lecture Notes In Computer Science, Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages, Vol 1555, pp. 381-396, 1998, London, UK: Springer-Verlag.

[24] JADE software and documentation. Available from http://jade.tilab.com.

[25] Jena software and documentation. Available from http://jena.sourceforge.net.

[26] JESS software and documentation. Available from http://herzberg.ca.sandia.gov/jess.

[27] Khare, R., Rifkin, A. Weaving a Web of trust, World Wide Web Journal Special Issue on Security, 2(3):77–112, 1997.

[28] Labrou, Y. Agents and ontologies for e-business. Knowledge Engineering Review, 17(1):81-85, 2002.

[29] Lambda for Humans – The PetName Markup Language. Available from http://www.erights.org/elib/capability/pnml.html

[30] Li, N. Local Names in SPKI/SDSI. In Proc of the 13th IEEE Computer Security Foundations Workshop. 2000.

[31] Li, N., Mitchell, J.M.. RT: A Role-based Trust-management Framework. In Proc of the Third DARPA Information Survivability Conference and Exposition (DISCEX III), pp. 201-212, 2003. Washington, D.C.

[32] Negri, A., Poggi, A., Tomaiuolo, M., & Turci, P. Dynamic Grid Tasks Composition and Distribution through Agents. Concurrency and Computation: Practice and Experience, 2005.

[33] Papazoglou, M.P.. Agent-oriented technology in support of e-business. Communication of ACM, 44(4):71-77, 2001.

[34] Papazoglou, M.P. The World of e-Business: Web-Services, Workflows, and Business Transactions In Lecture Notes In Computer Science, CAiSE '02/ WES '02: Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web, Vol 2512, pp. 153-173, 2002. London, UK. Springer-Verlag

[35] Poggi, A., Rimassa, G., Tomaiuolo, M. Multi-user and security support for multi-agent systems. In Proc. of WOA 2001, pp. 13-18, 2001. Modena, Italy: Pitagora.

[36] SAML - Security Assertion Markup Language. Available from http://xml.coverpages.org/saml.html.

[37] Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E. Role Based Access Control Models. IEEE Computer 29(2):38-43, 1996.

[38] Schroeder, M., & Wagner, G. Vivid agents: Theory, architecture, and applications. Applied Artificial Intelligence, 14(7):645-676, 2000.

[39] Sensory Graph Planner software and documentation. Available from http://www.cs.washington.edu/ai/sgp.html.

[40] Silva, N., Rocha, J., Cardoso, J. E-Business Interoperability Through Ontology Semantic Mapping. In Proc. of the Processes and Foundations for Virtual Organizations, pp. 315-322, 2003. Lugano, Switzerland.

[41] Tuecke, S. Welch, V. Engert, D. Pearlman L. Thompson, M. Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, 2004. Available from ftp://ftp.rfc-editor.org/in-notes/rfc3820.txt.

[42] Weikum G. Special Issue on Infrastructure for Advanced E-services, IEEE Data Engineering, 24(1), 2001.

[43] XACML - Extensible Access Control Markup Language. Available from http://xml.coverpages.org/xacml.html.

[44] YURL - Decentralized Identification. Available from http://www.waterken.com/dev/YURL.

Automated Semantic Web Service Discovery with OWLS-MX *

Matthias Klusch,
German Research Center for Artificial Intelligence
Multiagent Systems Group
Saarbruecken, Germany
klusch@dfki.de

Benedikt Fries
University of the Saarland
Computer Science Department
Saarbruecken, Germany
Develin@gmx.de

Katia Sycara
Carnegie Mellon University
Robotics Institute
Pittsburgh PA, USA
katia+@cs.cmu.edu

ABSTRACT

We present an approach to hybrid semantic Web service matching that complements logic based reasoning with approximate matching based on syntactic IR based similarity computations. The hybrid matchmaker, called OWLS-MX, applies this approach to services and requests specified in OWL-S. Experimental results of measuring performance and scalability of different variants of OWLS-MX show that under certain constraints logic based only approaches to OWL-S service I/O matching can be significantly outperformed by hybrid ones.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models; H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

OWL-S, matchmaking, information retrieval

1. INTRODUCTION

Key to the success of effectively retrieving relevant services in the future semantic Web is how well intelligent service agents may perform semantic matching in a way that goes far beyond of what standard service discovery protocols such as UPnP, Jini, or Salutation-Lite can deliver. Central to the majority of contemporary approaches to semantic Web service matching is that the formal semantics of services specified, for example, in OWL-S or WSMO are explicitly defined in some decidable description logic based ontology language such as OWL-DL [8] or F-Logic, respectively. This way, standard means of description logic reasoning can be exploited to automatically determine services that semantically match with a given service request based on the kind of terminological concept subsumption relations computed in the corresponding ontology. Prominent examples of such logic-based only approaches to semantic service discovery are provided by the OWLS-UDDI matchmaker [16], RACER [11], MAMA [5], and the WSMO service discovery approach [20].

These approaches do not exploit semantics that are implicit, for example, in patterns or relative frequencies of terms in service descriptions as computed by techniques from data mining, linguistics, or content-based information retrieval (IR). The objective of hybrid semantic Web service matching is to improve semantic service retrieval performance by appropriately exploiting means of both crisp logic based and approximate semantic matching where each of them alone would fail.

Consider, for example, a pair of real world concepts that are semantically synonymous or very closely related, but differing in their terminological definitions which are part of the underlying ontology. In particular, the crisp conjunctive logical concept expressions are differing with respect to a few pairs of unmatched logical constraints only. In this case, both concepts would be logically classified as disjoint siblings in a concept subsumption hierarchy such that any description logic reasoner would fail to detect the original real world semantic relationship. As a consequence, if the semantic comparison of both concepts is essential to discover services that are relevant to a given request, any logic based only matching approach would necessarily fail. The underpinning general problem is that standard logical specification of real world concept semantics is known to be inadequate. One operational way to cope with this problem would be to tolerate logical matching failures up to a specified extent by complementary approximate matching based on syntactic concept similarity computations. Of course, we acknowledge that the adaptation to the latter eventually is on the user's end.

In this paper, we present the first hybrid OWL-S service matchmaker called OWLS-MX, that exploits means of both crisp logic based and IR based approximate matching. Our experimental evaluation shows that under certain constraints this way of matching can indeed outperform logic

*This work has been supported by the German Ministry of Education and Research (BMBF 01-IW-D02-SCALLOPS), the European Commission (FP6 IST-511632-CASCOM), and the DARPA DAML program under contract F30601-00-2-0592.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS 2006 May 8-12, 2006, Hakodate, Hokkaido, Japan
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

based only approaches.

The remainder of the paper is structured as follows. After brief background information on OWL-S in section 2, we present the hybrid matching filters, the generic algorithm of OWLS-MX together with its variants, and a simple example in section 3. Some details on the implementation of OWLS-MX version 1.1 are given in sections 4. The experimental results of measuring performance and scalability of OWLS-MX are presented in section 5, before we briefly comment on related work in section 6, and conclude in section 7.

2. OWL-S SERVICES

In the following, we briefly introduce the essentials of the semantic Web service description language OWL-S that are needed to understand the concepts of hybrid service matching. For more details, we refer the reader to, for example, [13].

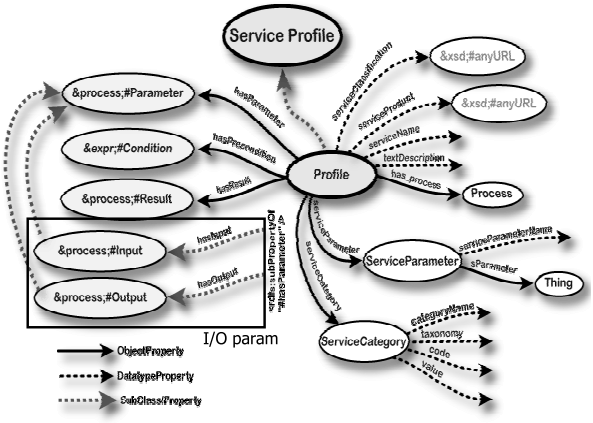


Figure 1: Parametric structure of OWL-S service profiles

OWL-S is an OWL-based Web service ontology, which supplies a core set of markup language constructs for describing the properties and capabilities of Web services in unambiguous, computer-intepretable form. The overall ontology consists of three main components: the service profile for advertising and discovering services; the process model, which gives a detailed description of a service's operation; and the grounding, which provides details on how to interoperate with a service, via messages. Specifically, it specifies the signature, that is the inputs required by the service and the outputs generated; furthermore, since a service may require external conditions to be satisfied, and it has the effect of changing such conditions, the profile describes the preconditions required by the service and the expected effects that result from the execution of the service.

To the best of our knowledge, the majority of current OWL-S matchmakers performs service I/O based profile matching that exploits defined semantics of concepts as values of service parameters hasInput and hasOutput (cf. figure 1). Exceptions include service process based approaches like in [3]. There exists no implemented matchmaker that performs an integrated service IOPE matching by means of additional reasoning on logically defined preconditions and effects. Related work on logic based semantic web rule languages such as SWRL and RuleML is ongoing.

3. HYBRID SERVICE MATCHING

Hybrid semantic service matching performed by the matchmaker OWLS-MX exploits both logic-based reasoning and content-based information retrieval techniques for OWL-S service profile I/O matching. In the following, we define the hybrid semantic filters of OWLS-MX, the generic OWLS-MX algorithm, and its five variants according to the used IR similarity metrics.

3.1 Matching filters of OWLS-MX

OWLS-MX computes the degree of semantic matching for a given pair of service advertisement and request by successively applying five different filters EXACT, PLUG IN, SUBSUMES, SUBSUMED-BY and NEAREST-NEIGHBOR. The first three are logic based only whereas the last two are hybrid due to the required additional computation of syntactic similarity values.

Let T be the terminology of the OWLS-MX matchmaker ontology specified in OWL-Lite (SHIF(D)) or OWL-DL (SHOIN(D)); CT_T the concept subsumption hierarchy of T; $LSC(C)$ the set of least specific concepts (direct children) C' of C, i.e. C' is immediate sub-concept of C in CT_T; $LGC(C)$ the set of least generic concepts (direct parents) C' of C, i.e., C' is immediate super-concept of C in CT_T; $Sim_{IR}(A,B) \in [0,1]$ the numeric degree of syntactic similarity between strings A and B according to chosen IR metric IR with used term weighting scheme and document collection, and $\alpha \in [0,1]$ given syntactic similarity threshold; $\doteq$ and $\dot{\geq}$ denote terminological concept equivalence and subsumption, respectively.

Exact match. Service S EXACTLY matches request R $\Leftrightarrow \forall$ IN_S $\exists$ IN_R: $\text{IN}_S \doteq \text{IN}_R \wedge \forall$ OUT_R $\exists$ OUT_S: $\text{OUT}_R \doteq \text{OUT}_S$. The service I/O signature perfectly matches with the request with respect to logic-based equivalence of their formal semantics.

Plug-in match. Service S PLUGS INTO request R $\Leftrightarrow \forall$ IN_S $\exists$ IN_R: $\text{IN}_S \dot{\geq} \text{IN}_R \wedge \forall$ OUT_R $\exists$ OUT_S: $\text{OUT}_S \in \text{LSC}(\text{OUT}_R)$. Relaxing the exact matching constraint, service S may require less input than it has been specified in the request R. This guarantees at a minimum that S will be executable with the provided input iff the involved OWL input concepts can be equivalently mapped to WSDL input messages and corresponding service signature data types. We assume this as a necessary constraint of each of the subsequent filters.

In addition, S is expected to return more specific output data whose logically defined semantics is exactly the same or very close to what has been requested by the user. This kind of match is borrowed from the software engineering domain, where software components are considered to plug-in match with each other as defined above but not restricting the output concepts to be direct children of those of the query.

Subsumes match. Request R SUBSUMES service S $\Leftrightarrow \forall$ IN_S $\exists$ IN_R: $\text{IN}_S \dot{\geq} \text{IN}_R \wedge \forall$ OUT_R $\exists$ OUT_S: $\text{OUT}_R \dot{\geq} \text{OUT}_S$. This filter is weaker than the plug-in filter with respect to the extent the returned output is more specific than requested by the user, since it relaxes the constraint of immediate output concept subsumption. As a consequence, the returned set of relevant services is extended in principle.

Subsumed-by match. Request R is SUBSUMED BY service $S \Leftrightarrow \forall\ \text{IN}_S\ \exists\ \text{IN}_R$: $\text{IN}_S \dot{\geq} \text{IN}_R \wedge \forall\ \text{OUT}_R\ \exists\ \text{OUT}_S$: ($\text{OUT}_S \doteq \text{OUT}_R \vee \text{OUT}_S \in \text{LGC}(\text{OUT}_R)) \wedge \text{SIM}_{IR}(\text{S, R}) \geq \alpha$. This filter selects services whose output data is more general than requested, hence, in this sense, subsumes the request. We focus on direct parent output concepts to avoid selecting services returning data which we think may be too general. Of course, it depends on the individual perspective taken by the user, the application domain, and the granularity of the underlying ontology at hand, whether a relaxation of this constraint is appropriate, or not.

Logic-based fail. Service S fails to match with request R according to the above logic-based semantic filter criteria.

Nearest-neighbor match. Service S is NEAREST NEIGHBOR of request $\text{R} \Leftrightarrow \forall\ \text{IN}_S\ \exists\ \text{IN}_R$: $\text{IN}_S \dot{\geq} \text{IN}_R \wedge \forall\ \text{OUT}_R\ \exists\ \text{OUT}_S$: $\text{OUT}_R \dot{\geq} \text{OUT}_S \vee \text{SIM}_{IR}(\text{S, R}) \geq \alpha$.

Fail. Service S does not match with request R according to any of the above filters.

The OWLS-MX matching filters are sorted according to the size of results they would return, in other words according to how relaxed the semantic matching. In this respect, we assume that service output data that are more general than requested relaxes a semantic match with a given query. As a consequence, we obtain the following total order of matching filters

EXACT < PLUG-IN < SUBSUMES < SUBSUMED-BY < LOGIC-BASED FAIL < NEAREST-NEIGHBOR < FAIL.

3.2 Generic OWLS-MX matching algorithm

The OWLS-MX matchmaker takes any OWL-S service as a query, and returns an ordered set of relevant services that match the query each of which annotated with its individual degree of matching, and syntactic similarity value. The user can specify the desired degree, and syntactic similarity threshold. OWLS-MX then first classifies the service query I/O concepts into its local service I/O concept ontology. For this purpose, it is assumed that the type of computed terminological subsumption relation determines the degree of semantic relation between pairs of input and concepts.

Auxiliary information on whether an individual concept is used as an input or output concept by any registered service is attached to this concept in the ontology. The respective lists of service identifiers are used by the matchmaker to compute the set of relevant services that I/O match the given query according to its five filters.

In particular, OWLS-MX does not only pairwisely determine the degree of logical match but syntactic similarity between the conjunctive I/O concept expressions in OWL-Lite. These expressions are built by recursively unfolding each query and service input (output) concept in the local matchmaker ontology. As a result, the unfolded concept expressions are including primitive components of a basic shared vocabulary only. Any failure of logical concept subsumption produced by the integrated description logic reasoner of OWLS-MX will be tolerated, if and only if the degree of syntactic similarity between the respective unfolded service and request concept expressions exceeds a given similarity threshold.

The pseudo-code of the generic OWLS-MX matching process is given below (cf. algorithms 1 - 3). Let $\text{INPUTS}_S = \{ \text{IN}_{S,i} | 0 \leq i \leq s \}$, $\text{INPUTS}_R = \{ \text{IN}_{R,j} | 0 \leq j \leq n \}$, $\text{OUTPUTS}_S = \{ \text{OUT}_{S,k} | 0 \leq k \leq r \}$, $\text{OUTPUTS}_R = \{ \text{OUT}_{R,t} | 0 \leq t \leq m \}$, set of input and output concepts used in the profile I/O parameters HASINPUT and HASOUTPUT of registered service S in the set *Advertisements*, and the service request R, respectively. Attached to each concept in the matchmaker ontology are auxiliary data that informs about which registered service is using this concept as an input and/or output concept.

Algorithm 1 *Match*: Find advertised services S that best hybridly match with a given request R; returns set of $(S, degreeOfMatch, SIM_{IR}(R,S))$ with maximum degree of match (*dom*) unequal FAIL (uses algs. 2 and 3 to compute *dom*), and syntactic similarity value exceeding a given threshold α.

```
1: function MATCH(Request R, α)
2:     local result, degreeOfMatch, hybridFilters = {
          SUBSUMED-BY, NEAREST NEIGHBOUR}
3:     for all (S, dom) ∈ CANDIDATES_inputset(INPUTS_R) ∧
          (S, dom') ∈ CANDIDATES_outputset(OUTPUTS_R) do
4:         degreeOfMatch ← MIN(dom, dom')
5:         if degreeOfMatch ≥ minDegree ∧ (
              degreeOfMatch ∉ hybridFilters ∨
              SIM_IR(R,S) ≥ α) then
6:             result := result ∪ { (S, degreeOfMatch,
                  SIM_IR(R,S) ) }
7:         end if
8:     end for
9:     return result
10: end function
```

In the following section, we present five variants of this generic OWLS-MX matchmaking scheme.

3.3 OWLS-MX variants

We implemented different variants of the generic OWLS-MX algorithm, called OWLS-M1 to OWLS-M4, each of which uses the same logic-based semantic filters but different IR similarity metric $\text{SIM}_{IR}(R,S)$ for content-based service I/O matching. The variant OWLS-MO performs logic based only semantic service I/O matching.

OWLS-M0. The logic-based semantic filters EXACT, PLUG-IN, and SUBSUMES are applied as defined in section 3.1, whereas the hybrid filter SUBSUMED-BY is utilized without checking the syntactic similarity constraint.

OWLS-M1 to OWLS-M4. The hybrid semantic matchmaker variants OWLS-M1, OWLS-M3, and OWLS-M4 compute the syntactic similarity value SIM_{IR} (OUT_S, OUT_R) by use of the loss-of-information measure, extended Jacquard similarity coefficient, the cosine similarity value, and the Jensen-Shannon information divergence based similarity value, respectively.

Based on the experimental results of measuring the performance of similarity metrics for text information retrieval provided by Cohen and his colleagues [4], we selected the top performing ones to build the OWLS-MX variants. These symmetric token-based string similarity measures are defined as follows.

Algorithm 2 Find services which *input* matches with that of the request; returns set of (S, dom) with minimum degree of match *dom* unequal FAIL.

function CANDIDATES$_{inputset}$(INPUTS$_R$)
  local H, dom, r
  ▷ *If a service input matches with multiple request inputs the best degree is returned*
  $H := \{ (S, \text{IN}_{S,i}, dom) \in \bigcup_{j=1..n} \text{CANDIDATES}_{input}(\text{IN}_{R_j}) \mid dom = argmax_l\{ (S, \text{IN}_{S,i}, dom_l) \mid 1 \le l \le n, 1 \le i \le s\} \}$
  ▷ *If all inputs of service S are matched by those of the request, S can be executed, and the minimum degree of its potential match is returned*
  for all $S \in Advertisement$ **do**
    if $\{ (S, \text{IN}_{S_1}, dom_1), \cdots, (S, \text{IN}_{S_s}, dom_s) \} \subseteq H$ **then**
      $r := r \cup \{ (S, \text{MIN}(dom_1, \cdots, dom_s)) \}$
    end if
  end for
  ▷ *Services with no input can always be executed and are preliminary* EXACT*-match candidates:* SERVNOIN() $= \{ (S,$ EXACT$) \mid S \in Advertisements \wedge \text{INPUTS}_S = \emptyset \}$
  ▷ *Remaining, unmatched services are at least* NEAREST NEIGHBOUR*-match candidates:* REMSERV() $= \{ (S,$ NEAREST NEIGHBOUR$) \mid S \in Advertisements \wedge \langle S, degreeOfMatch' \rangle \notin r \}$
  return $r := r \cup \text{SERVNOIN}() \cup \text{REMSERV}()$
end function

function CANDIDATES$_{input}$(IN$_{R,j}$) ▷ *Classify request input concept into ontology, and use the auxiliary concept data to collect services that at least plug-in match with respect to its input.*
  local r
  $r := r \cup \{ (S, \text{IN}_S, \text{EXACT}) \mid S \in Advertisements, \text{IN}_S \in inputs_S, \text{IN}_S \doteq \text{IN}_{R,j}, \}$
  $r := r \cup \{ (S, \text{IN}_S, \text{PLUG-IN}) \mid S \in Advertisements, \text{IN}_S \in inputs_S, \text{IN}_S \dot{\geq} \text{IN}_{R,j}, \}$
  return r
end function

Algorithm 3 Find services which *output* matches with that of the request; returns set of (S, dom) with minimum degree of match unequal FAIL.

function CANDIDATES$_{outputset}$(OUTPUTS$_R$)
  local r, dom
  if OUTPUTS$_R = \emptyset$ **then**
    return $\{ (S,$ EXACT$) \mid S \in Advertisements \}$
  end if
  for all $S \in Advertisements$ **do**
    if $(S, dom_t) \in \text{CANDIDATES}_{output}(\text{OUT}_{R,t}) \wedge dom_t \geq$ SUBSUMES **for** $t = 1..m$ **then**
      $r := r \cup \{ (S, \text{MIN}\{dom_1, \cdots, dom_m\})\}$
    else if $(S, dom_t) \in \text{CANDIDATES}_{output}(\text{OUT}_{R,t}) \wedge dom_t \in \{$ EXACT, SUBSUMES $\}$ **for** $t = 1..m$ **then**
      $r := r \cup \{ (S,$ SUBSUMED-BY $\}$
    end if
  end for
  ▷ *Any remaining, unmatched service is a potential* NEAREST NEIGHBOUR*-match:* REMSERV() $= \{ (S,$ NEAREST NEIGHBOUR$) \mid S \in Advertisements \wedge S \notin r \}$
  return $r := r \cup \text{REMSERV}()$
end function

function CANDIDATES$_{output}$(OUT$_{R,t}$) ▷ *Classify request output concept into ontology, and use the auxiliary concept data to collect services with output concepts that match with* OUT$_{R,t}$.
  local r
  $r := r \cup \{ (S,$ EXACT$) \mid \text{OUT}_S \doteq \text{OUT}_{R,t} \}$
  $r := r \cup \{ (S,$ PLUG-IN$) \mid \text{OUT}_S \in \text{LSC}(\text{OUT}_{R,t}) \wedge S \notin r \}$
  $r := r \cup \{ (S,$ SUBSUMES$) \mid \text{OUT}_S \dot{\leq} \text{OUT}_{R,t} \wedge S \notin r \}$
  $r := r \cup \{ (S,$ SUBSUMED-BY$) \mid \text{OUT}_S \in \text{LGC}(\text{OUT}_{R,t}) \}$
  return r
end function

- The *cosine* similarity metric

$$Sim_{Cos}(S,R) = \frac{\vec{R}\cdot\vec{S}}{||\vec{R}||_2^2 \cdot ||\vec{S}||_2^2} \quad (1)$$

with standard TFIDF term weighting scheme, and the unfolded concept expressions of request R and service S are represented as n-dimensional weighted index term vectors $\vec{R}$ and $\vec{S}$ respectively. $\vec{R}\cdot\vec{S} = \sum_{i=1}^{n} w_{i,R} \times w_{i,S}$, $||X||_2 = \sqrt{\sum_i^n w_{i,X}^2}$, and $w_{i,X}$ denotes the weight of the i-th index term in vector X.

- The *extended Jacquard* similarity metric $Sim_{EJ}(S,R) =$

$$\frac{\vec{R}\cdot\vec{S}}{||\vec{R}||_2^2 + ||\vec{S}||_2^2 - \vec{R}\cdot\vec{S}} \quad (2)$$

with standard TFIDF term weighting scheme.

- The intensional *loss of information* based similarity metric $Sim_{LOI}(S,R) =$

$$1 - \frac{LOI_{IN}(R,S) + LOI_{OUT}(R,S)}{2} \quad (3)$$

$$LOI_x(R,S) = \frac{|PC_{R,x} \cup PC_{S,x}| - |PC_{R,x} \cap PC_{S,x}|}{|PC_{R,x}| + |PC_{S,x}|} \quad (4)$$

with $x \in \{IN, OUT\}$, $PC_{R,x}$ and $PC_{S,x}$ set of primitive components in unfolded logical input/output concept expression of request R and service S

- The *Jensen-Shannon information divergence* based similarity measure $Sim_{JS}(S,R) = log2 - JS(S,R) =$

$$\frac{1}{2log2}\sum_{i=1}^{n} h(p_{i,R}) + h(p_{i,S}) - h(p_{i,R} + p_{i,S}) \quad (5)$$

with probability term frequency weigthing scheme, *e.g.*, $p_{i,R}$ denotes the probability of i-th index term occurrence in request R, and $h(x) = -xlogx$,

The extended Jacquard metric is a standard for measuring the degree of overlap as the ratio of the number of shared terms (primitive components) of unfolded concepts of both service and request, and the number of terms possessed by either of them. In contrast to the TFIDF/cosine similarity metric, it does not favor the document with common terms. The Jensen-Shannon measure is based on the information-theoretic, non-symmetrical Kullback-Leibler divergence measure. It measures the pairwise dissimilarity of conditional probability term distributions between service and request text rather than looking at the whole collection as it is the case for the TFIDF/cosine, or the extended Jacquard metric. The loss of (intensional) information in case some concept A is terminologically substituted by concept B, can be measured as the inverse ratio of the number of matching primitive components with those which remain unmatched in terminologically disjoint unfolded concept constraints. The symmetric LOI-based similarity value for a given pair of service and request is then computed analogously for all I/O concept definitions involved.

3.4 Example

Let us illustrate the hybrid service retrieval with OWLS-MX by means of a simple example. Suppose the concept subsumption hierarchy or taxonomy of the OWLS-MX matchmaker ontology, the service request R for physicians of hospital h that provide treatment to patient p, and relevant service advertisements S_1 and S_2 are as shown in figure 2.

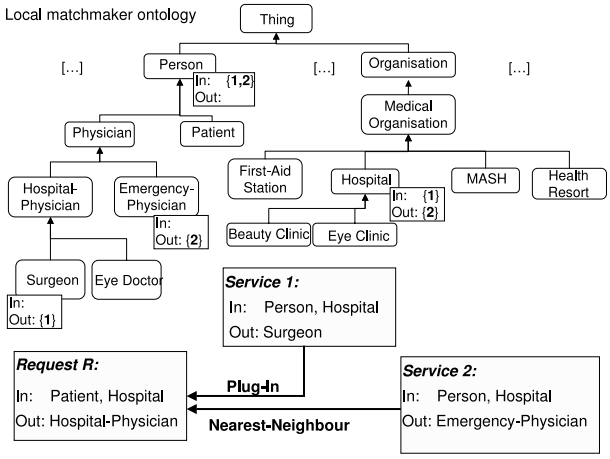


Figure 2: Example of hybrid service matching with OWLS-MX

Service S_1 is considered semantically relevant to request R, since it returns for any given person p and hospital h, the individual surgeon of h that operated on p. Likewise, service S_2 is relevant to R, since it returns those emergency physicians who provided emergency treatment to p before her transport to hospital h. Hence, both services S_1 and S_2 should be returned as matching results to the user.

However, the logic based only variant OWLS-M0 determines S_1 as plug-in matching with R but fails to return S_2, since the formal semantics of the output concept siblings "emergency physician" and "hospital physician" in the ontology are terminologically disjoint. In this example, the set of terminological constraints of unfolded concepts c correspond to the set of primitive components (c^p) of which the individual concepts are canonically defined in the matchmaker ontology T. Hence, the unfolded concept expressions are as follows.

- unfolded(Patient, T) = (and Patient^p Person^p)
- unfolded(Hospital, T) = (and Hospital^p (and MedicalOrganisation^p Organisation^p))
- unfolded(HospitalPhysician, T) = (and HospitalPhysician^p (and Physician^p Person^p))
- unfolded(Surgeon, T) = (and Surgeon^p (and HospitalPhysician^p (and Physician^p Person^p)))
- unfolded(EmergencyPhysician, T) = (and EmergencyPhysician^p (and Physician^p Person^p))

As a result, for example, OWLS-M1 would return S_1 as plug-in matching service with syntactic similarity value of Sim_{LOI} $(R, S_1) = 0.87$. In contrast to OWLS-MO, it also returns S_2, since this service is nearest-neighbor matching with the request R: Their implicit semantics exploited by the IR similarity metric LOI (cf. (5), (6)) with Sim_{LOI}

$(R, S_2) = \frac{(1-\frac{5-4}{5+4})+(1-\frac{4-2}{3+3})}{2} = 0.78 \geq \alpha = 0.7$ is sufficiently similar. Our preliminary experimental results show that this kind of matching relaxation may be useful in practice.

4. IMPLEMENTATION

We implemented the OWLS-MX matchmaker variants version 1.1 in Java using the OWL-S API 1.1 beta with the tableaux OWL-DL reasoner Pellet developed at university of Maryland (cf. `http://www.mindswap.org`). As the OWL-S API is tightly coupled with the Jena Semanic Web Framework, developed by the HP Labs Semantic Web research group (cf. `http://jena.sourceforge.net/`), the latter is also used to modify the OWLS-MX matchmaker ontology.

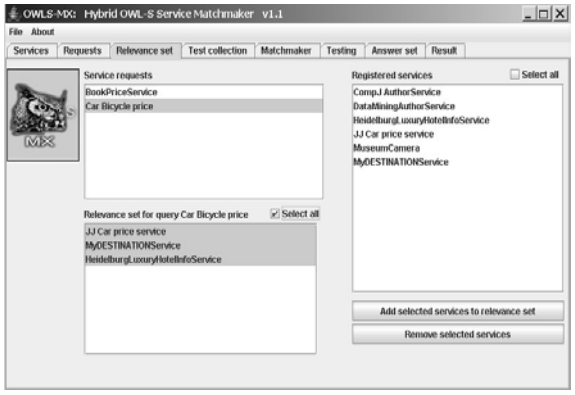


Figure 3: OWLS-MX v1.1 screenshot: Definition of service request and relevance set

Figures 3 to 5 show some screenshots of the OWLS-MX version 1.1 graphical user interface.

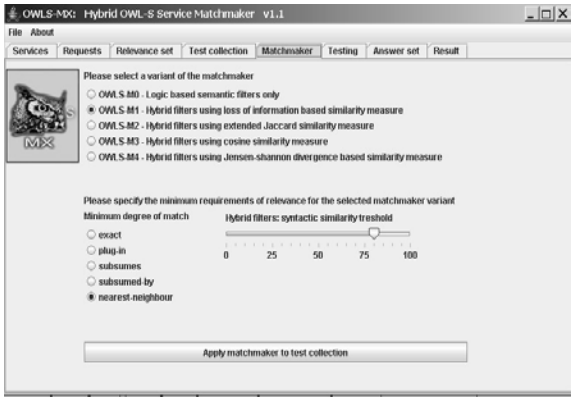


Figure 4: OWLS-MX v1.1 screenshot: Selection of OWLS-MX variant

After parsing service advertisements and requests, the respective input and output concepts are analysed and, if necessary, added to the local matchmaker ontology together with auxiliary data on their unfolding. As a consequence, the matchmaker ontology is dynamically built and growing with the number of services and underlying ontologies loaded. In addition, the matchmaker ontology is extended with auxiliary information for each concept whether it is used as an input or output concept of which service registered at the matchmaker. Service requests are treated similarly, except that they are not stored in the extended matchmaker ontology.

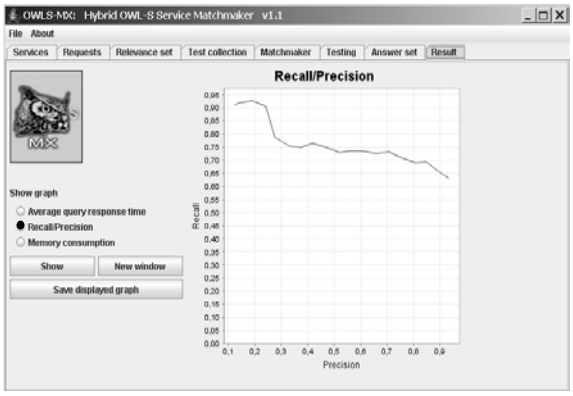


Figure 5: OWLS-MX v1.1 screenshot: Display of selected type of results (performance)

For each service request concept, the service identifiers attached to its immediate parent and child concepts of the enhanced matchmaker ontology are retrieved. The semantic degree of matching for each service is then determined by applying the semantic filters on this set of matching candidates. After this step, the syntactic similarity is computed by applying the selected IR similarity metric to the strings of unfolded concepts of the query and each registered service. Both the semantic degree of match and the syntactic similarity value determine the hybrid degree of matching of one service with the request. If this hybrid degree is better than or equal to the minimum degree specified by the user, then this service will be returned as potentially relevant.

In practice, OWLS-MX spend the largest amount of time with classifying the ontologies used by the registered services to check for new concepts not known to the matchmaker, and then to classify them into the matchmaker ontology.

5. EXPERIMENTAL EVALUATION

For measuring the service I/O retrieval performance of each OWLS-MX variant we used the OWL-S service retrieval test collection OWLS-TC v2. This collection consists of more than 570 services specified in OWL-S 1.1 covering seven application domains, that are education, medical care, food, travel, communication, economy, and weaponry. The majority of these services were retrieved from public IBM UDDI registries, and semi-automatically transformed from WSDL to OWL-S. OWLS-TC v2 provides a set of 28 test queries each of which is associated with a set of 10 to 20 services that two of the co-authors subjectively defined as relevant according to the standard TREC definition of binary relevance [17] [1]. The collection OWLS-TC v2 is available as open source at
`http://projects.semwebcentral.org/projects/owls-tc/`.

In terms of measuring the retrieval performance of each OWLS-MX variant, we adopted the evaluation strategy of

[1]Please note, that no standardized test collection for OWL-S service retrieval does exist yet. Therefore, like with any other reported results on retrieval performance of alternative OWL-S service matchmakers developed by different research groups world wide, we have to consider both our test collection and experimental results as preliminary.

micro-averaging the individual precision-recall curves [18]. Let Q be the set of test queries (service requests) in OWLS-TC, A the sum of relevant documents of all requests in Q, A_R the answer set of relevant services (service advertisements) for request $R \in Q$. For each request R, we consider $\lambda = 20$ steps up to its maximum recall value, and measure the number $B_{\lambda R}|$ of relevant documents retrieved (recall) at each of these steps. Similarly, we measure related precision with the number B_λ of retrieved documents at each step λ. The micro-averaging of recall and precision (at step λ) over all requests, as we used it for performance evaluation is then defined as

$$Rec_\lambda = \sum_{R \in Q} \frac{|A_R \cap B_{\lambda R}|}{|A|}, Prec_\lambda = \sum_{R \in Q} \frac{|A_R \cap B_{\lambda R}|}{|B_\lambda|} \quad (6)$$

The micro-averaged R-P curves of the top and worse performing IR similarity metric together with those for the OWLS-MX variants as well as the average query response time plots are displayed in figures 4 and 5, respectively.

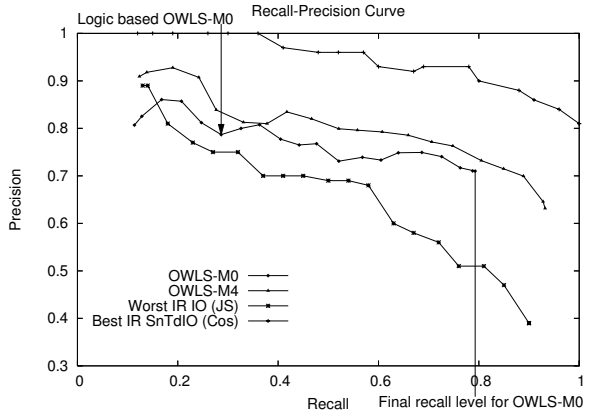


Figure 6: Recall-precision performance of logic based OWLS-M0 vs best hybrid OWLS-M4 vs IR based service I/O matching

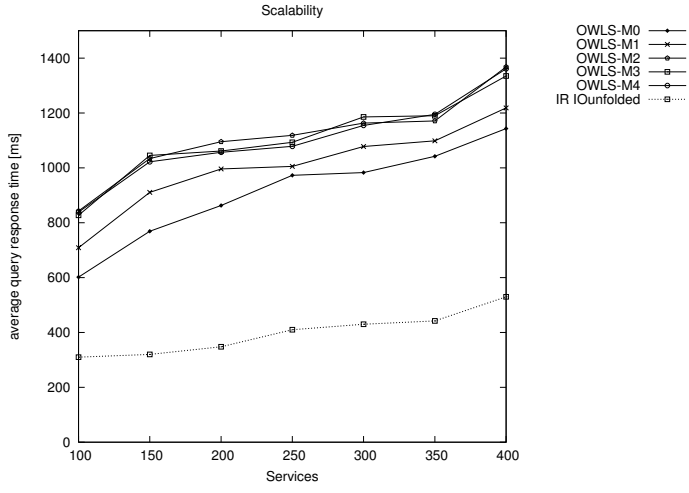


Figure 7: Average query response time of OWLS-MX vs IR based service matching

These experimental results provide, in particular, evidence in favor of following conclusions.

- The best IR similarity metric (Cosine/TFIDF) applied to the concatenated unfolded service profile I/O concept expressions performs close to the pure logic based OWLS-M0 (see figure 6: Best IR SnTdIO (Cos) vs. OWLS-M0). But OWLS-M0 is only superior to IR based matching (cf. figure 6: Worst IR IO (JS) denoting Jensen-Shannon divergence based similarity metric) at the very expense of its recall. However, for discovering semantic web services, precision may be more important to the user than recall, since the set of relevant services is supposed to be subject to continuous change in the semantic Web in practice.

- Pure logic based semantic matching by OWLS-M0 can be outperformed by hybrid semantic matching, in terms of both recall and precision. That is the case, for example, by use of the best performing hybrid matchmaker OWLS-M4 (cf. figure 6). The main reason for this is, that the additional IR based similarity check of the nearest-neighbor filter allows OWLS-M1 to M4 to find relevant services that OWLS-M0 would fail to retrieve.

- Hybrid semantic matching by OWLS-MX can be outperformed by each of the selected syntactic IR similarity metrics to the extent additional parameters with natural language text content are used. That is the case, for example, by applying the cosine similarity metric to the extended set of service profile parameters including not only hasInput and hasOutput but also serviceName and textDescription (cf. figure 6).

- Both pure logic based and all hybrid OWLS-MX matchmakers are significantly outrun by IR based service retrieval in terms of average query response time almost by size of magnitude (cf. figure 7). This is due to the additional computational efforts required by OWLS-MX to determine concept subsumption relationships in NEXPTIME description logic OWL-DL based on the imported large ontologies the OWL-S services refer to.

6. RELATED WORK

Quite a few semantic Web service matchmakers have been developed in the past couple of years such as the OWLS-UDDI matchmaker [16], RACER [11], SDS [12], MAMA [5], HotBlu [6], and [10]. Like OWLS-MX, the majority of them does perform profile based service signature (I/O) matching. Alternate approaches propose service process-model matching [3], recursive tree matching [2], P2P discovery [1], automated selection of WSMO services [20] and METEOR-S for WSDL-S services [19]. Except LARKS [15], none of them is hybrid, in the sense that it exploits both explicit and implicit semantics by complementary means of logic based and approximate matching. To the best of our knowledge, OWLS-MX is the only hybrid matchmaker for OWL-S services yet.

The OWLS-MX matchmaker bases on LARKS [15]. However, LARKS differs from OWLS-MX in that it uses a proprietary capability description language and description logic different from OWL-S and OWL-DL, respectively. Furthermore, LARKS does not perform any subsumes and subsumed-

by nor nearest-neighbour matching, and has not been experimentally evaluated yet.

The purely logic based variant OWLS-M0 of OWLS-MX is quite similar to the OWLS-UDDI matchmaker [16] but differs from it in several aspects. Firstly, the latter makes use of a different notion of plug-in matching, and does not perform additional subsumed-by matching. Secondly, OWLS-M0 classifies arbitrary query concepts into its dynamically evolving ontology with commonly shared minimal basic vocabulary of primitive components instead of limiting query I/O concepts to terminologically equivalent service I/O concepts in a shared static ontology as the OWLS-UDDI matchmaker does.

7. CONCLUSIONS

Our approach to hybrid semantic Web service matching, called OWLS-MX, utilizes both logic based reasoning and IR techniques for semantic Web services in OWL-S. Experimental evaluation results provide evidence in favor of the proposition that building semantic Web service matchmakers *purely* on description logic reasoners may be insufficient, hence should give a clear impetus for further studies, research and development of more powerful approaches to service matching in the semantic Web across disciplines.

8. REFERENCES

[1] F. Banaei-Kashani, C.-C. Chen, and C. Shahabi. Wspds: Web services peer-to-peer discovery service. In *Proceedings of International Symposium on Web Services and Applications (ISWS)*, 2004.

[2] S. Bansal and J. Vidal. Matchmaking of web services based on the daml-s service model. In *Proceedings of Second International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Melbourne, Australia*, 2003.

[3] A. Bernstein and M. Klein. Towards high-precision service retrieval. In *IEEE Internet Computing, 8(1):30-36*, 2004.

[4] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proc. IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03)*. DBLP at http://dblp.uni-trier.de, 2003.

[5] S. Colucci, T. D. Noia, E. D. Sciascio, F. Donini, and M. Mongiello. Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. In *Proc. 6th Int Conference on Electronic Commerce (ICEC 2004)*. ACM Press, 2004.

[6] I. Constantinescu and B. Faltings. Efficient matchmaking and directory services. In *Proceedings of IEEE/WIC International Conference on Web Intelligence*, 2003.

[7] T. Grabs and H.-J. Schek. Flexible information retrieval on xml documents. In *Intelligent Search on XML Data, Applications, Languages, Models, Implementations, and Benchmarks*. Springer, 2003.

[8] I. Horrocks, P. Patel-Schneider, and F. van Harmelen. From shiq and rdf to owl: The making of a web ontology language. *Web Semantics, 1(1), Elsevier*, 2004.

[9] U. Keller, R. Lara, A. Polleres, and D. Fensel. Automatic location of services. In *Proceedings of European Semantic Web Conference (ESWC), Springer, LNAI 3532*, 2005.

[10] M. Klein and B. Koenig-Ries. Coupled signature and specification matching for automatic service binding. In *Proceedings of European Conference on Web Services, Springer, LNAI, 183-197*, 2004.

[11] L. Li and I. Horrock. A software framework for matchmaking based on semantic web technology. In *Proc. 12th Int World Wide Web Conference Workshop on E-Services and the Semantic Web (ESSW 2003)*, 2003.

[12] D. Mandell and S. McIllraith. A bottom-up approach to automating web service discovery, customization, and semantic translation. In *Proc. 12th Int Conference on the World Wide Web (WWW 2003)*. ACM Press, 2003.

[13] OWL-S. Semantic markup for web services; w3c member submission 22 november 2004. http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/.

[14] A. Sheth, C. Ramakrishnan, and C. Thomas. Semantics for the semantic web: The implicit, the formal, and the powerful. *Semantic Web and Information Systems, 1(1), Idea Group*, 2005.

[15] K. Sycara, M. Klusch, S. Widoff, and J. Lu. Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. *Autonomous Agents and Multi-Agent Systems, 5(2), Kluwer*, 2002.

[16] K. Sycara, M. Paolucci, A. Anolekar, and N. Srinivasan. Automated discovery, interaction and composition of semantic web services. *Web Semantics, 1(1), Elsevier*, 2003.

[17] TREC. Text retrieval conference. http://trec.nist.gov/data/.

[18] C. van Rijsbergen. *Information Retrieval*. 1979.

[19] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller. Meteor-s wsdi: A scalable infrastructure of registries for semantic publication and discovery of web services. *Information Technology and Management*, 2004.

[20] U. Keller, R. Lara, H. Lausen, A. Polleres, D. Fensel. Automatic Location of Services. In *Proceedings of the 2nd European Semantic Web Conference*, LNCS 3532, 2005.

Ontology-Guided Learning to Improve Communication between Groups of Agents

Mohsen Afsharchi
Department of Electrical and Computer Engineering
University of Calgary
Calgary, Canada
mafsharc@ucalgary.ca

Behrouz H.Far
Department of Electrical and Computer Engineering
University of Calgary
Calgary, Canada
far@ucalgary.ca

Jörg Denzinger
Department of Computer Science
University of Calgary
Calgary, Canada
denzinge@cpsc.ucalgary.ca

ABSTRACT

We present a general method for agents using ontologies as part of their knowledge representation to teach each other concepts to improve their communication and thus cooperation abilities. Our method aims at getting positive and negative examples for a concept only very vaguely understood by a particular agent from the other agents. This agent then uses one of the known concept learning methods to learn the concept in question, involving the other agents again by taking votes in case of conflicts in the received knowledge. This method allows agents that are not sharing common ontologies to establish common grounds on concepts known only to some of them, if these common grounds are needed during cooperation. While the concepts learned by an agent are only compromises between the views of the other agents, the method nevertheless enhances the autonomy of agents using it substantially.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; I.2.6 [**Learning**]: Concept learning

General Terms

Algorithms, Experimentation

Keywords

ontology, concept learning, multi-agent communication

1. INTRODUCTION

The ability to communicate efficiently is one of the cornerstones of cooperation between human beings. In order to communicate efficiently, not only a common language is needed, people communicating also need to have some basic understanding of world concepts and they can use this basic understanding to teach each other additional concepts, at least up to the point where the understanding of the new concepts by the different people overlaps largely.

If we look at research in multi-agent systems concerning communication between agents, then many researchers have assumed that it is possible to not only establish a common language among agents, but also a complete common understanding of all the concepts the agents communicate about. In cooperative multi-agent systems which aim at solving a particular problem, usually all of the agents are designed by one group of developers and the problem already provides all of the concepts and the needed understanding (i.e. semantics) of them, so that the above assumption is valid. If we look at systems that have agents designed by different developers, many researchers assume that it is possible to have a common ontology for these agents and that the agent developers naturally will integrate this common ontology into their agents, thus allowing for easy communication (and understanding) among agents. But the assumption of a common ontology is often too strong or unrealistic. For many application areas, there is no agreement among developers on one ontology for the area, for many areas the potential ontologies are large, unwieldy and encompass more than a particular agent most probably will ever need and implementing complex ontologies can also easily lead to discrepancies between implementations.

Recently, the idea of having agents *learn* concepts (or languages) from other agents has been suggested as a solution to the problems above (see [3] for learning, resp. making up, a language and [11] for learning a concept). The work in [11] has focused on interactions between two agents only and single concepts, while [3] was not concerned with concepts. In this paper, we present a general method for having agents learn concepts from several other agents and our method also makes use of information from the ontologies in which the concepts to learn are integrated.

The basic ideas behind our method are to have an agent, that realizes that there seems to be a concept it does not currently know of but expects to need to know, query the other agents about this concept by providing features (and their values) or examples the agent thinks are associated with the concept. The queried agents provide the agent with positive and negative examples from their understanding of which of their concepts (i.e. concepts known by them) seem to fit the query which allows the learning agent to use one of the known concept learning techniques from machine learning. To help focus the negative examples, the teaching agents make use of selected relations between concepts from their

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

ontologies. The agent that wants to learn the concept deals with the fact that the other agents in this group might not totally agree on which examples fit the concept and which not, by letting the teaching agents vote on the examples for which it got contradictory information in the first place. Our learning agent also uses information from the teacher agents about the paths to the concept of interest in their ontologies to prepare its own ontology for potential future concepts it might have to learn.

Our experiments with agents with ontologies representing courses and the organizational unit structure of universities show that our approach allows an agent to learn needed concepts and that what it learns is a compromise of what the other agents think about the concept in question (essentially the common ground of the agents regarding the concept). This is possible, even if the agents do not use the same features. The experiments also show that the use of ontology information allows for more accuracy in the learned concept if the number of exchanged examples is low, thus improving on the amount of information exchanged between agents.

2. BASIC DEFINITIONS

In this section, we will provide some basic definitions about ontologies and agents and we will also provide the instantiations of these concepts that we require for our methods.

2.1 Ontologies and Concepts

The usage of the term "ontology" in (computer science) literature faces problems very similar to the usage of the term "agent": there is no agreed-upon formal definition for the term, but nevertheless it is used very intensively and there are many systems out there that come with a (somehow) build-in definition of the term. Common to most usages of ontology is that it is considered a way for representing objects or concepts in a hierarchy with additional ways to define relationships between these objects/concepts.

This is very nicely reflected by Stume's formal definition (see [9]) who defines a *core ontology* as a structure $\mathcal{O} := (C, \leq_C, R, \sigma, \leq_R)$. C and R are two disjoint sets and the elements of C are called *concept identifiers* and the elements of R are so-called *relation identifiers*. $\leq_C$ and $\leq_R$ are partial orders on C, respectively R, called *concept hierarchy* or *taxonomy* ($\leq_C$), respectively *relation hierarchy* ($\leq_R$). $\sigma : R \rightarrow C^+$ is a function providing a signature for a relation such that $|\sigma(r_1)| = |\sigma(r_2)|$ for every $r_1, r_2 \in R$ with $r_1 \leq_R r_2$ and for every projection π_i $(1 \leq i \leq |\sigma(r_1)|)$ of the vectors $\sigma(r_1)$ and $\sigma(r_2)$ we have $\pi_i(\sigma(r_1)) \leq_C \pi_i(\sigma(r_2))$. If $c_1 \leq_C c_2$ for $c_1, c_2 \in C$, then c_1 is called a *subconcept* of c_2 and c_2 is a *superconcept* of c_1.

By viewing an ontology $\mathcal{O}$ as a structure on top of a set of concepts and a set of relations, this definition is able to cover a lot of the definitions of ontologies in the literature, but without providing a more precise definition of what concepts are, we are far from anything that can be used by agents to provide a basis for communication. Many works in databases and machine learning define concepts as collections of objects that share certain feature instantiations and for this work we will follow this guidance.

So, in the following we assume that we have a set $\mathcal{F} = \{f_1, ..., f_n\}$ of features and for each feature f_i we have its domain $D_i = \{v_{i1}, ..., v_{im_i}\}$ that defines the possible values the feature can have. Then an object $o = ([f_1 = v_1], ..., [f_n = v_n])$ is characterized by its values for each of the features (often one feature is the identifying name of an object and then each object has a unique feature combination). By $\mathcal{U}$ we denote the set of all (possible) objects. In machine learning, often every subset of $\mathcal{U}$ is considered as a concept. In databases and in this work we want to be able to characterize a concept by using feature values. Therefore, a *symbolic concept* C_k is denoted by $C_k = ([f_1 = V_1], ..., [f_n = V_n])$ where $V_i = \{v'_{i1}, ..., v'_{ij_i}\} \subseteq D_i$ (if $V_i = D_i$ then we often omit the entry for f_i). An object $o = ([f_1 = v_1], ..., [f_n = v_n])$ is *covered* by a concept C_k, if for all i we have $v_i \in V_i$. In an ontology according to the definition above, we assign a concept identifier to each symbolic concept that we want to represent in our ontology.

Obviously, the relation $\leq_C$ is supposed to be connected with how concepts are defined. In the literature, taxonomies are often build using the subset relation, i.e. we have

$C_i \leq_C C_j$ iff for all $o \in C_i$ we have $o \in C_j$.

This definition of $\leq_C$ produces a partial order on C as defined above and we will use this definition in the following for the ontologies that our agents use.

From the point of view of knowledge representation the really interesting part of ontologies are the relations in R that a particular ontology allows. This is also the part where we see a lot of differences between different authors. In general, all possible relations between tuples of concepts can be used in ontologies, but usually researchers assume a small set of build-in relations and tool developers sometimes throw in the possibility to have (limited) user-defined relations. But unfortunately, different ontologies can use the same relation identifiers for different build-in relations, so that there is quite some confusion in this area. So, if we have two systems build by different people using ontologies over the same set $\mathcal{U}$ then it is very important to either identify those relations that occur in both ontologies or to find ways how the knowledge contained in the (usage of) relations in one ontology can be used in communications between the systems. In this work, we will show such a usage for one relation that we have called `is-similar-to` with $\sigma(\texttt{is-similar-to}) \in C^2$.

2.2 Agents

A general definition that can be instantiated to most of the views of agents in literature sees an agent $\mathcal{A}g$ as a quadruple $\mathcal{A}g = (Sit, Act, Dat, f_{Ag})$. Sit is a set of situations the agent can be in, the representation of a situation naturally depending on the agent's sensory capabilities, Act is the set of actions that $\mathcal{A}g$ can perform and Dat is the set of possible values that $\mathcal{A}g$'s internal data areas can have. In order to determine its next action, $\mathcal{A}g$ uses $f_{Ag} : Sit \times Dat \rightarrow Act$ applied to the current situation and the current values of its internal data areas.

For the agents that we are interested in, we can instantiate this definition a little bit more. We want to focus on the knowledge representation used by agents, so we start by looking more closely at Dat. We assume that every element of Dat of an agent $\mathcal{A}g$ contains an ontology area $\mathcal{O}_{Ag}$ as defined in the previous subsection that represents the agent's view and knowledge of concepts. For the concepts in the taxonomy of $\mathcal{O}_{Ag}$ there might be additional data, beyond features, that the agent requires from time to time (see Section 3). Naturally, there will be additional data areas representing information about the agent itself, knowledge about other agents and the world that the designer of the agent

may want to be represented differently than in $\mathcal{O}_{Ag}$. But in the rest of this paper, we will concentrate on how the agent uses and manipulates its ontology.

The actions of an agent obviously depend a lot on the application area the agent is designed for. We require our agents to be able to communicate with other agents using information from the agent's ontology and to manipulate this ontology based on information received from other agents and the agent's own deduction actions that include actions performing learning. Again, we will provide more information in the next section.

An element of Sit usually contains parts representing observations of other agents and of the environment the agent is in. In this paper, we assume that among the observations of an agent are all messages send by other agents since the last situation an agent was in. And we will specify the parts of f_{Ag} that deal with the relevant messages and the relevant knowledge in and around the ontology of an agent in the next section.

3. LEARNING OF NEW CONCEPTS

The goal of this research is to develop a method how an agent can learn new concepts for its ontology with the help of other agents. This naturally assumes that not all agents have the same ontology (otherwise learning would not be necessary). In fact, we additionally assume that there are only some base features $\mathcal{F}_{base} \subseteq \mathcal{F}$ that are known, respectively can be recognized, by all agents and that there are only some base symbolic concepts C_{base} that are known to all agents by name, their feature values for the base features and the objects that are covered by them. Outside of this base common knowledge, individual agents may come with additional features they can recognize and additional concepts they know. Agents might refer to the same such features and concepts by different names and they may have features and concepts that have the same name but are not the same. While all the ontologies used by the agents will use as taxonomy the subset-relation, agents may use different other relations in their ontologies and two agents cannot rely on same relation identifiers referring to the same relation and vice versa, again.

Given this setting, agents will develop problems in working together, since the common grounds for communication are not there or too small. To solve this problem, agents need to acquire the concepts outside of C_{base} that other agents have, at least those concepts that are needed to establish the necessary communication to work together on a given task. Our basic idea is to have an agent *learn* required concepts (or at least a good approximation) with the *help* of the other agents. Due to the potential differences in the ontologies of agents, objects that are positive and negative examples for a concept will play the major role in teaching an agent a new concept. We will assume in the following that the identifying name of an object is a feature in $\mathcal{F}_{base}$. We do not see this as a big limitation, since it is usually not too difficult to establish a clear identification of objects. For example, if the objects are part of the environment, pointing to a particular object is sufficient to identify it.

In the following, we will first present the general interaction scheme between agents that is our method to have an agent learn a particular concept. Then we will focus in more detail on the important steps in this interaction scheme.

3.1 Our general interaction scheme

Although we want all agents to be able to learn new concepts, for explaining our interaction scheme we designate one agent, $\mathcal{A}g_L$, as the one that wants to learn a new concept and the other agents, $\mathcal{A}g_1$,...,$\mathcal{A}g_m$, will be its teachers. $\mathcal{A}g_L$ has an ontology $\mathcal{O}_L = (C_L, \leq_C, R_L, \sigma_L, \leq_{R_L})$ and knows a set of features $\mathcal{F}_L$. Analogously, $\mathcal{A}g_i$ has as ontology $\mathcal{O}_i = (C_i, \leq_C, R_i, \sigma_i, \leq_{R_i})$ and knows a set of features $\mathcal{F}_i$. For a concept c known to the agent $\mathcal{A}g_i$, this agent has in its data areas a set $pex_i^c \subseteq \mathcal{U}$ of positive examples for c that it can use to teach c to $\mathcal{A}g_L$. Part of Act_L are actions `QueryConcept`, `AskClassify`, `Learn`, and `Integrate`, while part of the Act_is are the actions `FindConcept`, `CreateNegEx`, `ReplyQuery`, `ClassifyEx` and `ReplyClass`; all with appropriate arguments. These actions form our interaction scheme in the following manner:

1. $\mathcal{A}g_L$ determines it needs to know about a particular concept c_{goal} and performs `QueryConcept`("c_{goal}") to inform the other agents about this need.
2. Each agent $\mathcal{A}g_i$ reacts to $\mathcal{A}g_L$'s query by (see 3.3):
   (a) performing `FindConcept`("c_{goal}"), which leads to a set of candidate concepts C_i^{cand},
   (b) selecting the "best" candidate c_i out of C_i^{cand},
   (c) selecting a given number of elements out of $pex_i^{c_i}$, thus creating p_i,
   (d) performing `CreateNegEx`(c_i) to produce a given number of (good) negative examples for c_i, which we call the set n_i,
   (e) performing `ReplyQuery`($path(c_i)$,p_i,n_i).
3. $\mathcal{A}g_L$ collects the answers ($path(c_i)$,p_i,n_i) from all agents and uses a learner to learn c_{goal} from these combined examples (action `Learn`((p_1,n_1),...,(p_m,n_m))). If there are conflicts, then it resolves them with the help of the other agents using `AskClassify` (resp. `ClassifyEx` and `ReplyClass` by the other agents, see 3.4).
4. $\mathcal{A}g_L$ uses the learned c_{goal} and the collected $path(c_i)$s from the other agents to construct an ontology path C_{path} leading to c_{goal} (see 3.4) within its ontology $\mathcal{O}_L$ (action `Integrate`($path(c_1)$,...,$path(c_m)$)).

The result of this learning/teaching scheme is the description of c_{goal} in terms of $\mathcal{A}g_L$'s feature set $\mathcal{F}_L$ and an updated ontology $\mathcal{O}_L^{new} = (C_L^{new}, \leq_C, R_L, \sigma_L, \leq_{R_L})$. $\mathcal{A}g_L$ will also create a set $pex_L^{c_{goal}}$ in case another agent wants $\mathcal{A}g_L$ to teach it c_{goal}.

3.2 The initial query

In order to initiate the initial query, $\mathcal{A}g_L$ first needs to become aware that there is a concept that it needs to learn. There are a couple of scenarios that can lead to this realization. $\mathcal{A}g_L$ might observe a conversation between other agents in which an unknown concept identifier is used (or an identifier known by $\mathcal{A}g_L$, but in a way that does not make sense). Or $\mathcal{A}g_L$ might be dealing with a set of objects that share certain feature values from $\mathcal{F}_L$ and it wants to know if other agents know more about the similarities of these objects.

Based on these scenarios, we require action `QueryConcept` to have 3 parameters to be used to define c_{goal} to the other

agents:

`QueryConcept`(identifier,{$[f'_1 = V'_1]$,...,$[f'_l = V'_l]$},O_{goal}).
Here identifier is an element of C_i for some agent(s) $\mathcal{A}g_i$, $\{[f'_1 = V'_1],...,[f'_l = V'_l]\}$ is a selection of features $f'_j \in \mathcal{F}_{base}$ and the values $V'_j \subseteq D_{f'_i}$ that $\mathcal{A}g_L$ thinks are related to the concept c_{goal} and $O_{goal} \subseteq \mathcal{U}$ is a set of objects that $\mathcal{A}g_L$ thinks are covered by c_{goal}. Due to the different scenarios from above, each of the three parameters can be empty, if $\mathcal{A}g_L$ does not have any information on the parameter. Also, $\mathcal{A}g_L$ can decide to address only a subset of $\{\mathcal{A}g_1,...,\mathcal{A}g_m\}$ and then it can use as f'_is also features that are known to this subset and itself. If $\mathcal{A}g_L$ uses the identifier parameter, then there is naturally a chance that different agents use this identifier for different concepts. Again, $\mathcal{A}g_L$ might therefore address only some of the agents. Note that if the addressed agents associate different concepts with the identifier, then $\mathcal{A}g_L$ will end up with learning a subconcept of these concepts.

3.3 Answering the query

Due to our basic assumptions about what our agents are not having in common, a teacher agent $\mathcal{A}g_i$ without any additional knowledge about the ontology $\mathcal{O}_L$ of $\mathcal{A}g_L$ is rather limited in what it can put into an answer to the query by $\mathcal{A}g_L$. In fact, we are not even guaranteed that $\mathcal{A}g_i$ can really grasp what $\mathcal{A}g_L$ wants to know, since $\mathcal{A}g_L$ was already limited in its ways to express what concept c_{goal} it wants to know about. But sets of objects are something that all of our agents can communicate to each other, even if they might perceive these objects differently, and therefore we use mostly sets of objects in the answers that our teachers are creating. The answer of $\mathcal{A}g_i$ will also include information about the concept c_i it thinks $\mathcal{A}g_L$ wants to know about and how this concept is placed in $\mathcal{O}_i$, so that $\mathcal{A}g_L$ can use whatever information pieces it can understand (see Section 3.4).

After receiving the query from $\mathcal{A}g_L$, an $\mathcal{A}g_i$ first has to determine which of the concepts in C_i fits the query the best (i.e. what is c_i). Then it has to select positive and negative examples for this concept and finally it sends this information to $\mathcal{A}g_L$.

3.3.1 Finding the best known concept

The query from $\mathcal{A}g_L$ consists of three parts that it uses to describe what concept it wants to learn more about. Due to the differences between agents, each of these parts can point to different concepts that an agent $\mathcal{A}g_i$ knows of. In fact, if $\mathcal{A}g_L$ provides several objects in O_{goal}, they might be classified by $\mathcal{A}g_i$ into several of its concepts. As a consequence, $\mathcal{A}g_i$ has first to collect all the concepts that fulfill the query into a candidate set C_i^{cand} and then it has to evaluate all these concepts to determine the concept that is, in its opinion, the best fit.

A concept is a candidate with respect to identifier, if its identifier is identical to the identifier in the query. A concept $c = \{[f"_1 = V"_1], ..., [f"_p = V"_p]\}$ is among the candidates due to the feature part of the query, if for all $f"_j$ with $f"_j \in \{f'_1, ..., f'_l\}$ –let us assume that $f"_j = f'_a$– we have that $V"_j \subseteq V'_a$. Finally, a concept is also a candidate, if it covers one of the objects in O_{goal}. Note that the last two conditions put all superconcepts of a concept from the candidate set also in C_i^{cand}.

There are many different ways how an evaluation of the candidates can be performed, especially if we allow for criteria coming from the application area the agents are working in. Each of the 3 query parts can contribute to a measure that defines what is "best", but how these contributions are combined can be realized differently. The identifier part does either produce a contribution or not. Every object in O_{goal} also is binary in its contribution. But this favors the more general concepts, so that some additional criterion is needed that makes the count of covered objects relative to the depth of the concept in the taxonomy tree. Finally, the feature part of the query suggests a good fit of a concept if it agrees with the feature values of many of the features used in the query. But this can also be strengthened by looking at how good the fit for a particular feature is.

We will present an example measure for how a concept fits a query in Section 4. Note that in theory different teacher agents could use different such measures. After the best concept c_i for the query is determined, an agent $\mathcal{A}g_i$ includes into its answer information on the path that leads in its taxonomy to c_i and the subtree below c_i's node (we refer to this information by $path(c_i)$).

3.3.2 Selecting positive and negative examples

Since each agent $\mathcal{A}g_i$ stores for each concept c_j in C_i a set $pex_i^{c_j}$ of positive examples for c_j, i.e. a set of objects covered by c_j, coming up with positive example objects for a concept known to $\mathcal{A}g_i$ is not a big problem. While the more examples normally are the better, in our case we have to take into account that the more objects from $pex_i^{c_i}$ are selected, the more expensive the communication becomes and the more effort $\mathcal{A}g_L$ will have to spent on learning. On the other side, less positive examples usually means a less precise result of the learning. Therefore we suggest to have the number of examples communicated to $\mathcal{A}g_L$ by each agent as a parameter of the whole system. Then selecting the appropriate number of elements for p_i is easily realized by randomly sampling $pex_i^{c_i}$.

Selecting negative examples for a concept is not as easy. Obviously, the set of negative examples nex^c for a concept c is defined as

$nex^c = \mathcal{U} - \{o | o \text{ covered by } c\}$.

This can be a very large set and usually different elements of this set provide learners with a different quality of advice. Good negative examples are examples that "nearly" are in the set covered by the concept, a kind of "near-misses" that allow to highlight the borders of a concept. The fact that our agents have ontologies allows us to do a better job in selecting negative examples than randomly selecting out of nex^{c_i} (by $\mathcal{A}g_i$). The key for this better selection is to make use of the taxonomy information $\mathcal{A}g_i$ has and the relations in R_i. The later naturally depends on what relations are available.

Let us first look at the possibilities that the *taxonomy* offers. Each superconcept of the concept c_i –that $\mathcal{A}g_i$ sees as the best concept to answer $\mathcal{A}g_L$'s query– can be used to limit the set of negative examples $nex_i^{c_i}$ that $\mathcal{A}g_i$ should consider for its answer. As a superconcept of c_i, these concepts share a lot of feature values with c_i, so that the elements in their set of positive examples that are not covered by c_i are good candidates for "near-misses". In fact, sibling concepts of c_i or its superconcepts are an even better source for negative examples since all their positive examples are not covered by c_i. Figure 1 indicates these candidates for the selection of negative examples with the help of taxonomy information

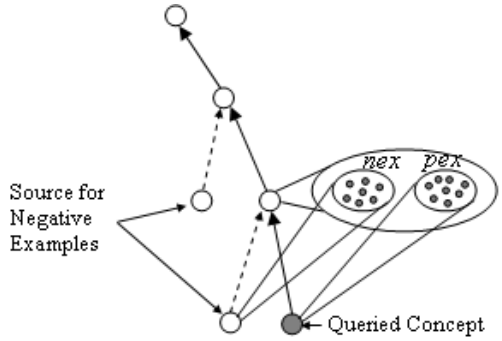


Figure 1: Negative examples using the taxonomy

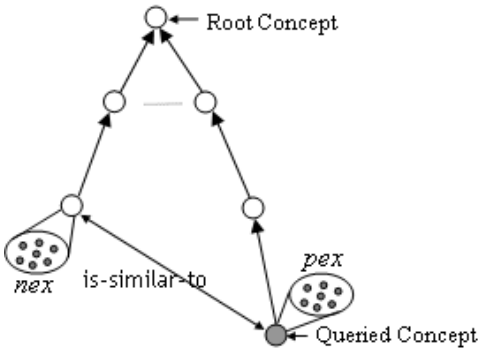


Figure 2: Negative examples using `is-similar-to`

(we use here and in the following the common graphical representation of taxonomies by trees).

Since all agents use the same relation $\leq_C$, all agents can use the taxonomy information to limit the pool of negative examples to choose from. But also information provided by some other relations can be used. As an example, let us look at the usage of the relation `is-similar-to` that we mentioned earlier. The motivation for `is-similar-to` is to allow to express the similarity between two concepts that are far away from each other in the taxonomy tree, but that share a lot of feature values. This makes `is-similar-to` a perfect candidate for helping with the selection of negative examples. Figure 2 shows how we can use concepts that c_i is similar to by using their positive examples as candidates for $nex_i^{c_i}$. After collecting all candidates in $nex_i^{c_i}$, we again select the given number of examples for n_i as a random sample.

Note that an `is-similar-to`-relation can be automatically computed for a given C_i and $\mathcal{F}_i$ by introducing a similarity measure sim_i^f on feature values for each feature $f \in \mathcal{F}_i$ with domain D: $sim_i^f : D \times D \rightarrow [0..1]$. We can create out of this a similarity measure $sim_i^{\mathcal{U}}$ for objects by, for example, summing up the similarities for each feature. More formally, let $o = ([f_1 = v_1], ..., [f_n = v_n])$ and $o' = ([f_1 = v_1'], ..., [f_n = v_n'])$, then

$sim_i^{\mathcal{U}} = \sum_{j=1}^{n} sim_i^{f_j}(v_j, v_j')$

where $sim_i^{f_j}(x, y) = 0$, if $f_j \neq \mathcal{F}_i$.

Out of this, we can create `is-similar-to`$_i$ between two concepts c and c', if $sim_i^{\mathcal{U}}(o, o') \geq simthreshold$ for all $o \in pex_i^c$ and $o' \in pex_i^{o'}$, with $simthreshold$ as a given parameter. While it would be better to use all objects covered by c and c', this can be impossible or at least very expensive, so that we suggest to use the examples that are already there.

3.4 Learning a new concept and path for $\mathcal{O}_L$

Learning a concept, in form of feature values, from a set of positive and negative examples is a problem that is very well researched in literature and there are many algorithms and systems available for this task. Examples are Rocchio (see [6]), k-Nearest-Neighbor (see [5]), and Naive Bayes (see [5]) in our experimental context.

With $pex^{c_{goal}} = \cup_{i=1}^{m} p_i$ and $nex^{c_{goal}} = \cup_{i=1}^{m} n_i$, we have the necessary input for such a learning system, with one potential problem: conflicts between the teacher agents. Due to the differences between agents it can easily happen that the best concepts c_i and c_j that $\mathcal{A}g_i$ and $\mathcal{A}g_j$ identified do not have much in common. The worst case can be that an example that $\mathcal{A}g_i$ sent as being positive for c_{goal} $\mathcal{A}g_j$ sent as a negative one. Obviously, such a contradiction will stop every learner. But we can also have more indirect conflicts where a learning algorithm simply cannot come up with a concept description that covers all objects in $pex^{c_{goal}}$ while not including any objects in $nex^{c_{goal}}$. There are several ways how we can solve this problem and these ways allow us to produce different degrees of willingness to satisfy the teacher agents (by $\mathcal{A}g_L$).

For our system, we have chosen the following conflict resolution. After the learning component of $\mathcal{A}g_L$ has performed `Learn` and produced a more precise c_{goal}, $\mathcal{A}g_L$ will test all elements of $pex^{c_{goal}}$ and $nex^{c_{goal}}$ for correct classification by this new c_{goal}. For all the example objects that are not correctly classified, we get back to the teacher agents and ask them to classify these examples according to the c_i they used to produce their examples. We then treat the answers as votes and have the majority decide how the conflict cases should be classified. Those examples that, according to the vote, are wrongly classified are fed back to the learner with priority. This is repeated until all examples are classified according to the votes. In order to avoid running in cycles we delete examples that keep coming up as misqualified from both example sets (which can be seen as imposing a partial blindness on $\mathcal{A}g_L$ to avoid the problem, but this is definitely something that we can observe in human beings also).

After having created the final description of c_{goal} in terms of $\mathcal{F}_L$, we could use the standard techniques for ontologies to integrate a single new concept into the taxonomy (see [12]). But since the goal of our whole method is to improve the communication between agents, we can use some more of the information provided by the teacher agents to *pre-structure* $\mathcal{A}g_L$'s ontology with concepts that $\mathcal{A}g_L$ will most probably have to learn if it communicates more intensely on the subject of c_{goal}.

The key information used in pre-structuring is the path information send by the teacher agents in their answers to the query. Our pre-structuring uses these paths to create *shells* for concepts that might be useful for future communications. These shells can also be used to indicate to an agent concepts it might want to learn in the future and potential queries for them. For the normal usage of the ontology, we treat shells as non-existent.

Each $path(c_i)$ from an agent $\mathcal{A}g_i$ contains the path in $\mathcal{O}_i$ leading to c_i and the taxonomy tree below c_i in $\mathcal{O}_i$ using the features from $\mathcal{F}_i$ for characterization. We first strip all concepts in $path(c_i)$ that are not in C_{base} of all features

that are not in $\mathcal{F}_{base}$. This creates what we call a concept shell. We then merge shells that are identical in their feature descriptions, both within one $path(c_i)$ and between paths from different agents. Then we combine the shells from all the teachers into one path for $\mathcal{O}_L$ (if there are concept shells that are not comparable with respect to $\leq_C$, then we select the longest path from a concept in C_{base} to c_{goal} that can be formed using the concept shells available; for concept shells smaller than c_{goal} with respect to $\leq_C$ we just create the subtree) and integrate this path into $\mathcal{O}_L$.

4. AN EXAMPLE APPLICATION

Before we present an example for a, in our opinion, very successful application of our ideas from the last section, let us point out that it is relatively easy to construct scenarios in which these ideas may not be very useful. These scenarios have in common that the teacher agents do not only disagree on what concepts fulfill a particular query, their ontologies and their perception of the world, i.e. their $\mathcal{F}_i$'s, are totally different. As a result, their attempts to teach an agent result in total confusion.

If, for example, all teacher agents associate with a particular concept identifier totally different real concepts, then the learning agent will not be able to get useful concepts out of their information. If $\mathcal{F}_{base}$ and C_{base} are empty or nearly empty and $\mathcal{F}_i \cap \mathcal{F}_j = \emptyset$ (or near empty) for all pairs of agents ($\mathcal{A}g_i$,$\mathcal{A}g_j$) (including $\mathcal{A}g_L$), then there is simply no common ground for a successful learning experience. Obviously also a human agent would find it impossible to learn useful information from the teachers in this situation.

With the following application we have chosen an area where the teacher agents have some differences in their "world view", simply because there are different ways how to organize the objects in the world, but where there is nevertheless a large agreement on many things.

4.1 The University Units and Courses Domain

For evaluating our method, we have chosen the course catalog ontology domain (see [2]). The set of objects $\mathcal{U}$ consists of files describing the courses offered by Cornell University, the University of Washington and the University of Michigan. The domain is additionally structured according to the university units of these universities, which creates different ontologies for each of them. In fact, our teacher agents will be agents that each represents one of these 3 universities ($\mathcal{A}g_C$, $\mathcal{A}g_W$, $\mathcal{A}g_M$). The course files (and unit structure) for Cornell and Washington were taken from [2], the ones for Michigan from their web site at [10]. A course file contains course identifier, course description and the prerequisites of a course. Note that this is not a small domain. The three universities together offer 19061 courses and each university's ontology has at least 166 concepts on top of their courses. For each of the following examples, our agents used their full ontologies even if we report only on parts of them.

To represent the courses in terms of features, we had a little bit of preparation to do, borrowing ideas from the field of information retrieval. Obvious features describing a course are its identifier f_{idnt}, a string, and its prerequisites f_{prereq}, a set of course identifiers. Different universities use different systems to create identifiers, so that these features are not really of any help for our purpose. The course description usually determines by which organizational units a course should be taught and the descriptions are text-based.

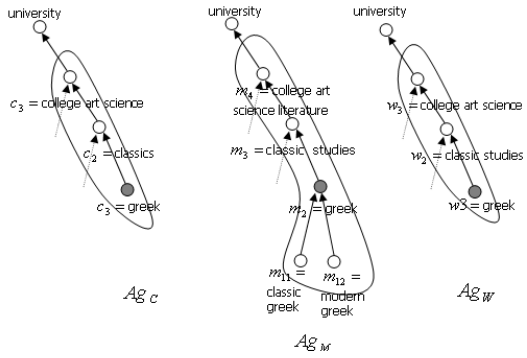


Figure 3: Relevant taxonomy paths of teachers

Defining concepts based on objects that consist of natural language texts is not easy, but an area of quite a lot of interest and practical applications. One way of defining features for such texts to group them is to look for particular words in the texts or word combinations (see [4]). Unfortunately, there is a lot of substitutivity in these word combinations, so that we need features that allow us to express this substitutivity. For example, feature $f_{\texttt{picture,photo,figure}}$: text $\rightarrow$ Boolean is true for a text t, if either `picture` or `photo` or `figure` occurs in t. For our application domain, it is not clear what substitutivities should be considered (just synonyms are not what we are looking for here), so that we base our features for the course descriptions on what we call a set $\mathcal{K}$ of key words. Then we have a feature for each possible subset of $\mathcal{K}$ (excluding the empty set) as described above. Different key sets create different feature sets.

4.2 An example query

To provide a better picture of how our method works, we take a look at a particular query, the relevant parts of the ontologies of the three teacher agents and what an agent can learn from these teachers with regard to this query. Let us assume that the learning agent is supposed to provide someone at a university with suggestions for how a unit concerned with `Greek` should be characterized. This learning agent would pose a query based on providing a key set out of its own key set of words, in our example this query key set would be {`greek, program, attic, literature`} (as stated above, using an identifier does not make much sense here and we are not using example objects for this example due to lack of space). Let us further assume that the relevant concepts in C_{base} are C_{base} = {university} and the relevant concepts in $\mathcal{F}_{base}$ are created using the key set $\mathcal{K}_{base}$ = {`class, course, program, literature, modern, attic, classic, culture, prose, graduate, seminar, grammar, drama, greek`}.

Obviously, different universities can use different key sets of words to express the area a course belongs to, so that this creates additional features for the different agents. In our example, the relevant key sets for the features used in the agents' ontologies are $\mathcal{K}_C$ = {`democritus`}, $\mathcal{K}_W$ = {`tragedy, orator, antique`} and $\mathcal{K}_M$ = {`modern, epic, classic, odyssey, ancient, aristotelian`}. By relevant we mean those key words that really occur in the features that these agents use to characterize the concepts that are related to the query. To make things interesting, we give $\mathcal{A}g_L$ as key set $\mathcal{K}_L$ = {`epic, antique, ancient, tragedy,`

Table 1: Usage of key words in features

key word	$\mathcal{A}g_C$	$\mathcal{A}g_M$	$\mathcal{A}g_W$	a)	b)	c)
`literature`	6	19	6	6	12	15
`greek`	8	55	9	7	26	33
`attic`	3	8	12	3	11	22
`epic`	0	3	0	0	3	3
`antique`	0	0	1	0	0	1
`ancient`	0	6	0	0	4	4
`tragedy`	0	0	3	0	0	3
`orator`	0	0	1	0	0	1

`orator`}.

Figure 3 presents the paths in the taxonomies of the teachers that our system selected as the best concepts for the query. The measure used by the system was calculated as $w_{ident} \times P_{ident}(c) + w_{feat} \times M_{feat}(c) + w_{obj} \times M_{obj}(c)$. w_{ident}, w_{feat} and w_{obj} are weight parameters. In our experiments we used $w_{ident} = w_{feat} = w_{obj} = 1$. $P_{ident}(c)$ is 1, if the concept c's identifier is equal to the identifier in the query and 0 else. The submeasure $M_{obj}(c)$ counts the number of objects from the query covered by c and multiplies it with the length of the path to the concept in the taxonomy. This is then divided by the product of the number of objects in the query and the maximal length of a path in the taxonomy. The submeasure $M_{feat}(c)$ makes use of the key set in the query. For every feature that is true for c, we check if it is formed solely by words from the key set. $M_{feat}(c)$ is the number of these features divided by the number of all features that are true for c.

The set of relevant features for this example is still too big to be easily presented. We used a learner based on Naive Bayes (as described in [4]). This learner can, in fact, use the key sets directly and creates the features implicitely. To provide an idea on how the teachers influence what $\mathcal{A}g_L$ learns, let us look at how many features include some of the key words. As Table 1 shows (a) to c) refer to the concepts learned according to Figure 4, where a) is learned from $\mathcal{A}g_C$, b) from $\mathcal{A}g_C$ and $\mathcal{A}g_M$ and c) from all agents), the difference between the features the different agents know results in rather different numbers for the usage of some key words to make up for words that an agent cannot use in its features. This is especially obvious in the base key set, as the first three examples show.

As the above already suggests, the learned concept c_{goal} is different for different agents used as teachers. For example, with just $\mathcal{A}g_C$ as teacher, among the features enabled in c_{goal} we have $f_{\texttt{attic,prose,greek}}$, which is not enabled by any other agent and also not enabled in the learned concept for the other two scenarios. In the concept learned from $\mathcal{A}g_C$ and $\mathcal{A}g_M$ together, we have the feature using the words `classic, greek, epic, modern, ancient` and `literature` enabled, which is not enabled in the other two scenarios. Finally, learning from all three teachers results in a feature using `greek, tragedy, antique, orator` and `attic` enabled, which is, again, not used in cases a) and b). Remember that an enabled feature means that each course covered by the concept that uses the feature contains at least one word from the set of words associated with the feature!

Figure 4 shows the new taxonomy paths created by prestructuring. The teachers agree on the need for two superconcepts for c_{goal} and $\mathcal{A}g_M$ introduced two subconcepts that are added if $\mathcal{A}g_M$ is one of the teachers. Naturally, we can not associate meaningful names with these concepts

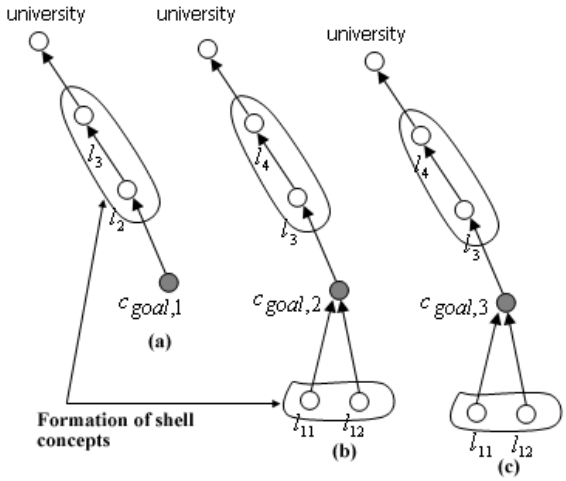


Figure 4: Query results

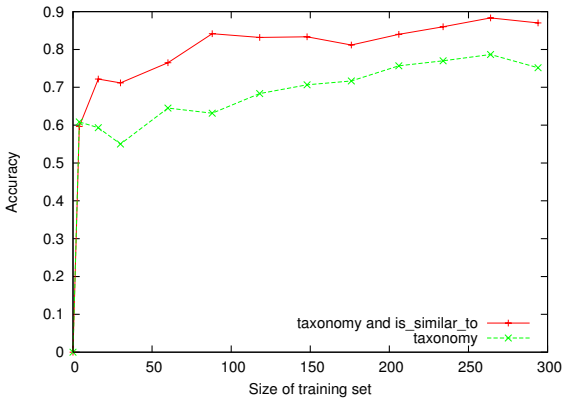


Figure 5: Accuracy results for Mathematics

and, again, there are too many features identifying them to represent them here.

4.3 Using more than the taxonomy

An important question to evaluate our concept is how efficient it is to have an agent learn a new concept. Clearly, the more examples are provided to this agent the better it can learn. But more examples mean more communication overhead, more learning effort and more chances for confusion between the different teachers. We tested several queries in our example domain and present here a typical example. The query is consisting of the key word set {`mathematics, program, arithmetic`} as the feature part and one Mathematics course from each of the three teachers in the example part of the query. Obviously, this query should result in learning the concept "Mathematics Department". Each data point in Figure 5 represents the average value of 5 runs, since the selection of positive and negative examples is performed randomly by the teachers out of the sets they consider (see Section 3.3.2).

Since there is not really a total agreement on what courses constitute Mathematics (beyond a certain strong core, naturally) there is no possibility to achieve 100 percent accuracy. In fact, it can be already debated what accuracy

means in our context, since there is no clearly defined concept $\mathcal{A}g_L$ is supposed to learn. For our experiments, every course classified under Mathematics by one of the 3 teachers was expected to be a positive example, all others a negative one. Naturally, this is not how our approach classifies, but, as Figure 5 shows, we come rather near to that. Figure 5 also shows that using the `is-similar-to`-relation greatly enhances accuracy (especially for small numbers of exchanged examples), due to providing well focused negative examples. $\mathcal{A}g_C$ has in its ontology that Mathematics is similar to theoretical-and-applied-mechanics, computer-science, operations-research-and-industrial-engineering, while $\mathcal{A}g_M$ has it similar to statistics, geological-sciences and astronomy, and $\mathcal{A}g_W$ to statistics, aeronautics-and-astronautics and earth-and-space-sciences. Especially for the range between 50 and 100 training examples, which means 17 to 34 examples per teacher agent, the usage of `is-similar-to` clearly pays off. Note that the "dips" in accuracy are not only due to presenting an average here, more examples also means that more conflicts between teachers occur (remember that using the test set as we do means that a conflict always results in a potential misclassification).

5. RELATED WORK

As already mentioned, most works in literature assume that agents use a common ontology (see, for example, [1]). The works by Williams (see [11]) introduced the idea of using learning to improve the mutual understanding about a concept between two agents. In contrast to our method, Williams uses only a flat repository of concepts, not a real ontology. The learning is used to have only two agents develop a common feature description about a particular concept assuming that the agents share the same perception of objects. No negative examples are used and there is naturally no pre-structuring and no need to deal with conflicts between teachers. [7] presents a method how one agent can train another agent to recognize a concept by providing selected positive training examples. The multi-agent dimension is not addressed and no usage of ontologies is made.

Steels (see [8]), like us, allows for differences in ontologies of agents and wants to minimize these differences for concepts that are of interest to some of his agents. In contrast to us, his agents do not use learning to allow for teaching a concept to an agent, his approach suggests to have the agents cooperatively learn (evolve) a common set of features (using what he calls naming games) and then use a common method for individual agents to create their ontologies by experiencing their environment. For agents with different perceptions, this must pose problems which our approach does not have. But we do not allow for agents to change their feature sets. Also, the emphasis of Steels' work is more on language than concepts.

6. CONCLUSION AND FUTURE WORK

We presented a general concept for improving communication between agents that use ontologies as part of their knowledge representation and that not only have different ontologies but even different features sets that they can recognize. Similar to the behavior of humans, the key idea is to have agents that know a needed concept (but that might not agree totally on all details) teach an interested agent the concept by providing positive and negative examples for the concept that the learning agent feeds into a concept learner. Conflicts are resolved by voting/elimination of examples. We provided an example that illustrates how our method comes up with a concept (and its integration into the agent's ontology) that represents the common ground between the teachers. We also showed in experiments that the use of ontologies allows for a more targeted selection of negative examples resulting in a good understanding of a concept after relatively few exchanged examples.

This work represents a proof-of-concept, not more. We have indicated various parts of our method where alternative realizations of these parts are possible and future work should explore these possibilities. Especially the conflict handling allows for many different ways that could create alternative levels of inclusion of examples. For example, we could have agents vote on every communicated example, not only the clearly problematic ones. Also, selection, resp. filtering, of the available teachers is an interesting future topic. There is also the need for more explorations with the current application, for example by allowing for learning of new features as in [8], and for instantiating our method for other applications.

7. REFERENCES

[1] T. Finin, Y. Labrou, J. Mayfied: KQML as an Agent Communication Language, in J. Bradshaw (Ed.), Software Agents, MIT Press, 1997.

[2] Illinois Semantic Integration Archive. http://anhai.cs.uiuc.edu/archive/, as seen on Jan 30, 2005.

[3] K-C. Jim, C.L. Giles: Talking Helps: Evolving Communicating Agents for the Predator-Prey Pursuit Problem, Artificial Life 6(3), 2000, pp. 237–254.

[4] D. Koller, M. Sahami: Hierarchically Classifying Documents Using Very Few Words, Proc. ICML-97, 1997, pp. 170–178.

[5] T.M. Mitchell: Machine Learning, McGraw-Hill, 1997.

[6] J.J. Rocchio: Relevance feedback in information retrieval, in The SMART Retrieval System - Experiments in Automatic Document Processing, Prentice Hall, 1971, pp. 313–323.

[7] S. Sen, P.P. Kar: Sharing a concept, AAAI Tech Report SS-02-02, Stanford, 2002.

[8] L. Steels: The Origins of Ontologies and Communication Conventions in Multi-Agent Systems, Autonomous Agents and Multi-Agent Systems 1(2), 1998, pp. 169–194.

[9] G. Stumme: Using Ontologies and Formal Concept Analysis for Organizing Business Knowledge, in J. Becker, R. Knackstedt (Eds.): Wissensmanagement mit Referenzmodellen – Konzepte für die Anwendungssystem- und Organisationsgestaltung, Physica, 2002, pp. 163–174.

[10] University of Michigan academic units. http://www.umich.edu/units.html, as seen on Jan 30, 2005.

[11] A.B. Williams: Learning to Share Meaning in a Multi Agent System, Autonomous Agents and Multi Agent Systems 8(2), 2004, pp. 165–193.

[12] A.B. Williams, A. Padmanabhan, M.B. Blake: Local Consensus Ontologies for B2B-Oriented Service Discovery, Proc. AAMAS-03, 2003.

A Context-Aware Approach For Service Selection Using Ontologies *

Murat Şensoy
murat.sensoy@boun.edu.tr

Pınar Yolum
pinar.yolum@boun.edu.tr

Department of Computer Engineering
Boğaziçi University
Bebek, Istanbul,Turkey

ABSTRACT

Selecting the right parties to interact with is a fundamental problem in open and dynamic environments. The problem is exemplified when the number of interacting parties is high and the parties' reasons for selecting others vary. We examine the problem of service selection in an e-commerce setting where consumer agents cooperate to identify service providers that would satisfy their service needs the most. Previous approaches to service selection are based on capturing and exchanging the ratings of consumers to providers. Contrary to previous, rating-based service selection, this paper advocates an objective experience-based approach for service provider selection, in which consumers record their *experiences* with service providers rather than the overall, subjective ratings for a provider. A consumer's experience with a service provider is represented using an ontology that can capture subtle details including the context in which the service was requested. When a service consumer decides to share her experiences with a second service consumer, the receiving consumer evaluates the experience using its own context and evaluation criteria. By sharing experiences rather than ratings, the service consumers can model service providers more accurately and thus can select the service providers for their needs more correctly.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Design, Experimentation

Keywords

Service Selection, Service Ontology

*We thank to the anonymous reviewers for helpful comments. This research is supported by Bogazici University Research Fund under grant BAP05A104.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

Selecting partners to cooperate with is a challenging task in open environments, where the participants are autonomous. Open systems are not operated by a central authority that can monitor all agents' activities and ensure that everyone acts in the best interest of others. This implies that for a given service description, a plethora of service providers with substantially different service offerings will exist. A service consumer that is interested in receiving a particular service should then need to select a subset of the service providers that will satisfy her service needs in the best way.

Previous approaches have studied service selection by considering reputation and trust in agent societies. *Reputation systems* enable consumers to rate the service providers in a centralized location. The ratings of the consumers are then aggregated to decide whether a service provider will act as expected [12]. E-bay [1] uses such a reputation system. Whereas in closed settings a central authority (such as the company itself) exists to collect and aggregate ratings, there is no such authority for open systems. Hence the reputation systems are not directly applicable in open systems [16].

Distributed approaches to service selection consider *trust* among entities [17, 16, 9]. Trust captures a truster's expectation from a trustee for a particular service. Most formalizations of trust lack expressiveness and denote trust merely as a rating. However, the episode that leads to the rating is important for understanding the rationale for the rating [5]. For example, a service consumer may give a low rating to a service provider who delivers a book two days late. If the delivery date is not significant for a second service consumer, the first service consumer's low rating will not be significant, either. Hence, consumers should not record and exchange subjective ratings that depend on context, but objective information on actual experiences that can be reevaluated by the recipient based on context.

Accordingly, this paper develops an approach for distributed service selection that allows consumers to capture their experiences with the service providers using ontologies. The ontology used captures the requested service description and the received service in detail. The consumers can then exchange their detailed experiences of service providers rather than plain ratings. A consumer that receives another agent's particular experience evaluates the received experiences individually, considering her own context to decide on which service provider to select. Whereas rating-based approaches reflect the subjective opinion of the raters, the experience-based approach allows the objective facts of the experience to be communicated to the other party.

The rest of this paper is organized as follows. Section 2 explains representation of experiences using an ontology. Section 3 briefly describes our protocol for capturing and exchanging expe-

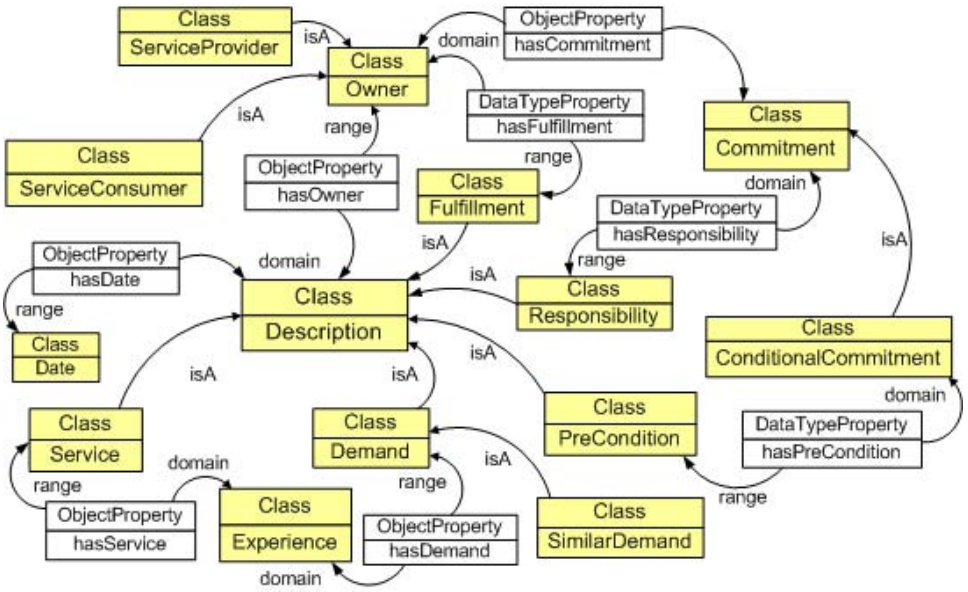


Figure 1: Base level ontology.

riences. Section 4 discusses how service consumers can reason about the experiences of others. Section 5 experimentally evaluates our experience-based service selection with comparisons to rating-based approaches. Section 6 discusses our work with references to the literature.

2. SHARING EXPERIENCES

We consider an architecture where service consumers are looking for service providers to handle their service demands. A *service demand* is expressed in terms of well-defined constraints on attributes of a service such as service completion time, service price, and so on. If a given service does not meet those constraints, then we assume that the owner of the demand be *unsatisfied*. However, another service consumer having weaker demand constraints could potentially be *satisfied*. If consumers only expose their levels of satisfaction (e.g., with a rating), the former service consumer will reveal a low level of satisfaction to the latter consumer. Even though the latter service consumer might have been satisfied with the service provider, she will not choose to interact with the service provider. Instead, if the latter consumer can use the *objective experience* data of the former by knowing the requested and supplied service characteristics, then she can make an informed choice. Consequently, a service consumer can model the service provider using the objective experience data of herself and other service consumers. Example 1 demonstrates how modelling of service providers using actual experiences plays an important role in making right decisions. Model of a service provider may be almost the same for different service consumers but interpretation of the model depends on the specific service consumer, her demand characteristics, and her satisfaction criteria.

EXAMPLE 1. *Consider a book seller who is competent in urgent deliveries to many places except Iowa City. If a service consumer wants an urgent delivery to Iowa City, she can use actual experiences of other consumers to model the service providers and recognize that the service provider could not make timely deliveries to Iowa City and choose another seller instead. Experience ontology should then capture the delivery location to express such nuances in experiences.*

Using objective experience data is promising for the selection of service providers. However, the following challenges need to be addressed:

- How will service consumers express their service demands and experiences?
- How will the experiences of interest be collected by a service consumer?
- How will service consumers use experiences to model service providers and to make decisions?

Service consumers use a common experience ontology for the specified service domain. This ontology covers the fundamental concepts (such as demand, service, and experience), which exist in the base level ontology and domain specific concepts and properties, which exist in the domain level ontology. Using those concepts and properties, a service consumer can express her service demands and experiences.

2.1 Base Level Ontology

The base level ontology (Figure 1) covers domain independent concepts of the experience ontology. Base level ontology is centered around the *Experience* class, whose instances represent experiences of service consumers. This is motivated by the concept of experiences in real life. An experience is a combination of what a consumer request from a service provider and what the consumer receives at the end. Accordingly, in the ontology, an experience consists of a service demand and supplied service for the demand. For this purpose, *Demand* and *Service* classes are included in the base ontology. Both demand and supplied service concepts are descriptions of a service for a specific domain and hence share a number of properties. In order to represent domain specific descriptions, *Description* class is introduced in base level ontology and it is extended in domain level ontology. This class includes generic information to describe basic concepts in the ontology. For example, descriptive classes such as *Demand* and *Service* are sub-classes

of *Description*. Domain-dependent properties of *Description* class can be used to describe service demands, supplied services, responsibilities and fulfillments of sides during transactions. These properties are shown in domain level ontology.

Each *Description* has an owner and a date. *Owner* class is used to represent owners of descriptions. For a demand, owner is a service consumer and for a service, owner is a service provider. *ServiceConsumer* and *ServiceProvider* classes are subclasses of *Owner* class and represent service consumers and service providers, respectively. *Date* class represents the date of description. For an instance of *Service*, *Date* refers to the date of supplied service, whereas, for an instance of *Demand*, it refers to the date on which the demand is created.

An owner may have commitments [15]. Commitments are used to capture the responsibilities of a debtor to a creditor. A commitment always has an instance of responsibility. This means that the owner of the commitment is responsible for the realization of conditions described in the responsibility instance. Example 2 demonstrates a simple responsibility instance. *Commitment* and *Responsibility* classes are used to express commitments and responsibilities respectively in the experience ontology. As a result of transactions, parties have fulfillments. Fulfillments are accomplishments of responsibilities and are denoted with the *Fulfillment* class. *Fulfillment* and *Responsibility* classes are subclasses of *Description*. Owners of responsibilities or fulfillments can be service consumers or providers depending on the context.

EXAMPLE 2. *If a service provider has a commitment and responsibility of the commitment has toLocation property referring to New York City and has hasShipmentCost property referring to 5$, this means that the service provider is responsible for the delivery of the goods to New York City with a shipment cost of 5$.*

Unlike commitments, conditional commitments have preconditions [15]. A conditional commitment $CC(X, Y, P, Q)$ denotes that if the precondition P is carried out by Y, X will be committed to carry out responsibility Q. In this definition, Y is the owner of the precondition and X is the owner of responsibility. *ConditionalCommitment* and *Precondition* classes are used in the ontology to specify conditional commitments and preconditions. *ConditionalCommitment* class is a subclass of *Commitment* and *Precondition* class is subclass of *Description*. Conditional commitments can be used to represent contracts and offers made by service consumers and providers. An example case is demonstrated in Example 3.

EXAMPLE 3. *If the provider makes the shipment one week early, the consumer is committed to pay 1100$ for a product whose actual value is only 1000$. Service providers can also make offers using conditional commitments. The ontology can represent such contracts in demands as well as in offers.*

A service consumer may want to communicate with other service consumers with similar demands. But similarity is a subjective concept and may change for each consumer. To allow a consumer to express her description of a similar demand, *SimilarDemand* class is included in the experience ontology. *SimilarDemand* class is a subclass of *Demand*. A service consumer can express what a similar demand is with respect to her similarity criteria using Semantic Web Rule Language (SWRL) [2]. That is, if a consumer has a particular service demand and a list of others' experiences, then she can apply the SWRL rule to select those experiences in which the service demands were similar to that of her own. If the consumer makes her SWRL rule for similar demands public, other consumers can also use this expression of similarity to reason about whether their past service demands were similar to the demand of the consumer or not.

```
<ruleml:imp>
  <ruleml:_head><swrlx:classAtom>
      <owlx:Class owlx:name="#similarDemand" />
      <ruleml:var>DEMAND</ruleml:var>
    </swrlx:classAtom></ruleml:_head>
  <ruleml:_body>
    <swrlx:datavaluedPropertyAtom swrlx:property="#isRefoundable">
    <ruleml:var>DEMAND</ruleml:var><ruleml:var>Refoundable</ruleml:var>
    </swrlx:datavaluedPropertyAtom>
    <swrlx:datavaluedPropertyAtom swrlx:property="#hasDeliveryDuration">
      <ruleml:var>DEMAND</ruleml:var>
      <ruleml:var>DeliveryDuration</ruleml:var>
    </swrlx:datavaluedPropertyAtom>
    <swrlx:predicateAtom swrlx:predicate="http://kaon2...#ifTrue">
      <owlx:DataValue owlx:datatype="http://...XMLSchema#string">
          $1 && $2 <= 14</owlx:DataValue>
      <ruleml:var>Refoundable</ruleml:var>
      <ruleml:var>DeliveryDuration</ruleml:var>
    </swrlx:predicateAtom>
  </ruleml:_body>
</ruleml:imp>
```

Figure 2: Example KAON2 rule for similar demands

We use KAON2 [3] to perform the mentioned reasoning on experiences. In addition to OWL predicates, KAON2 supports some useful non-OWL predicates such as *ifTrue* and *evaluate* to test and evaluate logical and mathematical expressions. A simple rule for similarity is shown in Figure 2. In this rule, the consumer states that a demand for a service is similar demand only if the demand requires a refundable service in addition to a delivery duration less than or equal to 14 days. Unlike this example, rules for similarity can contain complicated logical statements about demand properties and conditional commitments.

2.2 Domain Level Ontology

Domain level ontology captures domain specific properties and concepts. Core class of domain level ontology is *Description* class. Domain specific properties of *Description* class is used to describe service demands, supplied services, responsibilities and fulfillments of parties during transactions. A domain level ontology for online shopping is shown in Figure 3. Properties of Description class in this ontology are *hasShoppingItem*, *toLocation*, *hasDeliveryType*, *hasDeliveryDuration*, *hasShipmentCost*, and *hasPrice*. These properties refer to shopping items, delivery location, delivery type, delivery duration, shipment cost and price respectively. Some boolean properties are also included in this set of properties: *isRefundable* and *hasConsumerSupport*. These properties indicate whether the transaction is refundable or not and whether the service provider supplies consumer support or not. The range of *hasShoppingItem* property is *ShoppingItem* class. This class has properties, *hasQuantity*, *hasUnitPrice* and *hasQuality*. The range of *hasQuality* property is *Quality* class. This class describes quality properties of shopping items. The properties of *Quality* class are not depicted because of space restrictions.

The properties of *Description* class also have slightly different meanings for different concepts. For example, *hasPrice* property refers to the money a consumer is willing to pay for a service if it is used to describe a service demand. This property refers to the money the consumer is requested to pay for the supplied service if it is used to describe provided service. If *hasPrice* is used to describe a responsibility, *hasPrice* refers to the money a service consumer promises to pay for a service or it refers to the money a service provider will accept for a service, depending on the owner of the responsibility. For the first case, service consumer is the owner of responsibility and for the second case owner is service provider. If

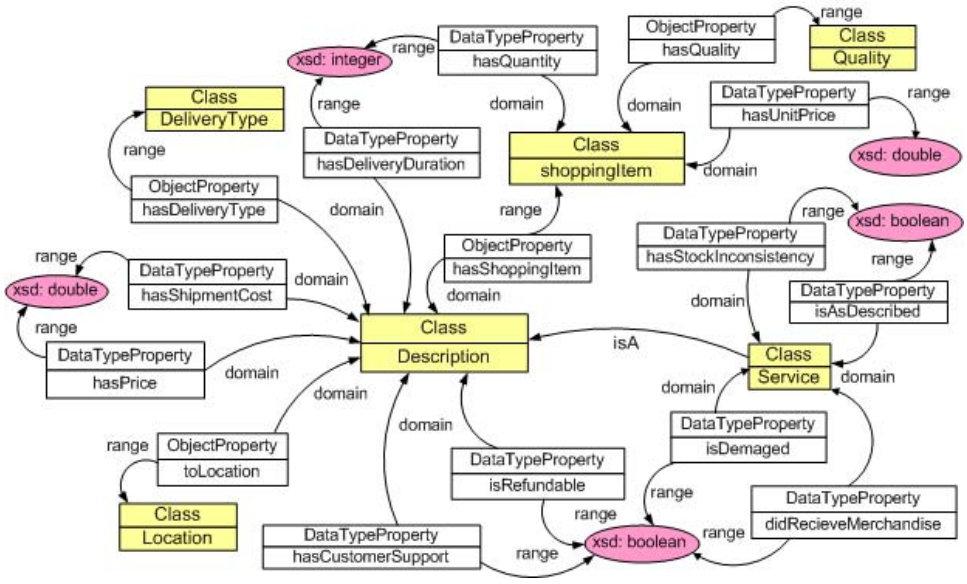


Figure 3: Domain level ontology for online shopping.

hasPrice is used to describe a fulfillment of a service consumer, *hasPrice* refers to the money paid by the service consumer for the specified service.

In addition to the properties of *Description* class, concepts in the ontology may also have domain-specific properties that other concepts do not have. For example, for consumer goods domain, properties such as *didRecieveMerchandise*, *hasStockInconsistency*, *isAsDescribed* and *isDemaged* are included as properties of *Service* class in domain level ontology. The property *didRecieveMerchandise* denotes whether the merchandise is received by the consumer or not. The *hasStockInconsistency* property denotes if the provider claimed to have the product in stock even though it was not. Other properties can be added to this ontology.

3. BUILDING A CONSUMER SOCIETY

A consumer society emerges as a result of consumers' need to retrieve experiences of other consumers. Each service consumer in the society is represented with an agent that is uniquely identified (possibly by an IPv6 address). Each service consumer knows only a subset of all consumers in the society and lists these consumers in an *acquaintance list*. An acquaintance list is a dynamic list of service consumers having service demands classified as similar demand by the owner of the list. When a new service consumer joins the society, its acquaintance list is populated with a small number of randomly chosen service consumers. Acquaintance lists are not symmetric: X may be on Y's list but that does not mean that Y will be on X's list. Each consumer collects others' experiences in an *experience repository*. Each time a service consumer makes a decision, it uses the experiences in this repository. The service consumer refreshes and updates its experience repository periodically by removing old experiences and adding new experiences.

Initially, service consumers do not have any experiences. When a service consumer X needs experiences for reasoning on which service provider to choose, it should discover other service consumers having similar demands and should populate its acquaintance list with those service consumers and their service demands. In real situations, consumers usually change their demand characteristics for the same service domain. Example 4 illustrates such a case. Hence, it may be necessary to collect relevant experiences periodically. In order to accomplish this, a consumer follows the procedure summarized in Algorithm 1 [6].

EXAMPLE 4. *A consumer living in Istanbul orders a best seller for himself and a second copy for his brother living in Iowa City. For the first book, the price is more important than delivery duration. However, the second book is a birthday present and must be delivered on time. For this order, delivery duration is more important than price for the consumer.*

Algorithm 1

```
1:  Decide Shopping
2:  if (Shopping) then
3:     Check Experience Repository
4:     while (Not Have Enough Experience) do
5:        Check Acquaintance List
6:        if (Not Have Enough Acquaintance) then
7:           Get New Acquaintances: Using PDM or RAM
8:        end if
9:        Get Experiences: Using REM
10:    end while
11:    Select Provider
12: end if
```

When a service consumer decides to receive a service (Line 1), it checks its experience repository. In order to make reliable decisions, the service consumer should compute the minimum number of experiences for decision making within a 99% confidence interval [11]. If the number of experiences in the repository is not enough, the service consumer collects new experiences. However, in order to collect new experiences, the consumer should have sufficient number of acquaintances. If it does not have, it should increase the number of its acquaintances (Lines 5, 6).

To increase its acquaintances, a consumer can use two messages: *Peer Discovery Message (PDM)* and *Request for Acquaintances Message (RAM)*. Both PDM and RAM messages contain an SWRL rule that expresses the similar demand criteria of the originator of

the message. When a consumer Y receives a PDM message, it checks if its service demands are similar to that of the originator X. If so, it notifies X and X adds Y as a new acquaintance entry in its acquaintance list. This entry contains identity of Y and its demands classified as similar demand by similarity criteria of X. The consumer Y also forwards the request to a set of service consumers in its acquaintance list. Y selects consumers having demands similar to demand of the originator to forward the request. If there is no such consumer, Y randomly selects consumers from its acquaintance list. How long the request is going to be forwarded is controlled using a time-to-live field. All other agents that receive the request act the same way Y does. When Y receives a RAM message from the originator X, it checks its acquaintance list for entries containing consumers having demands similar to the demand of X. Then Y sends these entries to X and X can add these entries to its acquaintance list.

The service consumer populates it acquaintance list through PDM and RAM messages. After having sufficient number of acquaintances, the consumer uses *Request for Experience Message (REM)* to collect new experiences (Line 9). Interestingly, an REM message also contains a rule for expressing similar demand criteria of the sender. When service consumer Y gets an REM message from service consumer X, it evaluates its demands in its experiences using the similarity criteria in the REM. Later, it can send its experiences to X if the experiences have similar demands with respect to similarity criteria encapsulated in the REM, so that X can populate its experience repository with these experiences. After collecting sufficient number of experiences, X uses experiences in its repository for decision making.

4. MODELLING SERVICE PROVIDERS

When a service consumer decides to receive a service, it models each service provider using the experience data available in its repository and selects a provider with the highest probability to satisfy its needs. In determining a suitable provider, the consumer uses information contained in actual experience data. The consumer can make use of experiences by giving them different weights: for example, it can give more weights to its own experiences than the experiences of others. Demand and service specifications within experiences are received in the form of ontologies, but then they can be converted into internal representation of the service consumer in order to speed up decision making.

For modelling service providers, aggregation of their past service offerings is used. To aggregate the experience instances, agents can use various methods. The method used in the simulations is parametric classification. In this method, the service consumer models each service provider and builds a multidimensional Gaussian model for each service provider using the collected experience data related to that provider. There are two classes for each model: satisfied and dissatisfied.

Demand and commitment information in each experience is represented as a vector. Each field in this vector is extracted from the experience ontology. These fields may correspond to property values in experience ontology such as service price. Then, supplied service for this demand is classified as satisfied or dissatisfied with respect to satisfaction criteria of the consumer. The (vector, class) pairs are used as training set.

For each class, the variance and the mean are extracted from the training set. Those parameters are used for modelling of the class. Then, for each of the classes, a discriminant function is defined to compute the probability of satisfaction [8]. The service consumer performs this computation for every service provider and chooses the provider with the highest satisfaction probability.

5. SIMULATIONS

In order to demonstrate the performance of the proposed methods, we implemented a simulator and conducted simulations on it. The main purpose of these simulations is to see the effect of our model on the selection of an appropriate service provider for a service consumer. In the simulator, two types of service provider selection strategies are implemented and compared with each other in terms of achieved satisfaction. Those strategies are explained below.

- **Service Provider Selection Using Experiences (SPS_{EXP}):** This strategy uses the proposed method for service provider selection. Experiences related to service demand of an agent are collected by the agent and decisions are made using those experiences.

- **Service Provider Selection Using Ratings ($SPS_{ratings}$):** For a new service demand, a service consumer agent can select a service provider using ratings from other consumer agents. In this strategy, in order to make better satisfaction-targeted selections, ratings are taken from those agents who have similar demands with respect to similarity criteria of the agent. So, both SPS_{EXP} and $SPS_{ratings}$ actually use the information from the same service consumers for a given decision process. $SPS_{ratings}$ uses ratings while SPS_{EXP} uses experiences.

To be more focused, the simulations enforce agents to make decisions based on others' experiences rather than their previous experiences. As is the case with real world, service consumers periodically change their service demands. This is done to allow variations on context such as the case shown in Example 1.

5.1 Simulation Environment and Settings

The mentioned simulator is implemented in Java and KAON2 is used as OWL-DL reasoner [3]. Simulations are run on an IBM server with 2.8 GHz CPU and 4.0 GB RAM under Windows 2003 server OS. Simulations are repeated 10 times in order to increase the reliability.

In the simulations, initially there are no experiences accumulated in the agent society yet. As time goes by, the number of experiences increases. During decisions, each agent uses only the experiences of others. Each service consumer changes its demand characteristics after receiving a service with a predefined probability denoted as P_{CD} and collects experiences for its new demand. Each service consumer has a probability of requesting a service for any epoch. This probability is uniformly chosen between 0 and 1. In other words, only around 50% of consumers consume service at a given epoch. An online shopping market with 20 service providers and 400 service consumers are simulated in the simulations. Simulations are run for 100 epochs. Experiences expire after 20 epochs to keep experience repositories fresh and small. Simulations are also conducted for other settings, because of space limitations they are not shown here. Simulation results for other settings are comparable to results shown in Section 5.2.

Note that service consumer X may find the demand of service consumer Y similar to its own, but the service provider that satisfies the demand of Y may not satisfy the demand of X. This could be the case because the satisfaction criteria of Y may be highly different or conflicting with the satisfaction criteria of X. This fact is also imitated in the simulations. That is, a fraction of the service consumers who have the similar service demand to X have a satisfaction criteria that is different from the satisfaction criteria of

X. This fraction is denoted with β. The remaining consumers have satisfaction criteria identical to that of X.

In the simulations, service characteristics of a service provider are generated as follows. Each property of *Service* class represents a dimension in a multidimensional service space. Dimensions of service space used in the simulations are tabulated in Table 1. Each service provider has a multi-dimensional region called service region in this service space. This region is randomly generated. Service space and service regions have 15 dimensions. Service region covers all of the services produced by the service provider.

EXAMPLE 5. *If a consumer located in Istanbul orders two books titled Anagrams from a service provider, then the service that the provider delivers will be constructed as follows: The properties that are specified (shopping item, quantity and location) will be fixed. For the remaining attributes, the service provider will choose random values making sure that the values stay in the range of its service region. So, for this example, the degree of freedom for generating services will be reduced to* 12.

Service description of service providers is enhanced by adding conditional commitments to service regions. Conditional commitments are produced using functions of service space dimensions. Example 6 depicts a production of conditional commitment. Similar conditional commitment functions are generated between other service attributes.

EXAMPLE 6. *If one book is bought, unit price will be 12$ and if four books are bought, unit price will be reduced to 6$. Such conditional commitments are represented using multi-dimensional functions. For the previous example, a non-linear function of quantity and unit price is used, (*$UnitPrice = 8/Quantity + 4$*).*

Demand of a service consumer is generated as the following. Demand space is constructed by removing dimensions of service space, which do not belong to *Demand* class. Then, a region named demand region is chosen randomly. Center of this region represents the demanded service. If provided service for this demand stays within the margins of demand region, the service consumer having this demand gets satisfied, otherwise she gets dissatisfied. If the shipped products are damaged or merchandise is not received, the consumer also gets dissatisfied. Demands are generated so that each demand will be satisfied by at least one service provider.

Similar demand criteria for the service consumer is created as the following. A new region named *similar demand region* is constructed by removing some dimensions of the demand region. Number of dimension to be removed and these dimensions are chosen randomly. Service demands staying within the margins of the *similar demand region* is classified as similar demand by the consumer. In simulations, experience ontology is used to describe experiences and similar demand specifications as explained in Section 2.

Table 1: Dimensions of service space and their ranges

Dimension Name	Type	Range
hasShoppingItem	Integer	1 - 1000
toLocation	Integer	1 - 100
hasDeliveryType	Integer	1 - 6
hasDeliveryDuration	Integer	1 - 60
hasShipmentCost	Double	0 - 250
hasPrice	Double	10 - 11000
hasUnitPrice	Double	1 - 100
hasQuantity	Integer	1 - 100
hasQuality	Integer	1 - 10
isRefundable	Boolean	0 - 1
hasConsumerSupport	Boolean	0 - 1
didRecieveMerchandise	Boolean	0 - 1
hasStockInconsistency	Boolean	0 - 1
isAsDescribed	Boolean	0 - 1
isDemaged	Boolean	0 - 1

5.2 Simulation Results

This section summarizes the results of the simulations. There are two primary parameters in the simulations: P_{CD} and β. Let X be a service consumer making service provider selection, and S be the set of service consumers having demands similar to the demand of X with respect to similarity criteria of X at the time X makes a decision. When X is using $SPS_{ratings}$ for decision making, it takes ratings only from consumers in the set S and X takes experiences from these service consumers while it is using SPS_{EXP} for decision making.

Figure 4 shows the simulation results for P_{CD}=0, β=0.5. This is the case when the consumers do not change their service demands and half of the consumers in set S will have satisfaction criteria conflicting with the satisfaction criteria of X.

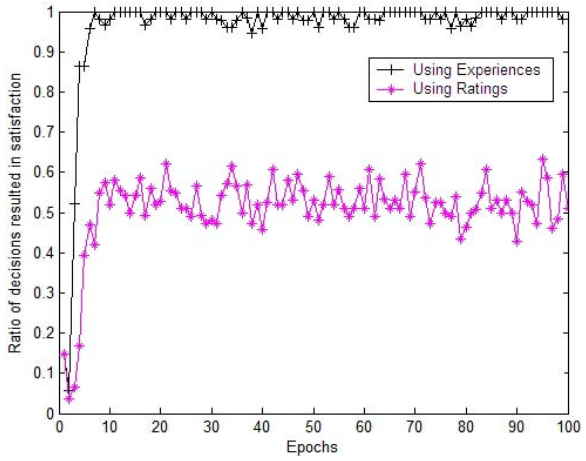


Figure 4: Simulation results for the settings: 20 service providers, 400 service consumers, P_{CD}=0, β=0.5.

When this is the case, only 50% of the services lead to satisfaction of service consumers if these service consumers use $SPS_{ratings}$. On the other hand, service consumers using SPS_{EXP} are almost always satisfied with the supplied service. This is because only half of the ratings taken from the consumers in S lead to decisions that result in satisfaction on the average. However, the other half of the ratings lead to dissatisfaction.

However, as was illustrated in Example 4, consumers change their demand based on circumstances. Figure 5 shows simulation results for parameters P_{CD}=0.2 and β=0. This means that a consumer will change its current service demand with a probability of 0.2 after receiving a service. Figure 5 indicates that the performance of $SPS_{ratings}$ decreases seriously with time, where as the performance of SPS_{EXP} is constant around 100% satisfaction. Since the ratings of a consumer will be aggregation of its past transactions for all of its previous demands, as the consumers change their demands, their ratings will be more misleading than before. However, SPS_{EXP} uses actual experience data, which is free of any evaluation and uses this information for building models of service providers. Hence, as seen in the Figure 5, SPS_{EXP} leads to decisions with 100% satisfaction but satisfaction ratio of

$SPS_{ratings}$ decreases as consumers change their demands.

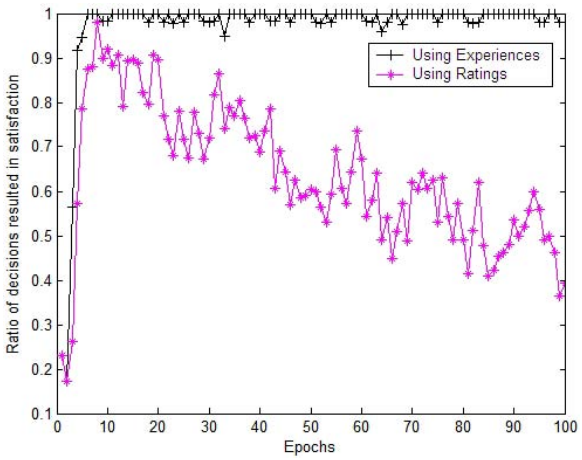


Figure 5: Simulation results for the settings: 20 service providers, 400 service consumers, P_{CD}=0.2, β=0.

Figure 6 shows simulation results for parameters P_{CD}=0.2 and β=0.5. This shows that the consumers will change their service demand with a probability of 0.2 and half of the consumers with the same demand will have conflicting satisfaction criteria. Performance of $SPS_{ratings}$ decreases further for these settings. However, for SPS_{EXP}, satisfaction ratio is around 1 after 5th epoch (before 5th epoch, there are not enough experiences accumulated in the environment for the modelling of service providers). This means that proposed method almost always leads to satisfaction of consumers. In order to see influences of β and P_{CD} on the satisfaction ratios achieved by SPS_{EXP} and $SPS_{ratings}$ strategies, simulations are repeated for different β and P_{CD} values.

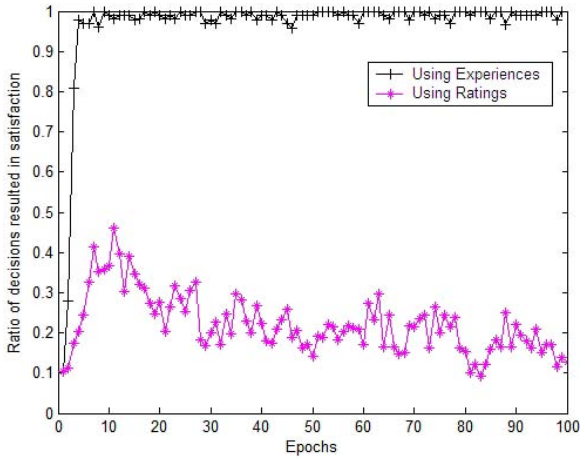


Figure 6: Simulation results for the settings: 20 service providers, 400 service consumers, P_{CD}=0.2, β=0.5.

Average ratio of satisfactions for these simulations are shown in Table 2 and Table 3. Performance of $SPS_{ratings}$ decreases considerably with an increase in the value of β or P_{CD}. These parameters are independent of each other. So, combinatorial effect of these parameters on the performance of $SPS_{ratings}$ is the multiplication of influences of each parameter. Simulations show that proposed method, SPS_{EXP} is robust to changes in β and P_{CD} parameters.

Unlike $SPS_{ratings}$, performance of SPS_{EXP} does not change with changing β and P_{CD} values as well as the achieved satisfaction is 100% if service consumers use SPS_{EXP} for decision making.

Table 2: Average ratio of satisfaction with respect to different β values (P_{CD} is set to 0).

β	0	0.1	0.2	0.3	0.4	0.5
SPS_{EXP}	0.97	0.97	0.975	0.98	0.98	0.98
$SPS_{ratings}$	0.95	0.85	0.76	0.68	0.61	0.53

Table 3: Average ratio of satisfaction with respect to different β and P_{CD} values.

P_{CD}	SPS_{EXP}		$SPS_{ratings}$	
	$\beta = 0$	$\beta = 0.5$	$\beta = 0$	$\beta = 0.5$
0.0	0.97	0.97	0.95	0.53
0.1	0.97	0.96	0.72	0.35
0.2	0.97	0.97	0.56	0.28
0.4	0.98	0.98	0.42	0.18
0.6	0.98	0.97	0.37	0.15
0.8	0.97	0.98	0.32	0.13
1.0	0.98	0.98	0.28	0.12

For the setting β=0 and P_{CD}=0, satisfaction ratio of the strategy $SPS_{ratings}$ approaches 1. However, this setting is far from reflecting the real world. If a consumer classifies the demand of another consumer as similar demand, any provider satisfying the latter will also satisfy the former in case of β=0. This means that satisfaction criteria of the latter is identical to satisfaction criteria of the former. Therefore, ratings will be adequate for the right selection.

6. DISCUSSION

Previous approaches for service provider selection are mostly based on ratings that are inherently subjective. Instead, we are proposing to represent the steps that result in the rating, such as the requested service, the received service, and so on explicitly. This enables the collected information to be reinterpreted based on context. The performed simulations show that the use of experiences improves the decisions of the service consumers and increase the overall satisfaction significantly compared with the rating-based service selection strategy.

Current service provider selection strategies accept ratings as first-class citizens, but do not allow more expressive representations like we have here. Whereas rating-based approaches assume that the ratings are given and taken in similar contexts (e.g., in response to similar service demand), we can make the context explicit. This allows agents to evaluate others' experiences based on their needs. Thus, the use of experiences improves the satisfaction ratio of the consumers.

Different AI techniques, such as case-based reasoning (CBR) can also be applied on objective experience data to select service providers. In [7], we propose a CBR approach for service selection and compare it with the parametric classification. In the CBR approach, a service consumer collects the most similar experiences from other service consumers. Then, the consumer evaluates the supplied services within these experiences using its satisfaction criteria and selects the service provider which provides the most satisfactory service. When service providers do not change the quality of their services, both reasoning techniques perform equally well

in finding service providers. However, CBR approach finds the service providers in a shorter time than the parametric classification. On the other hand, if the service providers vary their service offerings even a small percentage, then the CBR approach performs much worse than the parametric classification.

FIRE [9] is a trust and reputation model consisting of four components: interaction trust, role-based trust, witness reputation and certified reputation. Witness reputation component is directly related to our approach since it allows agents to locate others by making use of other agents' past experiences. However, in FIRE the past experiences are captured only as ratings. However, in our approach, agents exchange their experiences in the form of ontologies so that they can represent their demands, received services, and so on in more depth.

Sen and Sajja [13] develop a reputation-based trust model that is used for selecting processor agents for processor tasks. Each processor agent can vary its performance over time. Agents are looking for processor agents to send their tasks to using only evidence from others. Sen and Sajja propose a probabilistic algorithm to guarantee finding a trustworthy processor. In our framework, service demands among agents are not equivalent; hence a provider that is trustworthy for a consumer need not be so for a different consumer.

Yolum and Singh study properties of referral networks for service selection, where referrals are used among the service consumers to locate the service providers [16]. Current applications of referral networks also rely on exchanging ratings. Hence, suffer from circulation of subjective information. It would be interesting to combine referral networks with the ontology representation here so that agents can exploit the power of ontologies for knowledge representation as well as referrals for accurate routing.

Zhang *et. al.* propose a multiagent approach for distributed information retrieval task [18]. In their work, each agent has a view of its environment called agent-view. The agent-view structure of an agent contains information about the language models of documents owned by each agent. An agent-view reorganization algorithm is run to dynamically reorganize the underlying agent-view topology. An agent-view is analogous to acquaintance list structure in our work. Zhang *et al.*'s protocol does not use ontologies or description logic reasoners during information retrieval. However, if their protocol is modified to accommodate the experience ontology and DL reasoners, their protocol can be used for retrieving experiences instead of the protocol that we have proposed in Section 3.

Soh and Chen propose a multiagent approach to improve distributed information retrieval performance by balancing ontological and operational factors [14]. In this work, collaborating agents enhance their performance by learning ontologically and operationally. Soh and Chen show that their proposed approach is able to improve the quality of the collaborations in terms of the response time, quality of the retrieved results, the number of neighbors contacted and message complexity. A similar approach can also be applied to our work in order to improve quality of collaboration.

Maximilien and Singh develop a QoS ontology to represent the quality levels of service agents and the preferences of the consumers [10]. Their representation of QoS attributes is richer (such as availability, capacity, and so on), however, their ontology does not represent commitments and thus business contract as part of the ontology. Further, their system does not allow reasoning by agents individually as we have developed here.

As future work, we plan to study the role of contracts in service selection. Further, we plan to extend our ontology using OWL-S [4] to make our approach applicable for Web services.

7. REFERENCES

[1] E-bay web site: http://www.ebay.com, 1995.

[2] SWRL: A Semantic Web Rule Language Combining OWL and RuleML, 2004.

[3] Kaon2 web site: http://kaon2.semanticweb.org, 2005.

[4] Owl-s: Semantic markup for web services, 2005.

[5] C. Castelfranchi and R. Falcone. Principles of trust for MAS: cognitive anatomy,social importance, and quantification. In *Proceedings of the 3rd International Conference on MultiAgent Systems*, pages 72–79, 1998.

[6] M. Şensoy, F. C. Pembe, H. Zırtıloğlu, P. Yolum, and A. Bener. Experience-Based Service Provider Selection in Agent Mediated E-Commerce. *To appear in Engineering Applications of Artificial Intelligence*, 2006. Available at: www.cmpe.boun.edu.tr/˜pyolum/eaai-2005.pdf.

[7] M. Şensoy and P. Yolum. A comparative study of reasoning techniques for service selection. Technical Report FBE(CmpE01)2006-2, Boğaziçi University, 2006. Available at: www.cmpe.boun.edu.tr/˜pyolum/SensoyYolum-TechReport.pdf.

[8] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, West Sussex, 2001.

[9] T. D. Huynh, N. R. Jennings, and N. Shadbolt. Fire: An integrated trust and reputation model for open multi-agent systems. In *Proceedings of 16th European Conference on Artificial Intelligence*, pages 18–22, 2004.

[10] M. Maximilien and M. P. Singh. A framework and ontology for dynamic web services selection. *IEEE Internet Computing*, 8(5):84–93, 2004.

[11] D. C. Montgomery. *Design and analysis of experiments*. John Wiley and Sons, West Sussex, 2001.

[12] J. Sabater and C. Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 475–482, 2002.

[13] S. Sen and N. Sajja. Robustness of reputation-based trust: Boolean case. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 288–293, 2002.

[14] L.-K. Soh and C. Chen. Balancing ontological and operational factors in refining multiagent neighborhoods. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 745–752, 2005.

[15] P. Yolum and M. P. Singh. Flexible protocol specification and execution: Applying event calculus planning using commitments. In *Proceedings of the First International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 527–534, 2002.

[16] P. Yolum and M. P. Singh. Engineering self-organizing referral networks for trustworthy service selection. *IEEE Transactions on Systems, Man, and Cybernetics*, A35(3):396–407, 2005.

[17] B. Yu and M. P. Singh. Emergence of agent-based referral networks. In *Proceedings of the First International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1268–1269, 2002.

[18] H. Zhang, W. B. Croft, B. Levine, and V. Lesser. A multi-agent approach for peer-to-peer based information retrieval system. In *Proceedings of the Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 456–463, 2004.

TANDEM - a Design Method for Integrating Web Services into Multi-Agent Systems

Steve Goschnick | Sandrine Balbo
University of Melbourne
Interaction Design Group
+61 3 8344 1522 | 83441497
stevenbg|sandrine@unimelb.edu.au

Leon Sterling
University of Melbourne
Intelligent Agent Lab
111 Barry St, Carlton, 3010
leon@cs.mu.oz.au

Christine Sun
Independent Scholar
P.O. Box 218, Belgrave,
Australia, VIC, 3160
csun@taiwan.com.au

ABSTRACT
This paper introduces a new design method for multi-agent systems (MAS) that incorporate logic programming. The DigitalFriend is an example of a MAS with a built-in logic language interpreter, which is aimed at having the end-user as developer. It uses small fragments of logic as dynamic *glue*, bringing together numerous sub-agents that may exist as Web services. However, application developers need to devise well-formed predicates and logic rules - and there lies the need for a specific *design method*. The well-established techniques of Task Analysis and entity relation Normalization are drawn together into our new design method (TANDEM), introduced here with an example application - the *movie-cinema problem*.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *Intelligent agents, Languages and structures, Multiagent systems.*

General Terms
Design, Languages, Human Factors.

Keywords
Multi-Agent System, Web Services Orchestration, End-user Systems, Logic Programming Methodology.

1. INTRODUCTION
This paper introduces a new *design method* suitable for developers of multi-agent systems that incorporate a *logic language,* such as the DigitalFriend [3], that wrap web services and other agents. The method draws upon *T*ask *A*nalysis [1] and data *N*ormalization from entity relation (ER) modeling [2], together into a new *DE*sign *M*ethod named *TANDEM*. - for devising the logic predicates and rules, which are distributed amongst the sub-agents. The method is demonstrated in the design of a *movie-cinema* locating application, typical of a modern application scenario that gathers constituent components from distributed web services. The method has broader applicability for logic programming in general.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

2. THE *TANDEM* DESIGN METHOD
The analysis begins with a collection of possible *entities* uncovered by a requirements gathering exercise. In the working system, information representing these entities are often dynamically sourced from a number of unrelated distributed Web services. Nonetheless, ER modeling proceeds as if it were going to be implemented as a singular system.

2.1 The Movie-Cinema Problem
We present the methodology via an example application, one typical of projects developed by software engineering students at the University of Melbourne, centered on industry needs. Such projects often involve Internet resources and complex problems, which can benefit from recent research and development advances in software agents and MAS.

The problem: The client, a movie cinema operator, wants a system accessible by movie-going customers via the Internet, one which will allow users of smart hand-held devices, to search for specific movie details, ultimately leading to the booking of cinema tickets. Space prevents a full problem specification here of the *movie-cinema problem.*

2.2 Phase 1 - Applying Task Analysis
Task Analysis identifies a user's goals and the procedure or steps needed to achieve them. It includes the process of analyzing the structure of a task and decomposed the task into related sub-tasks. Some of the tasks identified for the *movie-cinema problem* were extracted using a task analysis and are listed below as *verb-first* directives:

Task 1 - *Movie-oriented search*:

- *Find* all recently released movies.
- *Find* all movies within a particular genre.
- *Find* which movies feature a specific actor.
- *Find* all movies by a specific director.

Task 2 - *Cinema-oriented search* (has a similarly sized list)

Task 3 - *Location-oriented search* (ditto)

Task 4 - *Seat-oriented search and ticket allocation* (ditto)

These four groups of tasks and sub-tasks - represent the first step in a *hierarchical task modeling* exercise [1], which is all we take from task analysis.

2.3 Phase 2 - Applying ER Normalization

In database applications a technique for designing new systems that is widely used and easy to teach, and is taught in both Information Systems and Computer Science, is entity relation (ER) modeling [2].

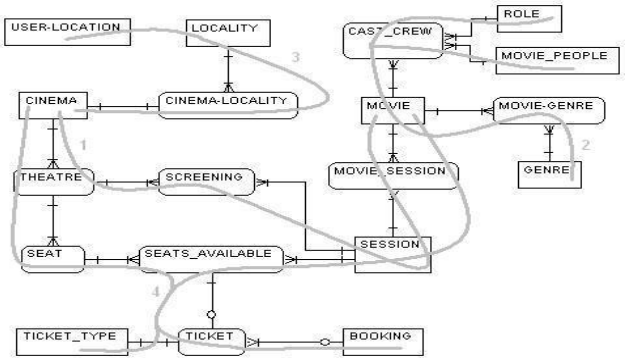


Fig.1 **– Normalized Entity Relation Diagram Modeling the Cinema Problem, with four *Logic Flow Lines***

And a technique that is applied to an ER model to get it into an *ideal form* - one that is most flexible with regard to queries put to it and to future change requirements - is called *Normalization*.

We adopt this well-known technique as a step within the TANDEM methodology - and since it is well-known, the process of normalization is not outlined here. We applied it to the movie-cinema problem above, and arrived at the normalized entity relation (ER) model in Figure 1 above.

Then, in order to divide the ER diagram into appropriate sub-areas, to be implemented as sub-agents (often powered by external web services), we identify *logic flow lines*, also depicted in Figure 1. These lines are mapped by taking the four high level tasks identified in the Task Analysis, and identifying which of the entities in the ER diagram are involved in each task - *without breaking the line*. – a condition that facilitates the process of *unification* [6] during the execution of the logic language.

2.4 Phase 3 - Create well-formed Logic Rules

In a logic program the predicate at the *head-of-the-rule* is a template for many of the queries that can be answered by the whole program. In the MAS application these rules and the associated logic is distributed between the various agents.

In our example solution five rules were derived, with the following five *head-of-the-rule* predicates, the first of which is explained in the next sub-section:

cinemaAssistant(CinemaName, Location, TheatreType, MovieTitle, GenreName, Rating, Date, StartTime)

movieAssistant(MovieTitle, Rating, GenreName, RunningTime, DvdRelease, RoleName, FirstName, Surname)

localityAssistant(Distance, Radius, Location, TownSuburb, Postcode, CinemaName)

seatSelection(CinemaName, Location, TheatreID, ImageMap, MovieTitle, Date, StartTime, RowNumber, SeatNumber, CoordX, CoordY, allocated)

ticketAssistant(CinemaID, TheatreID, Date, StartTime, RowNumber, SeatNumber, TicketNumber, TicketType, Cost, BookingID, FirstName, LastName, PaymentMethod, Paid)

A *query* is constructed by replacing one or more of the variables (i.e. those *terms* starting with a capital) with a specific literal instance, for example, the query in English:

What movies feature Sharon Stone as a star?

would be specified in a logic language (such as CoLoG within the DigitalFriend) as follows:

movieAssistant(***MovieTitle, Rating, GenreName, RunningTime, DvdRelease,*** *star, sharon, stone*)**?**

It can also be specified with under-scores in place of the variable names, leaving only the need to supply the literals:

movieAssistant(_ , _ , _ , _ , _ , *star, sharon, stone*)**?**

In the TANDEM methodology, we first decide on all the head-of-rule predicates that will be necessary to answer all of the tasks extracted via the Task Analysis. The four groupings of *primary tasks*, told us that we needed four predicates as outlined above (the fifth is via a branch in a logic flow line).

2.4.1 Deriving the Cinema Assistant Rule

To identify the actual *terms* needed by each predicate, we use a more detailed diagram than the ER diagram depicted in Figure 1. We could have chosen to depict all sub-area entity *attributes* in a more detailed style of ER diagram, however, in this paper, we have chosen to depict attributes for each of the sub-areas identified by a *logic flow line* in Figure 1, as UML Class diagrams – as we assumed that more agent application developers are likely to be familiar with the UML notation.

The constituent *terms* within each predicate are then determined by the atomic tasks that made up the high level task for which the predicate was instigated.

The next stage of this phase is to identify the best set of predicates on the *right-hand side* of each rule:

Head of the **Rule 1**:

cinemaAssistant(*CinemaName, Location, TheatreType, MovieTitle, GenreName, Rating, Date, StartTime*) ←

requires the following Predicates (from the normalized ER diagram) on the right hand side of the *implies* operator (←):

cinema(*CinemaID,* ***CinemaName***, ***Location***, *CinemaLatitude, CinemaLongitude*),

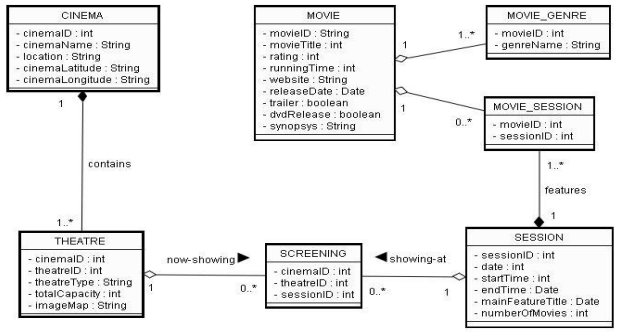


Fig.2 **– A UML Class diagram detailing attributes for entities associated with the *cinemaAssistant* rule.**

theatre(*CinemaID, TheatreID,* ***TheatreType****, TotalCapacity*),

session(*SessionID,* ***Date****,* ***StartTime****, EndTime, MainFeatureTitle, NumberOfMovies*),

movie(*MovieID,* ***MovieTitle****,* ***Rating****, RunningTime, Website, ReleaseDate, Trailer, DvdRelease, Synopsys*),

movieGenre(*MovieID,* ***GenreName***),

screening(*CinemaID, TheatreID, sessionID*),

movieSession(*MovieID, SessionID*);

Note: Commas between predicates are logical ANDs.

These *required* predicates making up the *body* of the rule, are identified via their member terms within the entity/class. i.e. Entities in Fig.2 with an *attribute* corresponding with a *term* in the head *predicate,* is included as a *rule body* predicate. Such attributes are highlighted in ***bold-italic*** font above.

3. IMPLEMENTING THE DESIGN

The DigitalFriend [3] - a user-friendly MAS - is the tool we used to *implement* the design. It is especially good at incorporating sub-agents that *wrap a web service* [4].

A tree view of the various agents is displayed via an octagonal-tile interface, recursively deep, but which displays three levels at a time. An insert in Figure 3 shows where *MovieAssistant*, *CinemaAssistant* and *LocalityAssistant* are grouped together under the EOC (envelope-of-capability) agent *MoviesCinemas.*

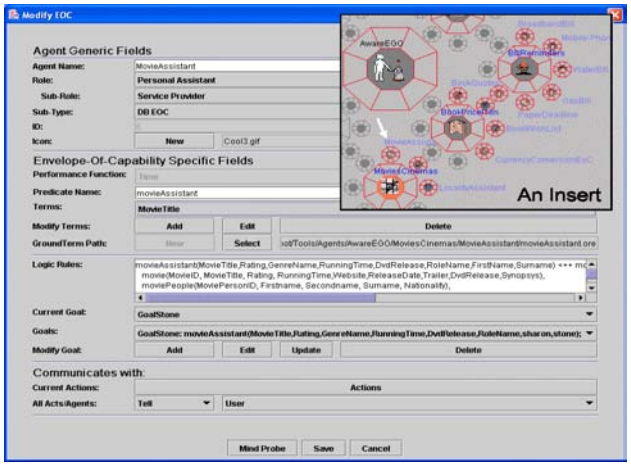


Fig.3 **– Modification of EoC Agent properties**

Both EOC agents and non-EOC sub-agents encompass predicates, which are created within the DigitalFriend interface via GUI dialog boxes. E.g. Figure 3 represents the property dialog box for an EOC agent – specifically for MovieAssistant.

There are four main sub-areas of input/edit fields in the dialog: *role and identity* fields; *predicate* and *term* related fields and buttons for interaction; logic *rules* and a series of predefined *goals/queries* for the agent; and an *inter-agent communication language* section (beyond the scope of this paper). The rule for *movieAssistant* – the second rule, as designed according to Phase 3 outlined in Section 2.4 – has been entered into the field *Logic Rules*. The goal/query that will return any movies involving the actress *Sharon Stone* as *star*, is the agent's *current goal*.

Sub-agents that retrieve records/ground-terms from web-services are similarly represented by a predicate with a list of terms – however, rather than rules, they also have *timers* set to retrieve new information at regular intervals set by the user, and they update stores of *ground-terms* that they maintain as agent *state*.

The DigitalFriend dynamically collects all rules, predicates and ground terms for an EOC agent and its sub-agents, and effectively combines them into a temporary logic program in the internal CoLoG language – built and executed when triggered [3]. Many of the resultant terms are passed to other agents – as specified via the inter-agent communication language in CoLoG – and many are passed as *messages* for the user's attention.

The CoLoG program which is dynamically constructed for a given EOC agent, can be inspected and executed (and tested and debugged) by the user from within the GUI, via the '*Mind Probe*' button at the bottom of the property dialog seen in Figure 3.

The only CoLoG code that the end-user had to devise is the *Rule*, entered via the 'Logic Rule' field in the property dialog in Figure 3. All other details have been entered via less textual GUI interaction, involving only buttons and single name terms.

4. DISCUSSION

Several current MAS development environments use internal logic languages, which then obligates the application developer to write well-formed logic predicates and rules. There is currently *a gap* in available methods for writing such predicates and rules [5].

ER modeling is more widely known than logic programming, and more teachable. Basic Task Analysis is straight forward. TANDEM combines these well-known techniques to find appropriate predicates, their constituent terms and the rules that bind them. It fills a gap in the methods for logic programming.

As seen via the *Movie-Cinema problem*, there is a need for such design methods in developing solutions that use such MAS. Compounding this need, the DigitalFriend is aimed at the *end-user*/developer market, where technical skills are at the spreadsheet-developer or the desktop-database-designer level. Our TANDEM design method provides *strong guidance* in finding appropriate predicates as early as possible, using well-tested techniques that can be learned in a short timeframe.

5. REFERENCES

[1] Annet, J. Hierarchical Task Analysis. In *The Handbook of Task Analysis for Human-Computer Interaction*, Diaper, D. and Stanton, N. (Eds), Lawrence Erlbaum, 2004, 67-82.

[2] Benyon, D. *Information and Data Modelling, Second Edition,* McGraw-Hill, England, 1997.

[3] Goschnick, S.B. & Graham, C. Augmenting Interaction and Cognition using Agent Architectures and Technology Inspired by Psychology and Social Worlds. To appear in: *Universal Access in the Information Society*, 4(3), 2005.

[4] Goschnick, S.B. and Sterling, L. Enacting and Interacting with an Agent-based Digital Self in a 24x7 Web Services World. In *Proceedings, Web Intelligence and Intelligent Agent Technology (WI/IAT)*, Halifax, Canada, 2003.

[5] Sterling, L. Patterns for Prolog Programming, In *Proceedings, Computational Logic: Logic Programming and Beyond 2002*: 374-401.

[6] Sterling, L. and Shapiro, E. *The Art of Prolog*, 2nd edition, The MIT Press, Massachusetts, 1994.

Reconciling Ontological Differences by Assistant Agents

Kendall Lister
The University of Melbourne
Department of Computer Science and Software Engineering
Victoria, 3010, Australia
+61 3 8344 1418
krl@cs.mu.oz.au

Leon Sterling
The University of Melbourne
Department of Computer Science and Software Engineering
Victoria, 3010, Australia
+61 3 8344 1404
leon@cs.mu.oz.au

Kuldar Taveter
The University of Melbourne
Department of Computer Science and Software Engineering
Victoria, 3010, Australia
+61 3 8344 1403
kuldar@cs.mu.oz.au

ABSTRACT
In this paper, we propose an alternative solution for approaching the problem of semantic interoperability, based on recent projects to develop software agents and systems that attempt to reconcile ontological differences without explicit formal ontologies. An application capable of reconciling ontologies based on example data is first described. The results from one of the case studies where the proposed solution has been applied are reported. Finally, the design of an agent embedding the solution is sketched. The solution is aimed at the creation of assistant agents helping the users to cope with heterogeneity in both information sources and contexts of their usage.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *coherence and coordination, intelligent agents*; D.2.12 [**Software Engineering**]: Interoperability – *data mapping*

General Terms
Experimentation

Keywords
Ontology, agent, ontology reconciliation, Web service, context, information retrieval

1. INTRODUCTION
It can be envisioned that in the near future the users of the Internet will make more use of different services, which are commonly termed as Web services [1], instead of just passive browsing. Increasingly important is the delivery of customised Web services which return information based on the profile and context information, for example the location and connection mode, of the user. With the growing diversity of access terminals other than computers, such as mobile phones and portable digital assistants (PDAs), mobile information access and remote transactions are becoming commonplace. Already today, wireless carriers offer services that allow information such as weather, stock quotes, news, traffic situation and sports updates to be accessed by a mobile phone. Mobile phones are also capable of proactive invocation of useful Web services depending on the user's context. For example, business or leisure travellers by car or train may interact with different systems in order to find and utilize context-based information or services offered by various content and service providers.

The data returned by a Web service is structured by using tags (labels) of the Extensible Markup Language (XML) [2]. An information source represented by a Web service can been seen as having an implicit view of the world – *ontology*[1] – that is defined by the tags it uses to present its data and the structure and representation it gives to that data. In a similar manner, the user can be seen as having his/her own ontology which is the user's view of the world. However, the ontologies of information sources are usually different from each other and from users' ontologies. Further, tags used for data structuring may be cryptic or represented in a language that the user doesn't master. Therefore we propose to interpret the data returned by a Web service by mapping the data elements of an unfamiliar information source to the data elements of another information source the tags of which the user understands. This can be done by means of a software agent, commonly understood as an *active* entity, possessing the features of *autonomy*, *proactiveness*, *responsiveness*, and *social behaviour* [4]. A user's access terminal would then be equipped with such an agent, termed an *assistant agent*, which helps the user to make sense of the information gathered from the Internet.

2. AReXS
AReXS (Automatic Reconciliation of XML Structures) is an application built at the Intelligent Agent Lab of the University of Melbourne that reconciles heterogenous information sources by aligning them according to their implicit ontological structure. It is able to reconcile differences of expression and representation across data streams structured by means of XML tags from heterogeneous sources without any predefined knowledge or human intervention [5, 6]. It achieves this by identifying XML elements whose meanings are similar enough to be considered equivalent. AReXS requires no knowledge or experience of the domain in which it works, and indeed is completely domain independent. It uses Example-Based Frame Matching (EBFM) [7] and is able to achieve very high recall with modest precision on real world data collected from commercial Web services and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

[1] A *problem-oriented ontology* is a description by truth values of the concepts and relationships of the problem domain [3].

sites. By requiring no domain knowledge, AReXS is suitable for application to any field; its success relies on its ability to identify and resolve the differences in representation that result from sourcing data from a multi-cultural environment.

AReXS does not fit into the traditional mould of an ontology management tool, as it does not use any type of formal ontology representation. Instead, it works in much the same way that two people who do not share a common language might teach each other by pointing at objects and saying the names that each person's language gives to that object. In formal terms, AReXS attempts to automatically resolve the problems of synonymy and polysemy that are significant hurdles to semantic interoperability [6, 8].

As a result of experiments conducted at the Intelligent Agent Lab, we modified the original EBFM algorithm described in [7] in two important ways. Whereas, in its original form, the algorithm would discard hypotheses that are based on a single element pair, we deliberately retain these hypotheses. Further, we made the reward for matching contents of data elements independent of the uniqueness of the match. The original EBFM algorithm devalues matches if they are common, whereas we found that this reduced the effectiveness of the reconciliation process.

3. PRACTICAL RECONCILIATION

Suppose that a user with a mobile access terminal requests a weather forecast for a particular geographical location. A currently available real Web service returned the following data:

```
<AYPY>
 <1>Moresby</1>
 <2>Papua New Guinea</2>
 <3>-09.26</3>
 <4>147.13</4>
 <5>Moresby, Papua New Guinea</5>
 <6>Sep 08, 2005 - 07:00 PM EDT</6>
 <7>from the SE (140 degrees) at 15 MPH (13 KT)</7>
 <8>greater than 7 mile(s)</8>
 <9>mostly cloudy</9>
 <10>78 F (26 C)</10>
 <11>68 F (20 C)</11>
 <12>69%</12>
 <13>29.85 in. Hg (1011 hPa)</13>
 <14>AYPY 082300Z 14013KT 9999 FEW018 BKN040 BKN140
26/20 Q1011</14>
</AYPY>
```

As one can see, the data returned makes use of cryptic tags and is therefore hard to comprehend at a glance, nor should the user have to consult the documentation page of the service. Here a personal software agent can come to the user's help. We propose that an agent faced with a situation described would retrieve data or use cached data from other information source(s) of the same purpose that the user is familiar with in order to interpret the data such as presented in the example above. For instance, another similar Web service returns the following record reporting the weather conditions of the same geographical location:

```
<AYPY>
 <1 Location>Moresby, Papua New Guinea</1 Location>
 <2 Humidity>69%</2 Humidity>
 <3 Pressure>29.85 in. Hg (1011 hPa)</3 Pressure>
 <4 Sky>mostly cloudy</4 Sky>
 <5 Temperature>78 F (26 C)</5 Temperature>
 <6 Visibility>greater than 7 mile(s)</6 Visibility>
 <7 Wind>from the SE (140 degrees) at 15 MPH (13 KT)
</7 Wind>
</AYPY>
```

In order to demonstrate how "understanding by reconciliation" could be achieved, we tested our proposed approach to ontology reconciliation on two sets of Web services. One of the sets consisted of two public Web services returning weather forecasts. The first of these services was the "World Weather" by InnerGears [9]. It retrieves the weather report for all airports in the world registered by the International Civil Aviation Organization (ICAO) based on a specified ICAO identifier. The second service "Airport Weather" by CapeScience [10] is similar to the first one but returns elements with different tag names. We devised a series of experiments where both services were invoked with the same ICAO identifiers which had been selected so that the weather forecasts returned were on the airports from all over the world. After applying AReXS to the data streams returned by the Web services, the application reported the candidate element pairs and its confidence in each match:

```
Slot 0 [1] <-> Slot 0 [1 Location]: 0.95012
Slot 1 [2] <-> Slot 0 [1 Location]: 0.95575
Slot 4 [5] <-> Slot 0 [1 Location]: 0.97643
Slot 6 [7] <-> Slot 6 [7 Wind]: 0.99999
Slot 9 [10] <-> Slot 4 [5 Temperature]: 0.97845
Slot 10 [11] <-> Slot 4 [5 Temperature]: 0.95376
Slot 11 [12] <-> Slot 1 [2 Humidity]: 0.99923
Slot 12 [13] <-> Slot 2 [3 Pressure]: 0.99968
```

The results of automatically reconciling the data streams returned by two Web services are that the elements labelled <1>, <2> and <5> of the first stream have unsurprisingly been matched with the element <1 Location> of the second stream. Analogously, the elements of the first data stream labelled <7>, <12> and <13> have been matched with the elements <7 Wind>, <2 Humidity> and <3 Pressure> of the second stream, respectively, with the highest possible degree of confidence. The degree of matching confidence between the elements <10> and <11> of the first data stream with the respective element <5 Temperature> of the second stream is somewhat lower because the first service returns both minimum and maximum temperatures, while the second service returns only the latter. Note that the elements of the two data streams could not have been matched by only considering the element names, as would generally occur in an explicit ontology alignment algorithm, such as, for example, Chimaera [11]. Also, it is worth noting that AReXS is able to cope with blank data tags.

One could argue that the matching reported between the elements of the two data streams is self-evident. It is really not impossible for the user to achieve with some effort the same results, but the user with a mobile access terminal usually lacks the time and an appropriate interface required for it.

4. DESIGNING AN ASSISTANT AGENT

In the Internet, it is inherently difficult to predict the information needs of individual agents, let alone which particular data representations they require. Although many proposed multi-agent systems enforce system-wide global ontologies, this is not always desirable or feasible, especially when the agents are embedded in mobile access terminals.

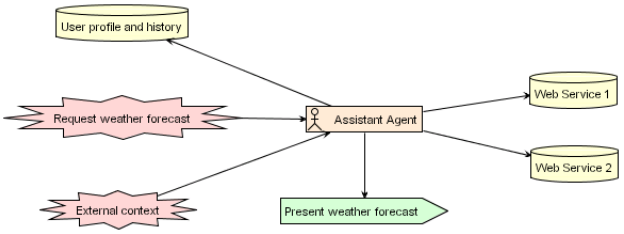


Figure 1. A system overview diagram.

We will now provide a sketch of the design of a system that has in its centre an assistant agent embedded in a mobile phone or PDA. For this purpose, we make use of the ROADMAP/Prometheus methodology [12, 13] of agent-based software engineering which is especially suitable for developing open multi-agent systems. Figure 1 represents an overview diagram of the system to be formed around the Assistant Agent. As Figure 1 reflects, the Assistant Agent perceives a request for a weather forecast by its user. In addition to that, the external context of the request, containing, for example, the current location of the user, serves as another input *percept* for the Assistant Agent. The agent also utilizes some previously collected data consisting of the user profile and the history of the previous actions by the user. Based on this information, the Assistant Agent determines a Web service that it queries for a weather forecast. However, the information returned by the Web service that the user is most familiar with may be incomplete or this Web service may not respond. In such a case, some or all parts of the weather forecast are retrieved from a different Web service. In order to decide in which form the information retrieved should be presented to the user, the Assistant Agent utilizes the data about the user profile and history. For example, provided that the Assistant Agent has cached some data from previous invocations of the user's favourite weather forecast Web service, it is able to "translate" the names of data elements used in the information retrieved from "alien" information sources to the data element names known by the user. As a result, the weather forecast is finally presented to the user in the form which is as close as possible to the one preferred by her/him. In Figure 1 this is reflected by the agent's *action* "Present weather forecast". Please note that the whole process described occurs in a dynamic manner without the use of *any* pre-specified domain ontology.

5. CONCLUSIONS

Even though the Internet is in principle global, a lot of information available in it is of only local significance. Context-dependent invocation of Web services and context-dependent retrieval of Web pages are therefore of an increasing importance. This implies that a growing number of different views of the world will be present in the Internet. However, it seems unlikely that any single proposed solution for adding semantics to the Internet, such as Ontology Web Language (OWL) [14], will be widely accepted in the near future. Even if such an event occurred, it is doubtful that many smaller commercial and individual publishers of information would be willing to devote the time and effort required to comply with a standard that requires ontology development. For this reason, we are proposing and developing technologies and methodologies that cope with heterogeneity and change in information sources and in contexts of their usage by performing implicit ontological reconciliation.

6. ACKNOWLEDGEMENTS

The authors would like to thank the Centre for Perceptive and Intelligent Machines in Complex Environments of the Australian Research Council for supporting the work reported.

7. REFERENCES

[1] *Web Services Activity*. The World Wide Web Consortium, http://www.w3.org/2002/ws/, retrieved on 21 Sept. 2005.

[2] *Extensible Markup Language* (XML). The World Wide Web Consortium, http://www.w3.org/XML/, retrieved on 21 Sept. 2005.

[3] Gruber, T. R. A Translation Approach to Portable Ontologies. *Knowledge Acquisition, 5*(2), 199-220, 1993.

[4] Jennings, N. R., Sycara, K., Wooldridge, M. A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems, 1*(1), 7-38, 1998.

[5] Hou, D. *Automatic Reconciliation of XML Structures*. Honours thesis, Department of Computer Science and Software Engineering, The University of Melbourne, Australia, 2001.

[6] Lister, K., Sterling, L. Agents in a Multi-Cultural World: Towards Ontological Reconciliation. In *AI 2001: Advances in Artificial Intelligence, 14th Australian Joint Conference on Artificial Intelligence, Adelaide, Australia, December 10-14, 2001, Proceedings*. LNCS 2256, Springer 2001, 321-332.

[7] Ikeda, Y., Itoh, F., Ueda, T. Example-Based Frame Mapping for Heterogeneous Information Agents. In *Proceedings of the 3rd International Conference on Multi-Agent Systems* (ICMAS 1998), 3-7 July 1998, Paris, France. IEEE 1998, 150-157.

[8] Klein, M. Supporting Evolving Ontologies on the Internet. In *EDBT 2002 Ph.D. Workshop, Prague, Czech Republic, March 28, Proceedings*. Matfyzpress 2002, 51-58.

[9] *GetWorldWeatherByICAO Documentation*. InnerGears, http://www.innergears.com/WebServices/WorldWeatherByICAO/DOC/WorldWeatherByICAO.htm, retrieved on 22 Sept. 2005.

[10] *Airport Weather Web Service*. CapeScience, http://capescience.capeclear.com/webservices/airportweather/index.shtml, retrieved on 22 Sept. 2005.

[11] McGuinness, D., Fikes, R., Rice, J., Wilder, S. The Chimaera Ontology Environment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, July 30 - August 3, 2000, Austin, Texas, USA. AAAI Press / The MIT Press 2000, 1123-1124.

[12] Kuan, P. P., Karunasakera, S., Sterling, L. Improving Goal and Role Oriented Analysis for Agent Based Systems. In *Proceedings of the 16th Australian Software Engineering Conference* (ASWEC 2005), 31 March - 1 April 2005, Brisbane, Australia. IEEE 2005, 40-47.

[13] Padgham, L., Winikoff, M. Developing Intelligent Agent Systems: A Practical Guide. John Wiley & Sons 2004.

[14] *OWL Web Ontology Language Reference*. W3C Recommendation 10 February 2004, http://www.w3.org/TR/owl-ref/, retrieved on 1 Feb. 2006.

Formalizing Typicality of Objects and Context-sensitivity in Ontologies

Ching-man Au Yeung and Ho-fung Leung
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, Hong Kong
{cmauyeun,lhf}@cse.cuhk.edu.hk

ABSTRACT

In multiagent systems ontologies are essential because they facilitate tasks like communications and reasoning. In this paper, inspired by studies in cognitive psychology, we present a theoretical framework of ontologies that model concepts and properties in a way that typicality of objects can be reflected, and in which context plays a major role in determining the interpretation of the concepts. Typicality is a very common phenomenon observed in different cognitive tasks, while context has been proved to have significant influences on reasoning and categorization. This framework brings together these issues and gives a formal representation of context-sensitive ontologies.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*semantic networks*

General Terms

Human Factors, Theory

Keywords

Semantic Web, Ontologies, Agent Communications, Context

1. INTRODUCTION

With the development of the Semantic Web [3], multiagent systems are expected to play a more essential role in information processing. Ontology [7] also becomes more important because it specifies the definitions of and relations between different concepts. In this paper, we focus on two problems of the Semantic Web. The first problem is the representation of fuzzy concepts and graded membership of instances in ontologies, while the second problem is categorization and forming of taxonomies of concepts in different contexts. We try to solve the problems by proposing a framework for context-sensitive ontologies.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. TYPICALITY AND CONTEXT

2.1 Prototype View and Typicality

Concepts are usually defined as crisp sets of objects in ontologies [11]. However, graded membership in concepts is found to be very common [8]. The *prototype view of concepts* [9] suggests that a concept is represented by a prototype which has all the salient properties of the objects that are classified to this concept. The term *typicality* refers to the degree that an object is considered as a member of a concept, and is determined by the number of matching properties of the object and the concept prototype [8, 9]. Moreover properties that are not necessarily in the definition of a concept (*non-necessary properties*) may also affect typicality of an object.

2.2 Context and Context Effect

In general *context* refers to the situation in which an event happens or an action is taken, or the surroundings that help to understand a particular concept. In this paper, we focus on its role in categorization, since this is a major task in the Semantic Web. According to [1], categorizations depends on the circumstance and perspective. In cognitive psychology, the term "context effect" [5] is used to refer to the influence of context in cognitive tasks. Barsalou [2] suggested that properties of a concept can be classified as context-independent or context-dependent. Roth and Shoben [10] also investigated the effect of context in categorization, and suggested context causes a re-weighing of the importance of the properties in a concept and resulting in a different categorization.

3. A COGNITIVE MODEL FOR CONTEXT-SENSITIVE ONTOLOGIES

Inspired by research on the prototype view and context effect, we propose a cognitive model of concepts which takes into account the calculation of typicality of instances, and propose a framework in which categorization of instances are dependent on the context.

3.1 Definition of an Ontology

Ontology is defined as an explicit specification of conceptualization [6]. Here we define an ontology O as a four-tuple

$O = (C, P, I, R)$, where C is a set of concepts, P is a set of properties of the concepts, I is a set of data instances, and lastly R is a set of rules, propositions or axioms.

3.2 Concepts and Properties

In this model, each concept is basically characterized by a set of properties which are singly necessary and jointly sufficient. In addition, a weight is associated with each property in a concept to reflect its importance. Each instance possesses a set of properties and a value is used to indicate the degree to which it possesses each property.

Definition 1. A **concept** $c \in C$ is a fuzzy subset of I, with a membership function μ_c assigning each instance $a \in I$ a membership grade in this concept.

3.3 Characteristic Vector, Property Vector and Subsumption of Concepts

We propose two data structures to represent how properties characterize concepts, and how individuals possess different properties. A *characteristic vector* of a concept is defined as a vector of real numbers in the range of 0 to 1, in which each element corresponds to the weight of a property.

Definition 2. A **characteristic vector** $\vec{c}_c$ of a concept c is a vector of real numbers, $\vec{c}_c = (c_{c,1}, c_{c,2}, ..., c_{c,n}), 0 \leq c_{c,i} \leq 1$.

A *property vector* of an individual is a vector of real numbers, in which each element corresponds to the degree to which the individual possesses a property.

Definition 3. The **property vector** $\vec{p}_a$ of an individual a is a vector of real numbers, $\vec{p}_a = (p_{a,1}, p_{a,2}, ..., p_{a,n}), 0 \leq p_{a,i} \leq 1$.

In addition, we have the following definitions concerning the concept subsumption relations.

Definition 4. For two concepts c and d, c is said to be **subsumed by** d, denoted by $c \sqsubseteq d$, if and only if $c_{c,i} \geq c_{d,i}$ for all $i = 1, 2, ..., n$.

Definition 5. If $c \sqsubseteq d$, then c is said to be a **sub-concept** of d, and d is said to be a **super-concept** of c.

Definition 6. The set of **defining properties** of c is the set which consists of properties having weights greater than zero in the characteristic vector of a concept c.

Definition 7. The set of **properties possessed by** a is the set which consists of properties having a degree greater than zero in the property vector of an individual a.

3.4 Likeliness of an Individual in a Concept

Likeliness of an individual refers to the degree that an individual is an instance of a concept according to the *defining properties* of a concept. This can be considered as the generalization of the crisp membership in existing ontological frameworks, and is similar to the fuzzy membership grade in fuzzy ontologies like [13]. A suitable formula for calculating likeliness should satisfy the following axioms.

AXIOM 1. *An individual a has a degree of likeliness of* 1 *in a concept c if and only if $c_{a,i} > 0 \rightarrow p_{a,i} = 1$ for all $i = 1, 2, ..., n$.*

AXIOM 2. *An individual a has a degree of likeliness of* 0 *in a concept c if and only if $c_{a,i} > 0$ and $p_{a,i} = 0$ for some $i \in [1, n]$.*

AXIOM 3. *For a concept c, and two individuals a and b, if for some j such that $c_{c,j} > 0$, $p_{a,j} > p_{b,j}$ and $p_{a,i} = p_{b,i}$ for all $i \neq j$, then $\mu_c(a) > \mu_c(b)$.*

AXIOM 4. *For two concepts c and d, and an individuals a, if for some j such that $c_{c,j} > c_{d,j} > 0$, $p_{a,j} > 0$ and $c_{c,i} = c_{d,i}$ for all $i \neq j$, then $\mu_d(a) > \mu_c(a)$.*

AXIOM 5. *For two concepts c and d such that c is a sub-concept of d, the degree of likeliness of an individual a must satisfy $\mu_c(a) \leq \mu_d(a)$.*

A possible function for calculating likeliness is

$$\mu_c(a) = \min_i \{l_i\} \tag{1}$$

where

$$l_i = \begin{cases} p_{a,i} + (1 - c_{c,i}) \times (1 - p_{a,i}) & \text{if } c_{c,i} > 0, p_{a,i} > 0 \\ 0 & \text{if } c_{c,i} > 0, p_{a,i} = 0 \\ 1 & \text{if } c_{c,i} = 0 \end{cases}$$

3.5 Prototype Vector and Typicality

Typicality is a measure of an individual's degree of membership in a concept depending on properties (both necessary and non-necessary) that are shared by most of the instances of the concept. The characteristic vector cannot handle typicality because it only contains information of necessary properties. Thus we introduce here the *prototype vector*. To calculate typicality of an instance, we construct a prototype for the concept, in the form of a *prototype vector*.

Definition 8. The **prototype vector** $\vec{t}_c$ of a concept c is a vector of real numbers, $\vec{t}_c = (t_{c,1}, t_{c,2}, ..., t_{c,n}), 0 \leq t_{c,i} \leq 1$, and is determined by the following equation:

$$\vec{t}_c = \frac{1}{|S|} \sum_{s \in S} \vec{c}_s$$

where S is the set of sub-concepts of c.

Typicality of an instance is determined by a "*weighted feature (property) sum*" [12]. This involves matching the properties in the prototype vector of a concept and the property vector of an individual. The typicality function of a concept, τ_c, should satisfy the following axioms.

AXIOM 6. *An individual a has a degree of typicality of* 1 *in a concept c if and only if $t_{c,i} > 0 \rightarrow p_{a,i} = 1$ for $i = 1, 2, ..., n$.*

AXIOM 7. *An individual a has a degree of typicality of* 0 *in a concept c if and only if $t_{c,i} > 0 \rightarrow p_{a,i} = 0$ for $i = 1, 2, ..., n$.*

AXIOM 8. *For a concept c, and two individuals a and b, if for some j such that $t_{c,j} > 0$, $p_{a,j} > p_{b,j} > 0$ and $p_{a,i} = p_{b,i}$ for all $i \neq j$, then $\tau_c(a) > \tau_c(b)$.*

AXIOM 9. *For a concept c, and two individuals a and b, if for some j such that $t_{c,j} > 0$, $p_{a,j} > p_{b,j} = 0$ and $p_{a,i} = p_{b,i}$ for all $i \neq j$, then $\tau_c(a) > \tau_c(b)$.*

AXIOM 10. *For two concepts c and d, and an individuals a, if for some j such that $t_{c,j} > t_{d,j} > 0$, $p_{a,j} > 0$ and $t_{c,i} = t_{d,i}$ for all $i \neq j$, then $\tau_d(a) > \tau_c(a)$.*

Here, we present a possible function for calculating an individual's typicality in a concept. The typicality of an individual a of a concept c, denoted by $\tau_c(a)$ is given by:

$$\tau_c(a) = \frac{\vec{p}_a \cdot \vec{t}_c}{\sum_{i=1}^{n} t_{c,i}} \quad (2)$$

4. CONTEXT IN CATEGORIZATION

The model described above serves as a basis for formalizing context in ontology. We consider contexts as different conditions which cause an agent to process information thereafter with different perspectives, and focuses on different properties of a concept.

4.1 Formal Definitions

We define a context x as a collection of propositions, objects and concepts perceived by an agent in the Semantic Web environment.

Definition 9. A **context** x is a three-tuple $\langle N_r, N_c, N_i \rangle$, where $N_r \subseteq R$, $N_c \subseteq C$ and $N_i \subseteq I$.

When an agent perceives a particular context, the agent forms a certain perspective, which is a certain view point on the concepts and individuals. We define perspective as a view point from which an agent assigns different weights to properties in concepts to reflect its foci on certain properties.

Definition 10. A **perspective** v is a mapping from the set $C \times P$ into the set of real numbers in the range 0 to 1, which represents the weights of the properties in the concepts: $v : C \times P \longrightarrow [0, 1]$.

Furthermore, we define the function *View* that maps a context to a perspective formed in an agent.

Definition 11. Let X be the set of contexts and V be the set of perspectives, the function ***View*** is a mapping from X to V: $View : X \longrightarrow V$.

4.2 Contextualized Ontology

After an agent has chosen a perspective according to the context, the agent forms a context-sensitive ontology by applying the weights to the properties in concepts. This can be formalized by a set of interpretations similar to those in description logics [4]. An interpretation assigns to each concept a characteristic vector which contains the corresponding weights of the properties according to the perspective.

Definition 12. An **interpretation** for context i, m_i, consists of a domain Δ_i and an interpretation function $\mathcal{I}_i$: $m_i = \langle \Delta_i, \mathcal{I}_i \rangle$, where Δ_i refers to the domain and $\mathcal{I}_i$ is an interpretation function.

The interpretation function $\mathcal{I}_i$ is a function that maps each concept $c \in C$ to a fuzzy subset of I, and assign a characteristic vector to the concept c. $\mu_{c,i}$ is the membership function of c in the context represented by interpretation m_i. On the other hand, the typicality of an individual can be different due to the different weights of the properties. We use $\tau_{c,i}(a)$ to denote the typicality of individual a in concept c under the interpretation m_i.

4.3 Contextual Subsumption Relations

A hierarchy of concept can be constructed by determining concept subsumption relations. Once contexts are modeled, a subsumption relation between concept c and d, say $c \sqsubseteq d$, may only hold within a certain context.

Definition 13. For two concepts c and d, c is said to be **subsumed by** d under the interpretation m_x, denoted by $c \sqsubseteq_{m_x} d$, if and only if $c_{c,j} \geq c_{d,j}$ for all $j = 1, 2, ..., n$.

5. CONCLUSION AND FUTURE WORKS

This paper attempts to provide a theoretical framework of context-sensitive ontology, which handles both fuzzy membership grades, typicality of objects, and models context so that agents are able to interpret concepts according to the current situations. Future works on this framework include investigation of how the function *View* can be implemented such as by using some learning algorithms, and extension of the framework to handle other aspects of context such as disambiguating terms with multiple meanings.

6. REFERENCES

[1] Varol Akman and Mehmet Surav. Steps toward formalizing context. *AI Magazine*, 17(3):55–72, 1996.

[2] Lawrence W. Barsalou. Context-independent and context-dependent information in concepts. *Memory and Cognition*, 10(1):82–93, 1982.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Sci. Am.*, 284(5):34–43, 2001.

[4] Franz Baader et al., editor. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[5] K. M. Galotti. *Cognitive Psychology In and Out of the Laboratory*. Wadsworth, third edition, 2004.

[6] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[7] N. Guarino and R. Poli. Formal ontology in conceptual analysis and knowledge representation. *Int. Journal of Human and Comp. Stud.*, 43:625–640, 1995.

[8] E. H. Rosch. On the internal structure of perceptual and semantic categories. In T. E. More, editor, *Cognitive Development and the Acquisition of Language*. New York: Academic Press, 1973.

[9] E. H. Rosch. Cognitive represerntations of semantic categories. *Journal of Exp. Psy.*, 104:192–233, 1975.

[10] Emilie M. Roth and Edward J. Shoben. The effect of context on the structure of categories. *Cognitive Psychology*, 15:346–378, 1983.

[11] U. Sattler. Description logics for ontologies. In *Proc. of the International Conference on Conceptual Structures (ICCS 2003)*, 2003.

[12] E. E. Smith and D. L. Medin. *Categories and Concepts*. Harvard University Press, 1981.

[13] Umberto Straccia. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.

Service-Oriented and Autonomous Distribution and Provision of Multimedia Contents

Fuyuki Ishikawa
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku,
Tokyo, Japan
f-ishikawa@nii.ac.jp

Nobukazu Yoshioka
National Institute of Informatics
2-1-2 Hitotsubashi,
Chiyoda-ku, Tokyo, Japan
nobukazu@nii.ac.jp

Shinichi Honiden
National Institute of Informatics
and The University of Tokyo
honiden@nii.ac.jp

ABSTRACT

This work considers encapsulation of multimedia contents as services, and their autonomous provision and distribution by agents. This approach enables content providers to have their contents provided and distributed together with their own business logics regarding interaction with other services and contents, control logic for various options upon provision, and value-added services. This paper proposes a framework for development of agents that provide and distribute such content services. In the framework, the interaction logic of agents is first developed in the same way as in development of ordinary Web services. Control descriptions are then given for each aspect, such as selection of partner agents, negotiation on service options, and migration. The separated descriptions for each aspect facilitate incremental development and test, as well as later modification, of agents with complex behaviors.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures—*Languages*

General Terms

Design

Keywords

Agent Framework, Service-Oriented Computing, Multimedia, Mobile Agents, Agreement

1. INTRODUCTION

Rapid progress in computing and network technologies has enabled people to use various kinds of multimedia contents, e.g., movies, business documents, and electronic medical charts, with their computers or mobile devices. Operations on multimedia contents include not only to play and edit but also to translate between different formats, extract properties to annotate the contents, and generate customized view. Currently, multimedia contents are often treated as data and operated using applications, e.g.,players, installed on users' computers, and content providers can only set control parameters for them in advance. Consequently, flexible policies regarding how their contents are manipulated after delivered cannot be incorporated, for example, flexible license control such as license division for multiple contents or multiple devices, design and license restriction upon content composition, and content-specific value-added functions such as generation of customized view.

In response to this problem, this work considers encapsulation of multimedia contents by autonomous agents. Multimedia contents are encapsulated together with business logic or control policies given by content providers, provided and distributed autonomously by agents in complying with the policies. This model enables for content providers to keep control on how their contents are handled and to provide value-added services such as integration of related contents or content-specific customization.

Agents encapsulating multimedia contents need to discover and cooperate with other agents providing other contents or functions to form composite services or contents. Issues in implementation of such behavior are not specific to this domain and have been investigated especially in the Service-Oriented Computing (SOC) activity. In SOC, software systems provide services to other applications through published and discoverable interfaces, and the services can be invoked over a network [2]. It thus leads to autonomous composition of a new service by integrating existing ones at runtime. In addition to the above general issues, agent mobility is necessary to implement (possibly autonomous) contents distribution as well as to allow for function migration. In addition, agents should be able to be part of another agent, to represent atomic contents embedded in composite contents or functions embedded in contents. They especially should work cooperatively, for example, migrate between hosts together as a unit and adjust their interfaces and behaviors so that they are seen as one consistent composite agent [4]. Besides, services that provide multimedia contents have various kinds of application-specific service options as mentioned above. It is thus necessary to introduce a mechanism to handle agreements on application-specific service options [3], though the effort on SLAs (Service Level Agreements) for SOC has considered only application-independent service options, especially QoS parameters.

The control flow of agents including these features can be very complex. For example, when an agent wants to initiate interaction with another agent providing required services or contents, it first needs to discover and select an agent playing the required role. It may then migrate and/or ne-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

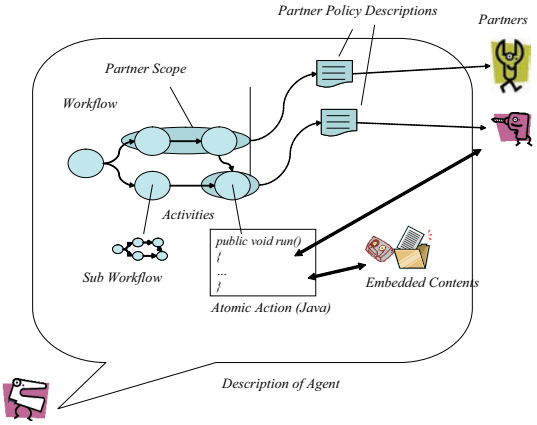


Figure 1: description Model for Active Contents

gotiate about service options such as licenses. The whole control flow woven with such behaviors for each partner agent is very complex and unmanageable. In response to the problem, this study proposes a framework named Freedia for development of agents encapsulating multimedia contents. It aims to enable developers to handle the complexity in development of agents including various aspects such as partner discovery, migration, and agreements on service options. The proposed Freedia framework allows developers to first specify the main interaction logic of an agent, that is, control and data flow between atomic actions including messaging actions. It is thus possible to start with development of agents that simply interact with their partners through remote messaging and without migration nor agreements. Developers can then incorporate policy descriptions that specify requirements and related behaviors for each interaction partner. The separated descriptions for each aspect facilitate incremental development and test, as well as later modification, of agents with complex behaviors.

2. FREEDIA FRAMEWORK

2.1 Workflow and Partner Scopes

This section describes the proposed framework, Freedia, for development of agents that autonomously distributes and provides multimedia contents. The model for agent description in Freedia is shown in Figure 1. The main logic, including interaction with partner agents as well as internal behavior, is described as a workflow description. The workflow description is based on a common approach in process languages for Service-Oriented Computing [5, 6], that is, interaction with partner agents of required types is specified without concrete description for binding partners. Information used for discovery and selection of partners is given separately. The workflow description specifies control and data flow between actions in a graph-oriented way. The activities are implemented in Java and include messaging actions (sending or receiving messages in FIPA ACL), internal computation, and/or operation on contents the agent contains. The notation and operational semantics of workflow are based on Web Services Flow Language [5]. For each partner declared in the workflow description, policy descriptions are given to control behaviors related to the partner, including discovery, negotiation and selection for service agreements, and migration/synthesis (*Partner Policy* descriptions described in Session 2.2).

In definition of activities in a workflow, types of the partner agents the agent interacts with during execution of the activity are declared. Types of partner agents are called *roles* and denote interface of agents in terms of observable messaging actions. If an agent has a certain role, it means the agent can send or receive messages in a certain sequence specified in a certain interaction protocol. In definition of an activity in a workflow, a partner is declared by specifying a set of roles required for the partner.

As partners that the agent can interact with are declared in each activity, a set of activities where interaction with a partner occurs can be extracted. This set of activities is called a *partner scope*, and used by Freedia to determine when to activate initialization of a partnership, e.g., discovery and negotiation, or termination, e.g., disposal of service agreements. For each partner, initialization is activated when execution enters the scope for the partner, that is, when execution reaches an activity where interaction with the partner occurs. Termination is activated when execution exits of the scope for the partner, that is, by default when execution gets out of the outermost loop enclosing the partner scope (intuitively, when interaction with the partner turns out not to occur anymore). It is also possible to specify whether interaction is iterated in a loop with the same partner or switching the partner. Details of initialization and termination are described in the following.

2.2 Partner Policy Descriptions

2.2.1 Overview

For each partner declared in the workflow description, Partner Policy descriptions are given to specify requirements for and behaviors related to the partner. Currently, a partnership is built between two agents, and connection of partnerships between agent pairs leads to a collaborative group of agents. Preliminary interaction is held to build a partnership according to Partner Policy descriptions. In each agent pair for any protocol, one agent is classified as a *requester* and initially sends a request in preliminary interaction to build a partnership. The other is a *responder* and waits for a request for interaction. Below each item in Partner Policy descriptions is first described separately, and how to integrate them is then described.

2.2.2 Items in Partner Policy Descriptions

2.2.2.1 Agent Properties.

Requirements for properties of partner agents, such as organizations providing them and service quality, are given in Partner Policy descriptions. In the case of a requester, these requirements descriptions are used for discovery of agents, together with required roles declared in the workflow description. In the case of a responder, they are used to determine whether to accept a request for interaction or not.

2.2.2.2 Interaction Style.

Requirements for interaction styles with partners are specified in Partner Policy descriptions (based on the authors' previous study [4]). Here interaction styles include whether migration is conducted to be on the same host or not, and where the agent pair gathers if conducted. They also include whether an agent compound is established or not, where multiple agents work cooperatively and migrate as a unit (called agent synthesis), and what kinds of control are enforced in the agent compound (if established). For example, monitoring and operating on messages allows for encapsu-

lation of the contained agent, e.g., inhibiting the containing agent from providing services directly. It also allows for override of services by the contained agent, e.g., adding additional access control check or filtering. Each agent has requirements on these aspects in Partner Policy descriptions, and Freedia framework matches them and builds agreements on interaction styles if matched. The Freedia framework is also responsible to enforce the agreements, i.e., to activate migration of an agent, to monitor and pass messages to given handlers of containing agents, and so on.

2.2.2.3 Service Agreements.

Behaviors regarding agreements on service options, both interaction to build agreements and change to comply with the agreements, are specified in Partner Policy descriptions (based on the authors' previous study [3]).

For behaviors to build agreements, templates to implement Iterated Contract Net Protocol (CNP) [1] to build agreements are provided. Various existing or emerging techniques are thus incorporated such as iterated negotiation. The Iterated CNP protocol considers the most complex situation, iterated negotiation and selection between a consumer and multiple candidates for service providers. Each internal behavior in this process is implemented by using Java abstract classes provided by the Freedia framework. In Partner Policy descriptions, such components are specified and the framework conducts the negotiation and selection process by invoking the given components and messaging actions including time-out check according to the protocol.

For change in behaviors to comply with established agreements, Aspect-Oriented rule descriptions are provided. Some service options are strongly related to the workflow and originally expected to be customizable, e.g., bandwidth and size of movies. In the case such options are determined not in each request but in long-term service agreements, rule descriptions for setting parameters in the workflow can be adopted. Other service options are independent of the workflow and require additional actions, e.g., access control check and merger of value-added contents. In this case, rule descriptions that insert additional actions before/after activities or upon receiving/sending messages.

2.2.3 Putting Together

At runtime, initialization and termination are conducted for each partner at execution points determined according to partner scopes. Each item described above includes preliminary interaction for initialization, i.e., requirements matching and negotiation. At runtime, requirements matching for service properties and migration/synthesis is integrated into the given interaction for service agreements by Freedia. Partnerships are established only if matching for both aspects is succeeded and service agreements are successfully built. After agreements for interaction styles and for service options are established, migration/synthesis is conducted and rules for enforcing service agreements are activated (if necessary). When execution of workflow descriptions exits out of the partner scope, dissolve of synthesis and deactivation of rules for enforcing service agreements are conducted as finalization (if necessary).

Every item in Partner Policy descriptions does not always need to be given. For example, if requirements for service properties are not given, candidates for a partner are obtained by querying directory services only with roles. This default behavior, together with separation of concerns in the description model, facilitates incremental development, that is, first develop and test the interaction logic, and then give control descriptions for each aspect and test one by one.

2.3 Experiences

The proposed Freedia framework is currently implemented in Java. The framework includes a visual editor and debugger for workflow descriptions, as well as the infrastructure for interpretation of the workflow and Partner Policy descriptions and agent management.

Three applications developed using the proposed Freedia are described briefly. In the first one, agents encapsulating electronic medical charts are delivered to different organizations while obeying policies given by the original distributor. Second, agents encapsulating instruction manual of health supplements follow movement of human users, proactively showing instruction of the supplements timely and offer value-added services such as setting parameters of electronic exercise equipments. Last, agents encapsulating e-learning contents interact with each other to support experiences of English conversation between students while cooperating with agents given teachers' management policies. All the scenarios include aspects of partner discovery, migration/synthesis (e.g., following mobile users or override of services using the mechanisms for agent synthesis), and service agreements (authentication and authorization or learning level). These applications were developed incrementally. First only the interaction logic of agents was implemented, without any policies on coupling, negotiation, or synthesis, i.e., only workflow descriptions are given. Policies about discovery, cooperative migration, and service agreements were introduced and tested gradually.

3. SUMMARY

This study has considered encapsulation of multimedia contents as agents. This approach enables content providers to have their contents provided and distributed together with their own business logics regarding interaction with other agents that provide required services or contents, control logic for various options upon provision, and value-added services. For implementation of such agents, this study has proposed a framework that facilitates incremental development of agents with complex behavior including various aspects.

4. REFERENCES

[1] FIPA. Fipa iterated contract net interaction protocol specification. `http://www.fipa.org/specs/fipa00030/SC00030H.html`, December 2002.

[2] H. Haas. Web services. `http://www.w3.org/2002/ws/`, June 2004. (Access: May 2005).

[3] F. Ishikawa, N. Yoshioka, and S. Honiden. Incorporating agreements on service options into bpel-based services. In *International Conference on Intelligent Agents, Web Technology and Internet Commerce (IAWTIC'2005)*, November 2005 (to appear).

[4] F. Ishikawa, N. Yoshioka, Y. Tahara, and S. Honiden. Developing service-oriented compounds of cooperative mobile agents. In *Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE'2005)*, July 2005.

[5] F. Leymann. Web services flow language (wsfl1.0). `http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf`, 2001.

[6] S. Thatte et al. Business process execution language for web services, version 1.1. `http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/`, May 2003.

Deriving Axioms Across Ontologies *

Dejing Dou
Computer and Information Science
University of Oregon
Eugene, Oregon 97403
dou@cs.uoregon.edu

Drew McDermott
Computer Science Department
Yale University
New Haven, Connecticut 06520
drew.mcdermott@yale.edu

ABSTRACT

Ontologies play a key role in agent communication and the emerging Semantic Web. Axioms are an important component of ontologies to describe the relationships among the concepts. The current research on ontology mapping and ontology translation mainly focuses on how to map and translate the vocabularies and associated data instances from one ontology to another. However, when extending one ontology using axioms from another, we must confront the problem of translating axioms. In this paper, we show that simple symbol replacement will not solve the problem of axiom translation because of *asymmetry of translation*. Instead we extend our inferential ontology translation for facts (ground atomic formulas) and queries to cover axioms using a method we call *axiom derivation*.

Keywords

Ontologies, the Semantic Web, Agent Communication, Ontology Translation

1. INTRODUCTION

Ontologies, which can be defined as the formal specification of a vocabulary of concepts and the relationships among them, play a key role in both agent communication and the emerging Semantic Web. We can distinguish several different *semantic heterogeneity* problems among multiple agents and web services, which may use different ontologies.

1. *Ontology mapping:* Finding correspondences between the symbols of two ontologies; the correspondences are often equivalences and subset-superset (or subproperty-superproperty) relationships.
2. *Ontology translation:* Translating a dataset (assertions) or a query expressed using one ontology into a form that uses a different ontology.
3. *Theory translation:* Translating the axioms of one ontology into the vocabulary of another, preserving their validity.

Much previous work (see a survey in [4]) has been in the first category; ours [2] has been mainly in the second. In this paper we focus on the third, in which the translation approach for simple assertions by replacing the symbols of one vocabulary with the corresponding symbols from another will not work for axioms in general. For example, in one genealogy ontology G_1, husband and married are two concepts (properties) and there may be a first-order logic (FOL) axiom to describe their relationship:

$$(\forall x, y)\mathsf{husband}(x, y) \rightarrow \mathsf{married}(x, y) \qquad (1)$$

where x and y are universal variables to represent man and woman respectively. Suppose another genealogy ontology G_2 has male_partner and in_marriage as corresponding (mapping) properties. Some facts (assertions) expressed in the language of G_1, can simply be translated into the language of G_2 by replacing corresponding properties:

$$\mathsf{husband}(\mathsf{John}, \mathsf{Mary}) \rightsquigarrow \mathsf{male\_partner}(\mathsf{John}, \mathsf{Mary})$$

$$\mathsf{married}(\mathsf{John}, \mathsf{Mary}) \rightsquigarrow \mathsf{in\_marriage}(\mathsf{John}, \mathsf{Mary})$$

We use a symbol $\rightsquigarrow$ to represent the process of translation. The translations are correct in terms of the semantics of G_1 and G_2, where husband can be thought of as a special kind of male_partner. However, if we use the same technique to translate the axiom (1) of G_1 to G_2, we will get:

$$(\forall x, y)\mathsf{male\_partner}(x, y) \rightarrow \mathsf{in\_marriage}(x, y) \qquad (2)$$

It is obvious that (2) is not always true since a man is a partner of a woman doesn't mean that he must be in a marriage with her; (2) is not a valid axiom in G_2.

What is the reason that the above translation for facts is correct, and the translation from (1) to (2) is not correct? Can we find a formal method for translating axioms from a source ontology to a target ontology that guarantees that the translations are also true in the target ontology?

In this paper, we will give a formal characterization of ontology translation that unifies the translation of facts, queries and axioms in one inferential framework (section 2). We call the translation of axioms from one ontology to another ontology *axiom derivation*. We point out that it is the *asymmetry of translation* that makes axiom derivation difficult. In section 3, we will elaborate how to derive axioms across different ontologies as the extension of our previous work for translating datasets and queries.

*This work has been partly supported by the DARPA DAML project, under contract F30602-00-2-0600.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. FRAMEWORK

2.1 Merged Ontologies & Bridging Axioms

In this section we first briefly review our previous work [2] on ontology translation. We assume that in order to translate facts from one ontology to another there must be a *merged ontology* in which all the relevant symbols are allowed to interact. For example, consider the pair of genealogy ontologies G_1 and G_2. The mappings between them can be represented as first order axioms:

$$(\forall x, y)\mathsf{husband}(x, y) \rightarrow \mathsf{male\_partner}(x, y) \quad (3)$$

$$(\forall x, y)\mathsf{married}(x, y) \leftrightarrow \mathsf{in\_marriage}(x, y) \quad (4)$$

We call these axiomatic mapping rules *bridging axioms*. We have developed a strongly typed first-order logic language, *Web-PDDL*, to express bridging axioms. These axioms are embedded in a *merged ontology* complete with namespace declarations and type-equivalence rules.

2.2 Inferential Ontology Translation

We will use the symbol $\rightsquigarrow$ to indicate translation: $\alpha \rightsquigarrow \beta$ means that β is the translation of α. We call the ontology O_s that α uses the *source* ontology and O_t, the one β uses, the *target*. In the case of sets of assertions ("datasets"), we stipulate that the translation of α_d is simply the strongest set of assertions, β_d, in O_t entailed by α_d. A consequence of this stipulation is that

$$(KB; \alpha_d) \rightsquigarrow \beta_d \textit{ only if } (KB; \alpha_d) \models \beta_d$$

where we have added to the left-hand sides the symbol KB to refer to the merged ontology with bridging axioms.

If α_q is a query in O_s, its translation is a query β_q in O_t such that any answer (set of bindings) to β_q is also an answer to α_q. In other words:

$$(KB + DB_\beta; \alpha_q) \rightsquigarrow \beta_q \textit{ only if } (KB + DB_\beta; \theta(\beta_q)) \models \theta(\alpha_q)$$

for any substitution θ, where $KB + DB_\beta$ now includes both the merged ontology and the facts from the target data instances. The point is that β_q need not be (and seldom is) *equivalent* to α_q, in the sense that any answer to one is an answer to the other. All we need is that any answer to β_q be an answer to α_q. If we take O_s to be G_2 and O_t to be G_1 (section 2.1), the query $\mathsf{male\_partner}(?x, ?y)$ in G_2 will be translated into the query $\mathsf{husband}(?x, ?y)$ in G_1. But the set of all husbands is not equivalent to the set of all male_partners.

2.3 Asymmetry of Translation

Given our approach, translation exhibits certain asymmetries that one must be wary of. We will subscript the symbol $\rightsquigarrow$ with a "Q" to indicate the query translation ($\rightsquigarrow_Q$), and with a "D" (for "data") to indicate the assertion or dataset translation($\rightsquigarrow_D$). (We leave the subscript off in those cases where the context allows either reading.) The asymmetry is: if β_t is the translation of α_s: $(KB; \alpha_s) \rightsquigarrow \beta_t$ that doesn't mean α_s is the translation of β_t: $(KB; \beta_t) \rightsquigarrow \alpha_s$. Slightly less obviously, if $(KB; P) \rightsquigarrow Q$ we can't conclude $(KB; \neg P) \rightsquigarrow \neg Q$.

Instead (not surprisingly), negation ends up involving the same duality as query translation. Assume that R is an expression which can be derived from KB and $\neg P_s$ by inference. Using the deduction theorem in FOL and considering that $\neg P_s \rightarrow R$ is an equivalent to $\neg R \rightarrow P_s$, we know that

$$\begin{aligned}(KB; \neg P_s) \vdash R \quad &\Leftrightarrow \quad KB \vdash (\neg P_s \rightarrow R) \\ &\Leftrightarrow \quad KB \vdash (\neg R \rightarrow P_s) \\ &\Leftrightarrow \quad (KB; \neg R) \vdash P_s\end{aligned}$$

This gives us a way to translate negations. We can think of P_s as a "ground query" ($\theta(P_s) = P_s$): Given P_s, try to find a $Q_t^{'}$, which satisfies $(KB; Q_t^{'}) \vdash P_s$. But this is just the problem of translating the query P_s: $(KB; P_s) \rightsquigarrow_Q ?$.

Therefore, if the query translation of P_s is $Q_t^{'}$, $\neg Q_t^{'}$ can be derived from KB and $\neg P_s$ by the data translation and vice versa:

$$\begin{aligned}(KB; P_s) \rightsquigarrow_Q Q_t^{'} \quad &\Rightarrow \quad (KB; \neg P_s) \rightsquigarrow_D \neg Q_t^{'} \\ (KB; P_s) \rightsquigarrow_D Q_t^{'} \quad &\Rightarrow \quad (KB; \neg P_s) \rightsquigarrow_Q \neg Q_t^{'}\end{aligned}$$

Axioms are usually more complex than a typical dataset element, and it would be useful if we could attack this complexity by translating the pieces of a complex formula and composing the results. For conjunctions and disjunctions, composition of translation is straightforward. Translating implications includes an algorithm for translating negations.

3. OUR APPROACH

3.1 Conditional Facts and ICF Axioms

To explain our approach to theory (axiom) translation, we first show how to translate *conditional facts* using our inference engine, OntoEngine [2]. A conditional fact is a formula of the form:

$$P_1 \wedge \cdots \wedge P_i \cdots \wedge P_n \rightarrow Q_1 \wedge \cdots \wedge Q_j \wedge \cdots \wedge Q_m$$

where all $P_i (1 \leq i \leq n)$ and $Q_j (1 \leq j \leq m)$ are ground atomic formulas (facts). All axioms, such as axioms 1, can be put in this form which we called *ICF (Implicative Conjunction Form)*, but of course axioms have quantified variables. It is unusual, but not unheard of, for people to need to express that some facts are true only if some other facts are also true:

$$\begin{aligned}&\mathit{precedes}(\mathit{deathof}(\mathit{Roosevelt}),\ \mathit{endof}(\mathit{WW2})) \rightarrow \\ &\qquad \mathit{president}(\mathit{Truman},\ \mathit{endof}(\mathit{WW2}))\end{aligned}$$

"If Roosevelt died before the end of World War 2, then Truman was president at the end of World War 2."

3.2 Conditional Fact Translation

Conditional fact translation is the translation of a conditional fact from the source ontology to the target ontology. It is not as simple as translating both the antecedent and conclusion in the source ontology to the antecedent and conclusion in the target ontology by the data translation with forward chaining. This is a typical example which we need to consider asymmetry of translation since the translation of implications actually includes the translation of negations and disjunctions, which has been discussed in section 2. For example, suppose we have a simple conditional fact:

$$@\mathsf{Onto1} : \mathsf{P}(A, B) \rightarrow @\mathsf{Onto1} : \mathsf{Q}(A, B)$$

where @Onto1:P means predicate P from Onto1. We want to translate this conditional fact to Onto2. Suppose @Onto1:P(A,B) can be translated to @Onto2:R(A,B) and @Onto1:Q(A,B) can

be translated to @Onto2:S(A,B), both by forward chaining. It doesn't mean that

$$@\mathsf{Onto2} : \mathsf{R}(\mathsf{A},\mathsf{B}) \rightarrow @\mathsf{Onto2} : \mathsf{S}(\mathsf{A},\mathsf{B})$$

is true. However, if there exists @Onto2:R'(A,B) satisfies:

$$@\mathsf{Onto2} : \mathsf{R}'(\mathsf{A},\mathsf{B}) \rightarrow @\mathsf{Onto1} : \mathsf{P}(\mathsf{A},\mathsf{B})$$

we know that

$$@\mathsf{Onto2} : \mathsf{R}'(\mathsf{A},\mathsf{B}) \rightarrow @\mathsf{Onto2} : \mathsf{S}(\mathsf{A},\mathsf{B})$$

is a true statement in Onto2. To get this, we need to do query translation with backward chaining from @Onto1:P(A,B) to a corresponding expression in the target ontology, such as @Onto2:R'(A,B).

It should be obvious that this process yields a valid result, in the sense that the translated fact follows from the original fact and the axioms. If backward chaining from the antecedent fails to find any goals in the target ontology, then the antecedent of the translated conditional fact will be the empty, or false, making the translation itself equivalent to true — and hence useless.

3.3 Axiom Derivation

Axiom derivation for ICF axioms still can be thought of as an inference process, if we can transform the axioms to conditional facts and transform the conditional facts back to axioms. The idea is to substitute Skolem constants for the variables temporarily. (A similar technique was used in [3].) In general, axiom derivation can be broken into three steps:

From ICF axioms to conditional facts: we can use Universal Elimination and Existential Elimination to transform ICF axioms to conditional facts. For instance, suppose that we have an axiom in the source ontology O_s:

$$(\forall x, y)@\mathsf{O\_s} : \mathsf{P}(x, y) \rightarrow (\exists z)@\mathsf{O\_s} : \mathsf{Q}(x, z) \wedge @\mathsf{O\_s} : \mathsf{R}(z, y)$$

We can substitute the universal quantified variables with constants (e.g., Atx and Bty) and substitute the existential quantified variables with uniquified Skolem terms (e.g., Skz01):

$$@\mathsf{O\_s} : \mathsf{P}(\mathsf{Atx}, \mathsf{Bty}) \rightarrow$$
$$@\mathsf{O\_s} : \mathsf{Q}(\mathsf{Atx}, \mathsf{Skz01}) \wedge @\mathsf{O\_s} : \mathsf{R}(\mathsf{Skz01}, \mathsf{Bty})$$

Conditional facts translation: suppose that the target ontology is O_t and we already have the merged ontology of O_s and O_t. The conditional fact in O_s can be translated to O_t. By backward chaining from @O_s:P(Atx,Bty) and forward chaining from @O_s:Q(Atx,Skz01) and @O_s:R(Skz01,Bty), we may finally get a conditional fact in O_t:

$$@\mathsf{O\_t} : \mathsf{S}'(\mathsf{Atx}, \mathsf{Ctc}) \wedge @\mathsf{O\_t} : \mathsf{T}'(\mathsf{Ctc}, \mathsf{Bty}) \rightarrow$$
$$@\mathsf{O\_t} : \mathsf{U}(\mathsf{Atx}, \mathsf{Skz01}) \wedge @\mathsf{O\_t} : \mathsf{V}(\mathsf{Skz01}, \mathsf{Skd02})$$
$$\wedge @\mathsf{O\_t} : \mathsf{W}(\mathsf{Bty}, \mathsf{Skz01})$$

where Ctc is a constant and Skd02 is a new generated Skolem term by forward chaining.

From conditional facts to ICF axioms: we can use Universal Generalization [3] and Existential Introduction to transform conditional facts back to ICF axioms.

For example, considering the above translated conditional fact in O_t, we can use Universal Generalization to replace all constants which have substituted universal variables with universal variables. Atx, Bty and Ctc can be replaced by x, y and c. We also can use Existential Introduction to replace all Skolem terms with existential variables. Skz01 and Skd02 can be replaced by z and d. Therefore, the generated ICF axiom looks thus:

$$(\forall x, y, c)@\mathsf{O\_t} : \mathsf{S}'(x, c) \wedge @\mathsf{O\_t} : \mathsf{T}'(c, y) \rightarrow$$
$$(\exists z, d)@\mathsf{O\_t} : \mathsf{U}(x, z) \wedge @\mathsf{O\_t} : \mathsf{V}(z, d) \wedge @\mathsf{O\_t} : \mathsf{W}(y, z)$$

It's not so obvious that this procedure works, but we have proved that it does. The detail proof can be found in a long version[1] of our paper.

3.4 Axiom Derivation for Time Ontologies

We have evaluated our axiom derivation algorithm in some real application scenarios, in which complex ontologies always have large sets of axioms. There are several time ontologies in current popular knowledge bases and (semantic) web resources, such as Cyc time [2], SUMO time and OWL-Time (formerly DAML-Time) [3]. They describe the temporal concepts, such as Instant, Interval and durationOf, and their relationships, which are represented as sets of logic axioms (e.g. OWL-Time has around 180 first order axioms.)

Researchers have manually built some mappings among the concepts of some time ontologies, but have not talked about how to represent the axioms in different ontologies. The axioms in the Cyc time ontology can be automatically translated to the OWL-Time ontology with our algorithm.

4. RELATED WORK AND CONCLUSION

To the best of our knowledge, the only other work on axiom (theory) translation is [1]. This work presents a formalism for knowledge translation based on the theory of contexts [3]. The authors define knowledge translation in terms of truth, and propose, as we do, using a theorem prover to perform the translations. However, the paper does not say exactly *how* the theorem-proving process would work.

Based on the inferential ontology translation framework, our paper has pointed out the asymmetry of translation and described an algorithm for theory translation from one ontology to another, which we call axiom derivation. Although our algorithm is provably correct, practical application requires further work on problems of incompleteness and redundancy, because our algorithm does not by itself guarantee that the axioms we produce are sufficient, nor does it avoid producing axioms that are already present in the target ontology.

5. REFERENCES

[1] S. Buvac and R. Fikes. A Declarative Formalization of Knowledge Translation. In *Proceedings of the ACM CIKM conference*, 1995.

[2] D. Dou, D. Mcdermott, and P. Qi. Ontology translation on the semantic web. *Journal of Data Semantics*, II:35–57, 2005.

[3] J. McCarthy and S. Buvac. Formalizing context (expanded notes). In A. Aliseda, R. van Glabbeek, and D. Westerstahl, editors, *Computing Natural Language*. University of Chicago Press, 1997.

[4] N. F. Noy. Semantic integration: A survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70, 2004.

[1] www.cs.uoregon.edu/~ dou/research/papers/onto.pdf
[2] http://www.cyc.com/cycdoc/vocab/time-vocab.html
[3] http://www.isi.edu/~ pan/OWL-Time.html

Web Presentation System using RST Events

Arturo Nakasone
Dept. of Information and Communication Eng.
University of Tokyo
7-3-1 Hongo, Bunkyo Ku
Tokyo 113-8656, Japan
+81-3-5841-6347
arturo@mi.ci.i.u-tokyo.ac.jp

Helmut Prendinger
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda Ku
Tokyo 101-8430, Japan
+81-3-4212-2650
helmut@nii.ac.jp

Mitsuru Ishizuka
Dept. of Information and Communication Eng.
University of Tokyo
7-3-1 Hongo, Bunkyo Ku
Tokyo 113-8656, Japan
+81-3-5841-6347
ishizuka@i.u-tokyo.ac.jp

ABSTRACT
Constructing presentations as stories is about presenting events that receive their relevance and meaning within the context of the presentation, and about means to define a well-structured relationship between events and the whole presentation scheme. Information in the form of annotated web content has become the most preferred source to create functional and interesting presentations because of its flexibility and availability. Nevertheless, due to the large and diverse amount of information contained in the web, the automatic construction of presentations becomes increasingly difficult in terms of the content annotation and selection in order to define events and the arrangement of such events in a coherent and story-like manner.

In this paper, we present a novel method for the construction of web presentations by implementing a simple event annotation scheme based on Rhetorical Structure Theory and an ontology-based reasoning engine that handles event sequencing to produce presentations in the form of dialogs between character agents.

Categories and Subject Descriptors
H.3.3 [**Information Systems**]: Information Interfaces and Presentation – *web based interaction.*

General Terms
Algorithms, Documentation, Human Factors.

Keywords
Storytelling, Narrative Construction, Story Ontology, Rhetorical Structure Theory

1. INTRODUCTION
Throughout the course of our lives, we are constantly incorporating experiences to enrich our personality. These experiences, perceived in the form of world events, are organized by our minds, so we can better grasp their significance. Narrative construction through stories helps us not only to organize these experiences, but also to communicate them to others. In this context, stories are unique sequences of events, mental states, or happenings involving human beings as characters or actors [1].

The notion of "story event" is defined in several ways, each one to suit a particular way to deliver the intended message to the audience. Events can be represented as plain text, speech with intonation features, video fragments, or multimedia content based on web technology (web events). Of all these types of events, web events are the most easily accessible and flexible in terms of construction and content delivery. However, given the large and diverse body of information, the creation of a coherent sequence of events that conforms to the principle of narrative construction can become a difficult task, not only in terms of event selection, but also in terms of event sequencing.

In this paper, we will propose a new method for story construction for presentations using web events with a simple annotation scheme based on Rhetorical Structure Theory (RST Events) [5], and an ontology-based story event arrangement engine.

The rest of the paper is organized as follows. The next section will present related work in the field of story construction applications. Section 3 discusses our approach to story construction using RST. Finally, our plans for future research topics and a summary will conclude the paper.

2. RELATED WORK
Several researchers assume a concept of "event" in order to organize the content of their storytelling applications. In the work of Silva et. al [9], the story construction process is implemented through the creation of events called StoryBits. Here, each event corresponds to a piece of text that will be read out by an agent. The author defines the links between StoryBits prior to the execution of the storytelling, which makes story updating cumbersome when the number of events increases.

Similarly, Wolff et. al [11] used the concept of domino tiles to let children construct their own story, based on beginning and ending conditions of each event, which in this case are video clips from a TV show. Other works dealt with different kinds of output like text [3,10] and 3D characters [2,6]. In most of these cases, events were generated for the particular purpose of the application,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005…$5.00.

which renders them unfit for reusability in other scenarios or even in other kind of stories. Other research like that of Geurts et. al [4] used ontology driven mechanisms to select and present events (in this case multimedia content) with very interesting results, but the dependence on highly customized annotated information may complicate their deployment in a more heterogeneous medium like the Web.

3. STORY CONSTRUCTION USING RST EVENTS

3.1 Basic Definitions

An RST Event is defined as a single utterance in the context of dialog acts, performed by an Agent. As implied by its name, this utterance corresponds to the minimal unit within a rhetorical relation as established in RST. In most cases, this unit is a sentence, although it could also represent a phrasal clause. Linking RST Events through rhetorical relations forms a Dialog structure. In terms of RST, Dialogs represent the tree formed by the rhetorical analysis on a set of RST Events, and help us establish the conditions under which Events will be presented.

On the other hand, Concept Ontology refers to a hierarchical organization of concepts relevant to a unique main topic, also defined as Taxonomy of Concepts. In our model, a Concept is the description of a topic that will constrain the scope of a Dialog, that is, to define what a particular Dialog is about. Thus, a Scene is defined as a set of Dialogs that have a Concept as their common topic. In practical terms, Scenes are organized around the Concept Ontology, with one Scene object per Concept.

3.2 RST Event Organization

As stated before, RST Events are linked through rhetorical relations to create Dialogs. In order to allow easy content creation and dialog flexibility, rhetorical relations in our model are classified as either Inferable, which are relations that can be automatically generated during the presentation, or Non-Inferable, which are relations that cannot be generated and must be included as content to guarantee dialog coherence. The *Non-Inferable relations* that have to be determined for RST Events in the present version of our model are: Background, Enablement, Evidence, Circumstance, Elaboration, Solutionhood, Cause, Result, Purpose, Evaluation, and Restatement. By using these relations, the application combines the events located in Dialogs to create an inferable relation, through the use of discourse markers. For example, to create a Justify relation, the application can use Events related by an Evidence relation and emphasize through verbal cues such as "I am sure because …", or "I believe so because …" When this construction is uttered by an Agent, it will produce the sense of justification of one Event for another. *Inferable relations* defined in the present version of the model are: Motivation, Concession, Contrast, Sequence, and Joint. For a detailed explanation of the meaning of each relation, please refer to the work of Mann and Thompson on Rhetorical Structure Theory [5].

3.3 RST Event Presentation

The concepts outlined in the previous sections define how information is stored. In order to use that information to create presentations, our model defines the concepts of Talk and Conversation.

A Talk is a unique path in the set of Dialogs defined for a Scene that determines the precise, sequential way in which Agents present RST Events. In order to comply with narrative principles, Talks are centered in the notion of Intra-Talk Conflict, which is primarily defined as a question in an Elaboration or Solutionhood relation when events are taken from only one Dialog. This type of conflict is solved in the model when a solution of detailed information is given to the initial question. Additionally, an Intra-Talk Conflict can include Contrast or Sequence relations. In this case, taking events from more than one Dialog creates Talks. Therefore, a Talk can be represented by the following pattern of relations (in Extended Backus-Naur Form-like notation; ">>" denotes the "next state" relation):

```
Background >>
((Elaboration | Solutionhood) >>
(Circumstance | Purpose | (Result >> Cause) | Interpretation)+ >>
(Motivation | Enablement | Evidence | Justify | Concession)* >>
(Restatement)* >>
(Evaluation)*) >>
(Sequence | Contrast)*
```

A Conversation is defined as a set of different Talks. An appropriate selection of Concepts (as determined by the designer) characterizes a Conversation. Like in the case of Talks, narrative principles are ensured through the use of a particular type of conflict called Inter-Talk Conflict. Inter-Talk Conflicts are mostly attained by the use of a Contrast relation between two Talks, and its resolution is mainly based on a direct comparison on the number of non-inferable relations in both Talks. Nevertheless, the use of conflicts between Talks is not restricted to Contrast relations, since Talks can be performed one after the other without any conflict at all, as in the case of Talks that constitute a list of events (defined by a Sequence or Joint relation). Thus, a Conversation can be represented by the following sequence of processes:

```
Opening >> Talk1
        >> (Sequence | Joint | Contrast)
        >> Talk2 >> … >> TalkN >>
Closing
```

The Opening and Closing processes refer to discourse markers that the system uses in order to introduce the Agents to the Conversation, and their utterances are defined depending on the role of each Agent in it. The present version of our model supports two main roles: Knowledgeable (the Agent that gives the explanation) and Not-Knowledgeable (the Agent that receives the explanation as a listener). Therefore, in this paradigm, the conversation is guided into the context of a person who looks for information from another regarding a particular topic, and the use of such utterances as greetings is in accordance to this environment.

Fig. 1 shows a code example in OWL [8] of a Conversation object that has one Scene with one Dialog and a Concept named "Multimedia", which means that each Dialog defined inside that Scene is about the Multimedia topic.

```
<Conversation rdf:ID="Conversation_Multimedia">
   <hasScenes>
     <Scene rdf:ID="Scene_Multimedia">
       <belongsScenes rdf:resource="#Conversation_Multimedia"/>
       <hasResolutionOutcome
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
       >0</hasResolutionOutcome>
       <hasConcept rdf:resource="#Multimedia"/>
       <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
       >Scene about the Multimedia Concept</rdfs:comment>
       <hasDialogs>
         <Dialog rdf:ID="Dialog_Multimedia_1">
           <hasRelName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
           >CONCESSION</hasRelName>
           <hasNucleus>
             <Event rdf:ID="RU_3">
               <hasUtterance
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
               >Transmedial research turns them to attractive and useful
information</hasUtterance>
               <hasEventTypes rdf:resource="#Response_Uncertain"/>
             </Event>
           </hasNucleus>
           <hasSatellite rdf:resource="#RU_2"/>
             <Event rdf:ID="RU_2">
               <hasUtterance
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
               >Raw digital data which consist of 0/1 are hardly
meaningful for human</hasUtterance>
               <hasEventTypes rdf:resource="#Response_Uncertain"/>
             </Event>
           </hasSatellite>
         </Dialog>
       </hasDialogs>
     </Scene>
   </hasScenes>
 </Conversation>
```

Figure 1. Content Code in OWL

3.4 Putting Everything Together: RST Event-based Story Ontology

Using the above-mentioned concepts as guidelines to implement our application, we have developed an OWL-based [8] story ontology to guide the decision process of constructing the story. The ontology diagram is shown in the Figure 2.

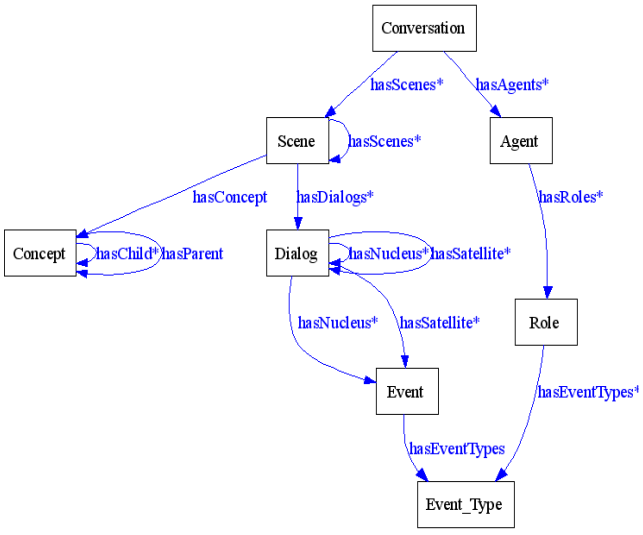


Figure 2. RST Event-based story ontology diagram

4. CONCLUSION AND FUTURE WORK

In this paper we briefly outlined a new method that may flexibly construct coherent stories using existing web content in the form of annotated events using RST.

Our most immediate consideration to improve this model is the addition of user interaction in the form of personalized, emotion-based stories. To this effect, we plan to use our emotion recognition module that detects the user's affective state and may thus support a user-dependent event selection process [7]. In this way, the application will be able to create presentations considering the user's sensitivity to the content being presented. For instance, if the user perceives certain events as boring or frustrating, the application may decide to skip that part of the story or introduce new events based on rhetorical relations that increase user's motivation.

Since there is still a fairly good amount of information that has to be manually annotated, another model improvement is to transform non-inferable relations into inferable ones. In this way, we can make the content creation task easier.

5. REFERENCES

[1] Bruner J. Acts of Meaning, Cambridge, Harvard University Press, 1991.

[2] Cavazza M., Charles F., and Mead S.J.. Interacting with virtual characters in interactive storytelling. In Proceedings First Conference on Autonomous Agents and Multiagent Systems (AAMAS-02), New York, 2002. ACM Press, 318-325

[3] Gervas P., Diaz-Agudo B., Peinado F, and Hervas R. Story plot generation based on CBR. Journal of Knowledge Based Systems 18 (2-3), 2005, 235-242.

[4] Geurts J., Bocconi S., van Ossenbruggen J., and Hardman L. Towards Ontology-Driven Discourse: From Semantic Graphs to Multimedia Presentations. In Proceedings of the International Semantic Web Conference 2003, 597-612.

[5] Mann, W., and Thompson S. *Rhetorical structure theory: A theory of text organization.* Marina del Rey, CA: Information Sciences Institute, 1987.

[6] Mateas M., and Stern A. Towards integrating plot and character for interactive drama. In Proceedings AAAI Fall Symposium on Socially Intelligent Agents: The Human in the Loop, AAAI Press, Technical Report FS-00-04, 2000, 113-118.

[7] Nakasone A., Prendinger H., and Ishizuka M. Emotion recognition from electromyography and skin conductance. *The Fifth International Workshop on Biosignal Interpretation (BSI-05)*, Tokyo, Japan, 2005, 219-222.

[8] OWL - Web Ontology Language http://www.w3.org/TR/2004/REC-owl-features-20040210/

[9] Silva A., Raimundo G., and Paiva A. Tell me that bit again… Bringing Interactivity to a Virtual Storyteller. In Proceedings of the 2nd International Conference on Virtual Storytelling - ICVS 2003, 146-154.

[10] Szilas N., Rety J. Minimal Structures for Stories. In Proceedings of the 1st ACM workshop on Story representation, mechanism and context. New York, USA, 2004, 25-32.

[11] Wolff A., Mulholland P., and Zdrahal Z. Scene-Driver: A Narrative-Driven Game Architecture Reusing Broadcast Animation Content. In Proceedings of the working conference on Advanced visual interfaces, AVI 2004, Gallipoli, Italy, 94-97.

Believable agents

Learning Empathy: A Data-Driven Framework for Modeling Empathetic Companion Agents

Scott W. McQuiggan
Department of Computer Science
North Carolina State University
Raleigh, NC 27695, USA
swmcquig@ncsu.edu

James C. Lester
Department of Computer Science
North Carolina State University
Raleigh, NC 27695, USA
lester@csc.ncsu.edu

ABSTRACT
Affective reasoning plays an increasingly important role in cognitive accounts of social interaction. Humans continuously assess one another's situational context, modify their own affective state accordingly, and then respond to these outcomes by expressing empathetic behaviors. Synthetic agents serving as companions should respond similarly. However, empathetic reasoning is riddled with the complexities stemming from the myriad factors bearing upon situational assessment. A key challenge posed by affective reasoning in synthetic agents is devising empirically informed models of empathy that accurately respond in social situations. This paper presents CARE, a data-driven affective architecture and methodology for learning models of empathy by observing human-human social interactions. First, in CARE training sessions, one trainer directs synthetic agents to perform a sequence of tasks while another trainer manipulates companion agents' affective states to produce empathetic behaviors (spoken language, gesture, posture). CARE tracks situational data including locational, intentional, and temporal information to induce a model of empathy. At runtime, CARE uses the model of empathy to drive situation-appropriate empathetic behaviors. CARE has been used in a virtual environment testbed, and an evaluation suggests that the CARE paradigm can provide the basis for effective empathetic behavior control for companion agents.

Categories and Subject Descriptors
H.5.1 [**Multimedia Information Systems**]: Artificial, augmented, and virtual realities; Evaluation/methodology.

General Terms
Algorithms, Design, Experimentation, Human Factors.

Keywords
Synthetic Agents, Affective Reasoning, Empathy.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

1. INTRODUCTION
Recent years have witnessed significant progress on synthetic agents. With a broad range of applications in entertainment, education, and training, foundational work on synthetic agents has yielded expressive models of embodied cognition and behavior that support rich interactions in virtual environments [1, 2, 6, 16, 20, 27]. Complementing advances in cognition and behavior, affective reasoning [10, 12, 22, 25] has begun to play a central role in synthetic agents [4, 5, 20, 21], and the community is now well positioned to investigate affective reasoning in the context of social interaction [17, 23, 26]. Transitioning affective synthetic agents into the social arena could yield companion agents that provide motivating support and comfort. *Companion agents* can facilitate social interaction, a critical capability in virtual environments for education [5, 7, 20] and training [26]. Companion agents help users cope with frustration [5], deal with stress [26], and counsel children on social behaviors [23].

Empathy is a key component of social interaction [14]. Because empathetic companion agents hold much promise for socially engaging virtual environments, empathy modeling is a logical next step in the evolution of synthetic agents. One can distinguish two fundamental approaches to modeling empathy: analytical and empirical. In the *analytical* approach, models of empathy can be constructed by analyzing the findings of the empathy literature. However, empathy is not well understood. It is only in the past two decades—this is very recent in the history of psychology—that empathy has become a focus of study for social psychologists [9]. Perhaps as a result of its limited study, while we have expressive computational models of affect, e.g., the OCC model [22], we do not have similarly rich models of empathy. Moreover, because empathetic reasoning requires drawing inferences about another's intentions, her affective state, and her situational context, devising a universal model of empathy seems to be well beyond our grasp at the current juncture.

An alternative to analytically devising models of empathy for synthetic agents is the *empirical* approach. If somehow we could create models of empathy that were derived directly from observations of "empathy in action," we could create empirically grounded models based on human-human empathetic behaviors exhibited during the performance of a specific task within a given domain. While it is not apparent that this approach could produce a universal model of empathy—a universal model may not even be achievable, at least in the near term—the empirical approach could nonetheless generate models of empathy that significantly



Figure 1: Treasure Hunt virtual world with companion agent (left) and the target's agent (right).

extend the communicative capabilities of socially intelligent companion agents.

The empirical approach calls for a data-driven framework for modeling empathy. This paper presents CARE,[1] a data-driven affective architecture and methodology for learning empirically informed models of empathy from observations of human-human social interactions. During training sessions, CARE monitors situational data including locational, intentional, and temporal information while one trainer (the *target*) directs her synthetic agent to perform a sequence of tasks in a virtual environment as another trainer (the *empathizer*) reactively manipulates her synthetic agent's affective state to produce empathetic behaviors (spoken language, gesture, and posture). Inducing a model of empathy, CARE uses situational data as predictive features for empathy assessment (when to exhibit an empathetic behavior) and for empathy interpretation (which levels of valence and arousal should be chosen, i.e., the affective state). At runtime, CARE uses the resulting model to drive situation-appropriate empathetic behaviors in the companion agent as it interacts with actual users.

The paper is organized as follows: Section 2 provides background on empathy and affective reasoning in synthetic agents. Section 3 presents the CARE architecture and methodology. CARE has been used to create a model of empathy for a companion agent inhabiting Treasure Hunt (Figure 1), a virtual environment in which a user and a companion agent search for treasures. Section 4 describes the CARE implementation and its generation of the empathy models for the Treasure Hunt companion agent. Section 5 presents an evaluation of CARE based on a 31-subject experiment. The study suggests that CARE models can provide the basis for effective empathetic behavior control for companion agents.

2. Empathetic Synthetic Agents

Devising computational models of empathy contributes to the broader enterprise of modeling affective reasoning [24]. Beginning with Elliott's implementation [10] of the OCC model [22], advances in affective reasoning have accelerated in the past few years, including the appearance of a sophisticated theory of appraisal [12] based on the Smith and Lazarus Appraisal Theory [19]. We have also begun to see probabilistic approaches to assessing users' affective state in educational games [7] and investigations of the role of affect and social factors in pedagogical agents [3, 5, 11, 17, 20, 26]. Recent work on empathy in synthetic agents has explored their affective responsiveness to biofeedback information and the communicative context [26]. It has also yielded agents that interact with one another and with the user in a virtual learning environment to elicit empathetic behaviors from its users [23].

Empathy is a complex socio-psychological construct. Defined as "the cognitive awareness of another person's internal states, that is, his thoughts, feelings, perceptions, and intentions" [15], empathy enables us to vicariously respond to another via "psychological processes that make a person have feelings that are more congruent with another's situation than with his own situation" [14].

Social psychologists describe three constituents of empathy. First, the *antecedent* consists of the empathizer's consideration of herself, the target's intent and affective state, and the situation at hand. Second, a*ssessment* consists of evaluating the antecedent. Third, *empathetic outcomes*, e.g., behaviors expressing concern, are the products of assessment [9] including both affective and non-affective outcomes (e.g., judgment, cognitive awareness). Two types of affective outcomes are possible. In *parallel outcomes*, the empathizer mimics the affective state of the target. For example, the empathizer may become fearful when assessing a target's situation in which the target is afraid. In *reactive outcomes*, empathizers exhibit a higher cognitive awareness of the situation to react with empathetic behaviors that do not necessarily match those of the target's affective state. For example, empathizers may become frustrated when the target does not meet with success in her task, even if the target herself may not be frustrated. Accurately modeling parallel and reactive empathetic reasoning presents significant challenges.

3. DATA-DRIVEN EMPATHY MODELING

The prospect of creating an "empathy learner" that can induce empirically grounded models of empathy from observations of human-human social interactions holds much appeal. To this end, this paper proposes CARE, an affective data-driven paradigm that learns empathetic assessment (when to be empathetic) and empathetic interpretation (how to be empathetic). CARE consists of a trainable agent architecture and a two-phase methodology of training and learning.

3.1 Architecture

The CARE architecture operates in two modes: empathetic model induction in which the architecture interacts with two trainers (depicted in the diagram with dotted arcs), and runtime operation, in which it manages empathetic behaviors for a companion agent interacting with a user (depicted in the diagram with solid arcs) (Figure 2):

- **Empathetic Model Induction:** Trainers interact with CARE via interfaces through which they direct synthetic agents in the virtual environment. The virtual environment tracks all

[1] CARE: Companion-Assisted Reactive Empathizer.

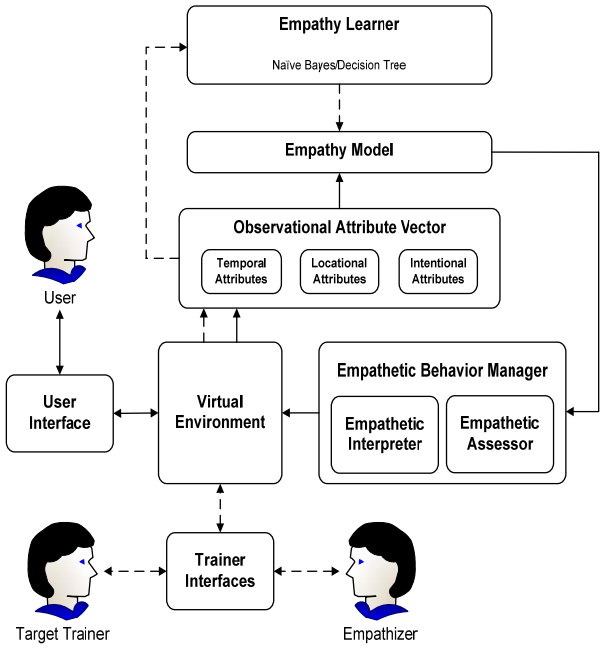


Figure 2: General Architecture

activities in the world and reports observable attributes pertaining to temporal, locational, and intentional information. These are passed to the empathy learner during the training phase. During the subsequent learning phase, the learner induces a model of empathy that is operational, i.e., it can be used at runtime.

- **Runtime Operation:** Users interact with CARE via an interface through which they direct a synthetic agent in the virtual environment. Throughout their experience, they interact with a companion agent controlled by CARE. The virtual environment again tracks all activities in the world and monitors the same observable attributes reported to the empathy learner during empathetic model induction. The induced model is used by the empathetic behavior manager to (1) assess the situation to determine *when* to be empathetic, and (2) interpret situations deemed "empathy-worthy" to decide *how* to be empathetic. When a situation calls for empathy, a suitable empathetic behavior (including speech, gesture, and posture) is selected for execution by the companion agent to react empathetically to the user's situation. Spoken components of companion agent empathetic behaviors explicitly state the affective state being conveyed with commonly associated gestures and posture, i.e., *dropped shoulders*, *arms crossed*, *looking down*, and *head shaking* gestures and postures accompany the companion agent's verbal communication, "This has become quite frustrating."

3.2 Training and Learning

In the training phase, CARE's trainable agent must be exposed to social situations similar to the ones it will encounter at runtime. Because empathy by its very nature involves multiple actors (here we focus on two), the training experience should revolve around

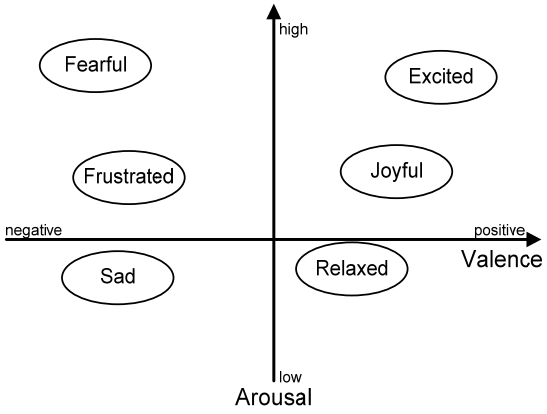


Figure 3: Two-dimensional Affective Space

the interaction of multiple subjects in situations that elicit empathetic behaviors.

CARE training sessions are therefore situated in task-oriented scenarios involving two trainers, a *target* and an *empathizer*, each of whom is represented by a synthetic agent in the 3D virtual environment where training takes place. The target, whom is given a multi-objective mission to complete, controls her agent to navigate and perform tasks in the virtual environment from a first person point-of-view (POV). It is the task of the empathizer, who looks on from a third-person POV, to monitor the target's activities and select suitable empathetic affective states based on the target's observed behaviors. Selecting an affective state causes her agent to perform an empathetic behavior.

To collect empathy data that is as representative as possible of that which will be encountered by the companion agent at runtime, training sessions must satisfy the following requirements:

- ***Affective space coverage***: At each stage of the mission, to promote the target's experiencing a range of emotions spanning the classic two-dimensional affective space defined by *valence* (degree of attraction, ranging from negative to positive) and *arousal* (level of stimulation, ranging from low to high) [18] (Figure 3), the target should be faced with goals of varying degrees of difficulty: some should be very easy to achieve, while others should be very challenging. For example, in Treasure Hunt, the virtual environment that serves as a testbed for CARE, some treasures are in plain view of the target while others are partially occluded and some are hidden altogether. Some targets should be exposed to virtual environments in which goals are easy to achieve, and some should be introduced into worlds in which goals are difficult to achieve. Thus, in some Treasure Hunt worlds, targets can score a specified number of points by collecting treasures very easily, while other worlds pose significant challenges stemming from the accessibility and varying point values. These unique situations offer opportunities for users to experience a variety of reactive emotions.

- ***Double-blind training***: Training sessions should be conducted such that the target is unaware that an empathizer is at the controls of the empathetic behaviors of the companion agent in the virtual world. Likewise, restricting the empathizer's

environment to the virtual world (i.e., without access to the target's facial or vocal expressions) enables empathetic decisions to be based solely on inferences from the observed virtual world (thus, similar inferences are likely to be made by the empathy models at runtime).

- ***Empathy-centered control***: The empathizer should be able to focus exclusively on empathy decision making. Thus, navigation control for the companion agent is provided by an autonomous path planning mechanism that ensures that the companion agent is always within a specified proximity to the target's agent in the virtual environment and that the empathizer has an adequate view of the target agent's experiences.

- ***Training session length***: Each training session must strike a careful balance between being long enough to yield a large body of data and short enough so that the trainers do not become overly fatigued. In the Treasure Hunt environment, experimentation indicated that 7-minute sessions satisfied this requirement and provided sufficient data for empathy modeling.

- ***Controlled affective expression***: Minimizing the complexity of the empathizer's task can be achieved by limiting the set of emotions at her disposal. For example, empathizers in Treasure Hunt have access to six affective states: excited, joyful, relaxed, fearful, frustrated and sad. This particular set of emotions was chosen because it covers the four quadrants of the two-dimensional affective space [18] and considers three levels arousal (high, medium and low) for each level of valence (positive or negative).

- ***Uniform agent personae***: While investigating different personae is a promising direction for future work, e.g., pedagogical agent personae experiments [3], baseline training should control for personae by holding both the target's agent and the empathizer's agent constant throughout training sessions.

- ***Situation data collection intervals***: Situation data should be collected at least as often as significant events occur, where an event is deemed significant if it can plausibly affect the empathizer's decisions. In Treasure Hunt, locational data were collected when events in the world indicated notable state changes, e.g., an agent's entering of a room, while some temporal data were monitored continuously, e.g., the amount of time between goal achievement. A typical training session in Treasure Hunt yields approximately 6,000-9,000 data points.

Accurately modeling empathy requires a representation of the situational context that satisfies two requirements. First, it must be sufficiently rich to support empathetic assessment and empathetic interpretation. Second, it must be encoded with features that are readily observable at runtime so that they may drive companion agents' empathetic decision making. CARE therefore employs an expressive representation of all activities in the virtual environment by encoding them in an observational attribute vector that is used in both modes of operation: during empathetic model induction, the *observational attribute vector* is passed to the empathy learner for model generation; during runtime operation, the attribute vector is monitored by the empathetic behavior manager for determining empathetic

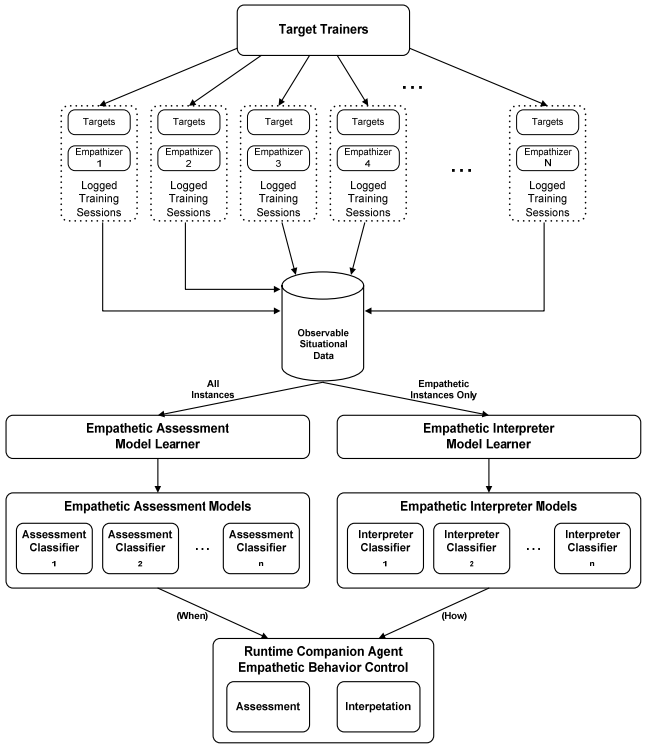


Figure 4: CARE Framework Data Flow.

behavior. CARE's observable attribute vector represents three interrelated categories of features for making empathetic decisions:

- **Temporal features**: CARE tracks the amount of time that has elapsed since the target/user arrived at the current location, since the target/user achieved a goal, since the empathizer/companion agent last behaved empathetically, and since the target/user was last presented with an opportunity to achieve a goal.

- **Locational features**: CARE continuously tracks the location of all agents in the environment. It monitors locations visited in the past, locations recently visited, locations not visited, and locations being approached. Locations are associated with specific areas in the virtual environment or areas containing significant objects or obstacles, e.g., goals or locked doors.

- **Intentional features**. CARE tracks goals being attempted (as inferred from locational and temporal features, e.g., approaching a location where a goal can be achieved), quantity and quality of goals achieved, the rate of goal achievement, and the effort expended to achieve a goal (as inferred from recent exploratory activities and locational features).

In the CARE implementation for Treasure Hunt, the observational attribute vector encodes 192 features. During empathetic model induction, an instance of the vector is logged every time a significant event occurs. On average, vectors are updated several hundred times each minute. At runtime, the same features are updated continuously by the virtual environment and are used by the empathetic behavior manager to select situation-appropriate empathetic behaviors. Figure 4 shows how information from



Figure 5: A relaxed companion agent and target agent.



Figure 6: A frustrated companion agent and target agent.

observations in CARE training sessions flows from the training phase to the learning phase for empathetic model induction.

Finally, in the learning phase, CARE induces a dual model of empathy. One component will be used at runtime to support empathetic assessment, and the other will be used to support empathetic interpretation. CARE's empathy learner first uses all of the data collected in the training session to induce the empathetic assessment model. Induction may be based on any standard classifier learning technique.

Two versions of CARE have been implemented in Treasure Hunt, one with a naïve Bayes classifier and one with a decision tree classifier. The evaluation reported in Section 5 discusses the performance of both approaches. CARE's empathy learner next uses a subset of the data collected in the training sessions to induce the empathetic interpretation model. Here, it only considers data instances in which empathy was in fact exhibited. The second induction produces a model of empathy interpretation that at runtime is used to guide agent's empathetic behaviors. The products of the learning phase are two classifiers used to determine when and how the companion agent should be empathetic as dictated by a generalized model induced from all of the empathizing trainers' empathetic behavior decisions. Because the classifiers employ features directly observable in the environment, they can be easily integrated into the runtime behavior control systems of companion agents in the form of rules or probabilistic statements.

4. EXAMPLE APPLICATION

The CARE paradigm has been used to train models of empathy and to control the behavior of a companion agent at runtime in Treasure Hunt, a virtual environment testbed in which targets/users are instructed to collect treasures in the allotted time. After introducing the Treasure Hunt virtual environment, we describe the implementation and present an illustrative example of CARE'S generating an empathetic behavior.

4.1 Treasure Hunt Virtual Environment

Treasure Hunt is a prototype virtual environment featuring a synthetic agent controlled by the user and a companion agent whose empathetic behaviors are controlled by CARE. The user navigates the 3D virtual world in search of hidden (and some not-so-hidden) treasures. Each treasure box is labeled with the value of its contents, representing points the user obtains when collecting the associated treasure. Throughout the users' quest for treasure, the companion agent follows along and expresses empathetic behaviors as appropriate situations arise in the users' experiences (Figure 5).

4.2 Implementation

CARE's empathetic assessment model and interpretation model have been implemented using naïve Bayes and decision tree approaches. A discussion of their relative performance follows in Section 5. The empathetic models were induced from a dataset consisting of a 192-dimensional observational attribute vector. The observational attribute vector consists of temporal, locational, and intentional features. For example, a sliding ten-second window was used as a temporal feature for tracking user goal attainment, while a binary locational feature monitored whether the user had yet visited the docks or rocky beach area, and an intentional feature was used to detect when the user was moving in the direction of a high-valued goal in the her view.

Treasure Hunt was implemented using a high-performance 3D game platform from Valve Software. The virtual environment, observational attribute monitoring, and empathetic models were implemented with Valve's Source™ engine (the game engine for Half-Life[2]) and the accompanying 3D game platform for Half-Life[2].

4.3 Example Scenario

To illustrate the empathetic behavior control posed by CARE, consider the following scenario, which repeatedly played out in CARE training sessions. As we catch up with the user, she has navigated her synthetic agent throughout the virtual environment struggling to find significant, high-valued treasure. The user and empathizer are aware that the user has not yet met her expected treasure collection quota (as depicted in the graphical HUD representation in the bottom corner of the display) and is quickly running out of time. Only 30 seconds remain.

Now, the user has found her agent's way into a location on the beach of the Treasure Hunt virtual environment, a location

visited by the user's agent in the early moments when the session began. The empathizer realizes that this particular location has been previously visited and was already determined to be an area without any treasure boxes. It has now been over one minute since the user last discovered any treasure at all. Assessing the situation, the empathizer selects the frustrated affective state, thereby initiating a behavioral sequence in which the companion agent announces her frustration by directly stating, "This is becoming quite frustrating," and using gestures and posture similar to the companion agent depicted in Figure 6. (The agent's speech segments are stored in pre-rendered audio clips.)

CARE's empathy learner monitored a variety of environmental characteristics, including those described above, during its training sessions. These recorded instances aid the empathy models in reproducing similar appropriate inferences in analogous situations where time is running out, the user's agent is in a previously visited location known to be without treasure, and the user's intended treasure collection goal is likely to fail. Thus, given the same situation with CARE driving the empathetic behaviors of the companion agent at runtime, empathetic assessment and interpreter models are likely to make similar appropriate empathetic decisions. The following section discusses how accurately the models learned by the agent are able to predict empathizer actions.

5. EVALUATION AND DISCUSSION

This section discusses the user study and CARE training sessions, assesses the naïve Bayes and decision tree classification approaches used for modeling empathetic assessment (when) and empathetic interpretation (how), and suggests several design implications.

5.1 User Study

In a formal evaluation, more than two hours of data were gathered from thirty-one subjects in an Institutional Review Board (IRB) of North Carolina State University approved user study. Participation included 25 targets and 6 empathizers. There were 20 male subjects and 5 female subjects varying in race, ethnicity, age and martial status who participated as training targets. There were 3 male and 3 female subjects participating as training empathizers. On average, empathizers completed 4 training sessions (SD = 1.94), each with its own unique target trainer.

Before empathizers began training, they completed Davis's Interpersonal Reactivity Index (IRI) [8], which provides a measure of an individual's empathetic nature. The index consists of 28 statements in which respondents are instructed to rate the degree to which each item describes them on a Likert scale of 0 to 4. The result is a set of 4 subscale values quantifying the following qualities of empathy: fantasy, perspective taking, empathic concern and personal distress [10]. Female empathizers scored higher than male empathizers in each quality except for perspective taking, in which the males scored one-half point higher than the female empathizers. Subjects were found to be representative of the general population in empathetic qualities, as reported in [8].

Next, empathizers were provided standard definitions of emotion and empathy, similar to those discussed in Section 2, and they were instructed to rely on these definitions for their empathetic decision making. In the meantime, training targets were prepared for the session and completed a pre-session demographic survey. Training targets were unaware of the empathizer's role until the training session was complete. In fact, the empathizer and target were housed in separate rooms for the entire session and were never exposed to the other's presence.

Once sessions began, empathizers observed training target interactions and made empathetic behavioral decisions based on the situation with which they were presented. Sessions lasted for 7 minutes. Both training targets and empathizers received post-session surveys and were interviewed to gain insight into the overall experience. In these interviews it was often mentioned by training targets that they felt the role of the companion agent was solely to provide "emotional commentary." In post-interviews with empathizers it was discovered that up to an additional set of 4 emotions, for a set of set of 10 emotions, would have been preferred.

5.2 Experiment

Both naïve Bayes and decision tree models were induced from data collected in the training sessions described above. As noted earlier, 192 observational attributes were used to define the feature vectors. Naïve Bayes and decision tree classifiers are effective machine learning techniques for generating preliminary predictive models. Naïve Bayes classification approaches produce probability tables that can be incorporated into runtime systems and used to continually update probabilities for monitoring when and how to be empathetic. Decision trees provide interpretable rules that support runtime decision making. Both the naïve Bayes and decision tree machine learning classification techniques are useful for preliminary predictive model induction for large multidimensional data. Because it is unclear precisely which runtime variables are likely to be the most predictive, naïve Bayes and decision tree modeling provide useful analyses that can inform more expressive machine learning techniques (e.g., Bayesian networks) that also leverage domain experts' knowledge.

All models were evaluated using a tenfold cross-validation scheme for producing training and testing datasets. In this scheme, data is decomposed into ten equal partitions, nine of which are used for training and one used for testing. The equal parts are swapped between training and testing sets until each partition has been used for both training and testing. Tenfold cross-validation is widely used for obtaining a sufficient estimate of error [28].

5.3 Results

Cross-validated ROC curves are useful for presenting the performance of classification algorithms for two reasons. First, they represent the positive classifications (true positives), included in a sample, as a percentage of the total number of positive classifications along the vertical axis, against the negative classifications (false positives) as a percentage of the total number of negative classifications [28]. Second, the area under ROC curves is widely accepted as a generalization of the measure of the probability of correctly classifying an instance [13]. Figure 7 shows the ROC curves for CARE's naïve Bayes and decision tree classification approaches for modeling empathetic assessment. Figure 8 shows ROC curves for CARE's naïve Bayes and decision tree classification approaches for empathetic interpreter modeling. Associated areas under each of the curves can be found in the figures' captions.

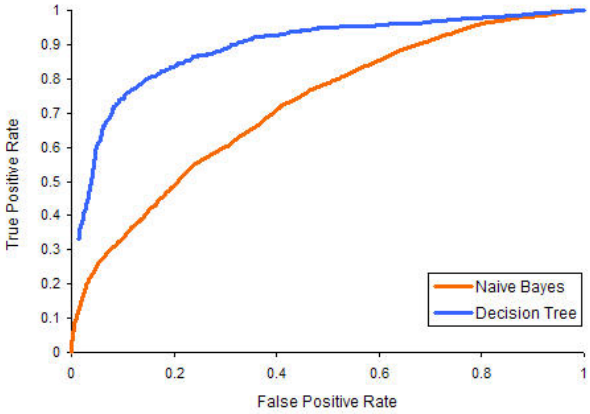


Figure 7: Empathetic Assessment (When) ROC Curves. The ROC curves for each model predicting empathetic behavior triggers in a ten second interval. The area under the naïve Bayes curve is 0.72 and the area under the Decision Tree curve is 0.89.

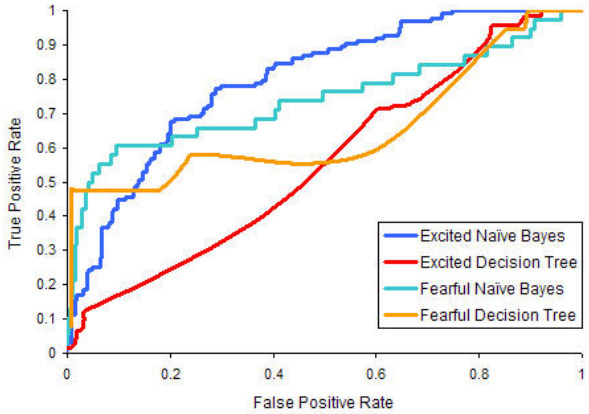


Figure 8: Empathetic Interpretation (How) ROC Curves. The ROC curves represent naïve Bayes and decision tree models for empathetic interpretation for the affective states *excited* and *fearful*. Areas under each curve are as follows: 0.80 (Excited Naïve Bayes), 0.56 (Excited Decision Tree), 0.74 (Fearful Naïve Bayes), and 0.66 (Fearful Decision Tree).

Two categories of functionality can be distinguished. First, the decision tree classifier was best suited for modeling empathy assessment, i.e., it was better able to determine *when* to be empathetic (Figure 7). Second, the naïve Bayes classifier was best suited to modeling empathy interpretation, i.e., it was better able to determine *how* to be empathetic (Figure 8). Although the figure shows only the excited and fearful emotions, all six emotions were evaluated and the naïve Bayes classifier bested the decision tree classifier in every case.

The smoothness of the curve in Figure 7 indicates that sufficient data seems to have been used for training empathy assessment, while the jaggedness of the curve in Figure 8 indicates that more data covering a large space of situations is called for in training empathy interpretation. For example, many empathizers only rarely used particular emotions, e.g., sad, and some trainers suggested that having more affective states available would have been helpful. In general, however, it appears that effective classifiers can indeed be learned for both empathy assessment and empathy interpretation.

6. CONCLUSIONS AND FUTURE WORK

Recent advances in affective reasoning have demonstrated that emotion plays a central role in human cognition and should therefore play an equally important role in synthetic agents. A key affective capability of human social intelligence is empathy. Because empathy is paramount in successful human-human interactions, it would therefore be useful to endow companion agents who are to accompany users in interactive virtual environments with the ability to empathize. Empathy modeling requires accurately assessing a social situation context in order to determine (1) if an empathetic reaction is warranted, and (2) if so, what sort of empathetic behavior should be performed.

This paper has presented a data-driven approach to learning empirically grounded models of empathy from observations of human-human social interactions. In this approach, training data is first generated as a by-product of trainers' interactions with a virtual environment, and models of empathy are induced from the resulting data sets. Critically, the training data employs only observable features, i.e., features that can be directly observed in the environment, so that at runtime, the same features can be used by the empathy models to drive the behavior of companion agents interacting with users. An evaluation of an implemented data-driven empathy modeler suggests that this empirical paradigm offers a promising technique for extending the affective capabilities of synthetic agents. Coupling models of social constructs with expressive controls of agent behavior could perhaps contribute to a new generation of socially and emotionally intelligent synthetic agents in the coming years.

In the future, it will be important to investigate mechanisms for varying empathetic responses in a manner that is most appropriate for individual users, perhaps integrating them with tools such as socio-psychologically validated empathy response instruments. It will also be important to devise integrated methods for employing user physiological response monitored via biofeedback apparati with empirically grounded models of empathy to further extend their range and increase their accuracy. Agent personae offer another direction for future research. For example, females tend to be more empathetic, supported by the fact that a disproportionately large percentage of all empathetic instances in CARE training sessions were performed by female trainers (one-half of all training empathizers were female). Furthermore, exploring models of empathy induced from attributes monitored at varying levels of abstraction may yield models that are transferable between different environments.

7. ACKNOWLEDGMENTS

The authors would like to thank members of the IntelliMedia Center for Intelligent Systems at North Carolina State University for their contributions to the implementation of CARE. Special thanks to Seung Lee for assisting with data analysis. The authors

also wish to thank Valve Software for authorizing the use of their Source™ engine and SDK.

8. REFERENCES

[1] André, E., and Müller, M. E. Learning affective behavior. In *Proceedings of the 10th International Conference on Human-Computer Interaction* (Heraklion, Crete, Greece, June 22-27, 2003). Lawrence Erlbaum, Mahwah, NJ, 2003, 512-516.

[2] Bates, J. *The role of emotion in believable agents.* Technical Report CMU-CS-94-136, Carnegie Mellon University, Pittsburgh, PA, 1994.

[3] Baylor, A. L. The impact of pedagogical agent image on affective outcomes. In *Proceedings of the Workshop on Affective Interactions: Computers in the Loop, International Conference on Intelligent User Interfaces* (San Diego, CA January 9-12, 2005).

[4] Bickmore, T. *Relational Agents: Effecting Change through Human-Computer Relationships*. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2003.

[5] Burleson, W., and Picard, R. W. Affective agents: sustaining motivation to learn through failure and a state of stuck. In *Proceedings of Workshop of Social and Emotional Intelligence in Learning Environments*, in *conjunction with the 7th International Conference on Intelligent Tutoring Systems* (Maceio, Alagoas, Brazil, August 30 – September 3, 2004).

[6] Cavazza, M., Charles, F., and Mead, S. J. Interacting with virtual characters in interactive storytelling. In *Proceedings of the 1st International Conference on Autonomous Agents and Multi-Agent Systems* (Bolonga, Italy, July 15-19, 2002). ACM Press, New York, NY, 2002. 318-325.

[7] Conati, C. Probabilistic assessment of user's emotions in educational games. *Applied Artificial Intelligence*, 16:555-575, 2002.

[8] Davis, M. H. Measuring individual differences in empathy: evidence for a multidimensional approach. *Journal of Personality and Social Psychology*, 44, 113-126, 1983.

[9] Davis, M. H. *Empathy: A Social Psychological Approach.* Brown and Benchmark Publishers, Madison, WI, 1994.

[10] Elliott, C. *The Affective Reasoner: A Process Model of Emotions in a Multi-agent System.* Ph.D. Thesis, Northwestern University, Chicago, IL, 1992.

[11] Elliott, C., Rickel, J., and Lester, J. Lifelike pedagogical agents and affective computing: an exploratory synthesis. *Artificial Intelligence Today*, Lecture Notes in Artificial Intelligence (Sub series of LNCS), Special Volume 1600, Wooldridge, M. and Veloso, M. (eds.), 195-212, Springer-Verlag, Berlin, 1999.

[12] Gratch, J., and Marsella, S. A domain-independent framework for modeling emotion. *Journal of Cognitive Systems Research*, 5(4):269-306, 2004.

[13] Hanley, J.A., and McNeil, B.J. The meaning and use of the area under the receiver operating characteristic (roc) curve. *Radiology* 143:29-36, 1982.

[14] Hoffman, M. L. *Empathy and Moral Development: Implications for Caring and Justice*. Cambridge University Press, Cambridge, UK, 2000.

[15] Ickes, W. *Empathic Accuracy*. New York: Guilford, 1997.

[16] Johnson, W. L., and Rickel, J. Steve: an animated pedagogical agent for procedural training in virtual environments. *SIGART Bulletin* 8:16-21, 1998.

[17] Johnson, W. L., and Rizzo, P. Politeness in tutoring dialogs: "run the factory, that's what I'd do". In *Proceedings of the 7th International Conference on Intelligent Tutoring Systems (*(Maceio, Alagoas, Brazil, August 30 – September 3, 2004). Springer-Verlag, New York, NY, 2004, 67-76.

[18] Lang, P. J. The emotion probe: studies of motivation and attention. *American Psychologist*, 50(5):372-285, 1995.

[19] Lazarus, R. *Emotion and Adaptation*. Oxford University Press, UK, 1991.

[20] Lester, J. C., Towns, S. G., Callaway, C. B., Voerman, J. L., and FitzGerald, P. J. Deictic and emotive communication in animated pedagogical agents. In *Embodied Conversational Agents*, Cassell, J., Prevost, S., Sullivan, J., and Churchill, E. (Eds.), pp. 123-154, MIT Press, Cambridge, 2000.

[21] Marsella, S., and Gratch, J. Modeling coping behavior in virtual humans: don't worry, be happy. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems* (Melbourne, Australia, July 14-18, 2003). ACM Press, New York, NY, 2003. 313-320.

[22] Ortony, A., Clore, G. L., and Collins, A. *The Cognitive Structure of Emotions*. Cambridge University Press, Cambridge, MA, 1988.

[23] Paiva, A., Dias, J., Sobral, D., Aylett, R., Woods, S., Hall, L., and Zoll, C. Learning by feeling: evoking empathy with synthetic characters. *Applied Artificial Intelligence*, 19:235-266, 2005.

[24] Picard, R. *Affective Computing*. MIT Press, Cambridge, MA, 1997.

[25] Porayska-Pomsta, K., and Pain, H. Providing cognitive and affective scaffolding through teaching strategies: applying linguistic politeness to the educational context. In *Proceedings of the 7th International Conference on Intelligent Tutoring Systems (*(Maceio, Alagoas, Brazil, August 30 – September 3, 2004). Springer-Verlag, New York, NY, 2004, 77-86.

[26] Prendinger, H., and Ishizuka, M. The empathic companion: a character-based interface that addresses users' affective states. *Applied Artificial Intelligence*. 19:267-285, 2005.

[27] Swartout, W., Gratch, J., Hill Jr., R. W., Hovy, E., Lindheim, R., Marsella, S., Rickel, J., and Traum, D. Simulation meets Hollywood: integrating graphics, sound, story and character for immersive simulation. In Stock, O., and Zancznaro, M. (eds.), *Multimodal Intelligent Information Presentation*, Kluwer, 2004.

[28] Witten, I., and Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques*. 2nd Edition, Morgan Kaufman, San Francisco, CA, 2005.

Embodied Mobile Agents

Bill Tomlinson
University of California, Irvine
Computer Science Bldg, Room 430A
Irvine, CA 92697
1-949-824-9333
wmt@uci.edu

Man Lok Yau
University of California, Irvine
127B Computer Science Trailer
Irvine, CA 92697
1-949-824-9333
mlyau@uci.edu

Eric Baumer
University of California, Irvine
127B Computer Science Trailer
Irvine, CA 92697
1-949-824-9333
ebaumer@uci.edu

ABSTRACT
The move in many societies toward individuals having multiple networked computational devices – workstations, notebooks computers, mobile phones, PDAs – radically changes the ways in which people engage those devices. However, we lack interaction paradigms that enable a coherent experience across these technologies. One possible approach to this problem involves the use of embodied mobile agents (EMAs), that is, graphically animated, autonomous or semi-autonomous software systems that can migrate seamlessly from one computational device to another. This paper describes an interactive museum exhibit that was implemented with EMAs, discusses the opportunities and challenges presented by this new form of agent, and consider other potential applications for EMAs. While not a universal solution to challenges of interacting with heterogeneous networks of devices, embodied mobile agents can help to provide a coherent user experience across multiple computational devices.

Categories and Subject Descriptors
H.5.1 [**Information Interfaces and Presentation**]: Multimedia information systems – *animation; artificial, augmented, and virtual realities.* I.3.6 [**Computer Graphics**]: Methodology and Techniques – *interaction techniques.*

General Terms
Design, Human Factors.

Keywords
Animated agents, virtual embodiment, mobile agents.

1. INTRODUCTION
Over the past several decades, computational devices have spread rapidly among many human societies. These devices are now sufficiently common that some people have more than one of them, for example owning a workstation, a notebook computer, a PDA, and a mobile phone. These devices are often networked to each other and to the Internet, together providing the platform for an individual's personal information space. However, the varied interaction paradigms that we use when engaging these devices do not always facilitate a coherent experience across multiple devices. In order for these devices to integrate smoothly together and for people to understand their cooperation, new interaction paradigms are needed.

This paper proposes the use of embodied mobile agents[1] (EMAs) – that is, graphically animated, autonomous or semi-autonomous software systems that can migrate seamlessly from one computational device to another – to help enable the development of more intuitive multi-device interaction paradigms. These EMAs may either be animated characters (personified agents designed to elicit social responses from people), animated creatures (biomorphic agents designed to offer some of the capabilities of animals), or animated objects (virtual agents designed to mimic real-world mobile physical objects). EMAs can help to streamline the process of working with several devices simultaneously, to increase the understanding that people have for how information is flowing among their devices, and to help different people collaborate using the capabilities of their multiple computational devices.

Applications of EMAs could reach across a wide variety of domains. In education, EMAs could be used to build multi-device, agent-based participatory simulations that enable people to interact with and learn about a variety of content domains from ecology to economics to the spread of disease. In an industrial setting, EMAs could be used to visualize the flows of information among human collaborators and their computational devices. In entertainment, EMAs could make possible entire new genres of games, with agents jumping quickly between devices carried by players in real space. Other application domains could include social computing systems, healthcare systems, and simulation & training.

To create these kinds of applications, there are a number of problems that need to be solved. How does an agent's embodied form adapt to the capabilities of each device? How can an agent be perceived by people as the same entity as it moves from device to device? How can an animated agent be socially and emotionally engaging across devices? Is it possible for the affordances of the heterogeneous networked devices to facilitate the presence of agents? By addressing these and other related problems, this research project has demonstrated that EMAs can be a robust and deployable technology.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

[1] The term "embodied mobile agents" was chosen over "mobile embodied agents" because of research on the latter term deals primarily with the navigation sense of the word "mobile". Here, the focus is on creating embodied forms of mobile agents in the sense of [12].

Embodied Mobile Agents have the potential to change the way in which people interact with our growing networks of stationary and mobile computational devices. From interactive educational systems to new forms of entertainment to multi-device industrial applications, EMAs could have a significant positive impact on society.

2. RELATED WORK

The EMAs in the two projects discussed in this paper have three main characteristics: they are animated (featuring expressive device-specific graphics), they are interactive (by means of engaging interfaces and autonomous behavior), and they are able to operate seamlessly across multiple devices (such as desktop computers, tablet PCs, PDAs and cell phones).

This work is most similar to two previous projects. The project relates closely to the work of Rekimoto [23], whose Pick-and-Drop system allowed people to drag digital objects from one computer's screen to another. The proposed research builds on this idea, but places the autonomy in the software itself, rather than having the transfer happen only when specifically controlled by a user. The work also draws inspiration from McIntyre *et al.* [18], who developed a system in which an embodied mobile agent could transfer among different workstations, interacting with people via webcams at each workstation. The proposed research is different from this work in two ways: first, the proposed work deals with heterogeneous networks – engaging the challenges that come from operating on different computational platforms – and second, it focuses on the moment of transfer as an important part of the process – the animation, sound, etc. that enables people to see where the agent has come from and where it is going.

Several other research projects have also combined physical elements and virtual agents. Sumi and Mase's AgentSalon [27] is an interactive installation that uses animated agents on large displays and handheld computers to facilitate face-to-face communication. PEACH [26] features agents on handheld computers that provide information to museum visitors. Agent Chameleons [20] described agents that can migrate between robotic and virtual platforms. The project described here is different from AgentSalon, PEACH, and Agent Chameleons as this project places greater emphasis on the animated transfer between devices and on seamless cross-device 3D character animation. Agent Chameleons also does not have a tangible interface.

The development of EMAs is based on a wide range of previous work in autonomous agents and multi-agent systems. Embodied agents (those with a virtual or physical body through which they sense and act on their environment) have also been a topic of much recent research, having been studied in fields including robotics [3] and human-computer interaction [7].

Mobile agents have also been an active area of research and development in recent years, e.g., [22]. Gray *et al*. [12] defined mobile agents as "programs that can move through a network under their own control, migrating from host to host and interacting with other agents and resources on each." (p. 8) Note that the term "mobile agent" is sometimes used to refer to an agent that is able to navigate around its world. While the mobile agents described in this paper are also able to navigate around their worlds, the term "mobile" is used here to denote the interdevice capabilities of the agents, rather than their world navigation capabilities.

Many researchers have explored different aspects of mobile agent technologies, including mobile agents for networked virtual environments [9], mobile agents for network management [28] and a virtual machine for mobile agents [19]. However, non-animated mobile agents are not designed to be engaging for the users in an interactive setting, as the users will only see the result, not the process in between. This engagement is especially important for applications in the domains of entertainment and education. To enhance this engagement, the mobile agents described in this document are embodied as graphically animated entities.

To enable EMAs to animate expressively, this research builds on previous work from both academic and industrial contexts. Academic researchers have looked closely at the topics of expressiveness and believability in autonomous agents, focusing most frequently on interactive virtual humans (e.g., [11]). The computer/video game industry has also expended great effort to make engaging embodied agents. Many character-based games have been exceedingly successful financially, selling millions of copies and generating billions of dollars in sales. Some commercial products have explored the use of virtual characters in handheld devices. Tamagotchi is a handheld device inhabited by a virtual character that owners can play with and feed. Animal Crossing is a computer game that allows the transfer of virtual characters from one virtual environment to another using a memory stick.

Special effects technicians have experimented with breaking the plane of the screen as a way of increasing the believability and impact of the characters and events portrayed. For example, a show at Walt Disney World features several Muppets walking back and forth between different monitors. This project also builds on work in distributed display environments, which investigates the ways that multiple monitors may enhance the user experience, e.g. [13]. Other researchers have explored ways to use distributed display environments to represent virtual environments, e.g. [10]. The project described here seeks to have the same seamlessness in transfers between computing machines as exists in these transfers between displays.

The animation of the EMAs moves beyond this previous work in several ways. This research focuses on the moment of transfer from one device to another as a critical component in the success of EMAs. No previous project has sought to produce expressive real-time animation across the boundary between devices. The interdevice animation in these projects involves the close coordination of autonomous behavior, interactive animation, real-world sensing, interdevice communication, interface design, physical set construction, and graphical and sound effects. This research has explored the details of each of these aspects as it applies to interdevice animation, and the persistence of an agent's identity across devices. This emphasis helps to connect the real and virtual worlds by enabling the EMAs to appear to move through real space. People are accustomed to seeing animated entities on single screens, but the movement of an EMA from one screen to another helps to reduce the inherent separation that usually occurs between the physical world and a virtual environment. Finally, it engages the physical sensing capabilities of the multiple computational devices on which the EMAs operate. The Virtual Raft Project [30] used accelerometers, webcams and IrDA to enhance the believability of EMA animation. EMAs

sense both their virtual environment and aspects of the real world through a blackboard architecture through which entities in the virtual environment exchange information.

The research described here also builds on previous work in autonomous behavior and interface design. In the area of autonomous behavior, Perlin and his colleagues have done pioneering work in creating synthetic actors, e.g., [21]. Cassell and her colleagues built virtual humanoids with the ability to express themselves like real people, in particular in the area of embodied conversational agents, e.g., [5]. Blumberg and his colleagues have contributed in several areas including agent learning [1].

In order to enable EMAs and people to interact with each other, the EMA system features input and output interfaces that connect the two. These interfaces are informed by previous work in engaging and tangible interface design [15]. In the areas of pervasive computing and mobile entertainment, researchers have explored the use of tangible hardware and social computing to enhance user experience in a reality game space [6]. Borovoy and his colleagues developed Meme Tags that allows the transfer of text fragments among mobile devices [2]. Research has been done to investigate ways to enhance the interaction paradigm between a user and a cellular phone [25]. Hand and head gestures have also been suggested as an interface to a mobile interactive paradigm [17].

3. EMA-BASED EXHIBITIONS

Embodied mobile agents have been implemented in two major projects in the past year – the Virtual Raft Project and the EcoRaft Project.

3.1 Virtual Raft Project

The first, the Virtual Raft Project, is an interactive exhibit involving three standard computer workstations and three tablet PCs. The workstations represent "virtual islands" populated by groups of animated humanoid EMAs and the tablet PCs represent "virtual rafts" that participants can carry between the islands (see Figure 1). When a raft is brought up to one of the islands, an EMA can jump onto it. When the participant then carries that raft to a different island, the EMA can jump off the raft onto this new island. A video demonstrating the Virtual Raft Project can be found at:

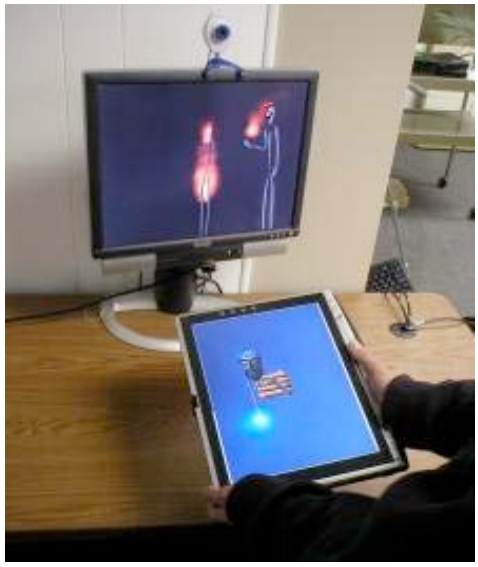


Figure 1: Embodied Mobile Agents may animate seamlessly between a desktop computer and a tablet PC.

http://www.ics.uci.edu/~wmt/movies/VirtualRaftProject.mov

This project enables people to engage physically with EMAs in three main ways. First, the act of moving the tablet PCs between the islands gives people a physical connection to the space and enables them to control the movements of EMAs among the islands. Second, webcams above each of the virtual islands, running a simple background subtraction algorithm, enable the EMAs to react to the presence of people standing in front of that island and respond to their motion. Third, accelerometers in each tablet PC let the EMAs react to the physical motion of the tablets as people carry them, swaying back and forth as the tablets are carried between islands. These three components of the implementation contribute to a sense that the EMAs inhabited the same physical space as people.

The Virtual Raft Project began several key research directions that relate to the proposed work. First, it offered a novel and intuitive interaction paradigm for a person to engage with autonomous animated characters. Second, the project began the exploration of EMAs as viable elements in a multi-user experience. Third, it explored the relevance of EMAs as part of an affective interaction, having a social and emotional impact on people. Fourth, it developed the idea of heterogeneous character animation – that is, animation that enables a real-time animated character to animate seamlessly among two or more heterogeneous devices.

Through four public exhibitions (the ACM CHI 05 conference, the CSCL 05 conference, the Games Learning and Society 05 conference, and the opening of a new building on the campus where it was developed) and numerous smaller scale demonstrations, approximately 800 people have interacted with the Virtual Raft Project to date.

3.2 EcoRaft Project

A second EMA-based project, the EcoRaft Project, was an extension of the Virtual Raft Project developed in collaboration with a team of restoration ecologists and the staff of Discovery Science Center, a science museum in Santa Ana, CA. Instead of featuring humanoid EMAs, this exhibit contained EMAs in the form of animated animal and plant species and was designed to enable 8- to 12-year-old children learn about restoration ecology. The installation consists of three stationary computers that represent virtual islands, and three mobile devices that represent rafts or boxes. Each virtual island represents a different ecosystem. The ecosystems can be populated with hummingbirds, coral trees, and heliconia flowers. Participants can use the mobile devices to transport species from one island to another by bringing a mobile device near one of the stationary computers. One of the virtual islands is the national forest, which has a fully populated ecosystem and can act as the reserve, while the other two virtual islands can be deforested by the press of a button. Participants can repopulate a deforested island by bringing different species in the right order by means of the mobile devices.

In addition to developing further the other themes mentioned above, the EcoRaft Project began to explore the viability of EMAs as tools for education. The Virtual Raft metaphor provides a set of organizing principles for an interactive



Figure 2: A group of children engaged with the Virtual Raft Project.

educational simulation. It distinguishes the operational domain of virtual creatures from that of people, thereby providing a way of explaining how people are meant to interact with the system. It provides inspiration for subject matter that may be effectively conveyed through this kind of simulation. And it provides guidance for the visual, acoustic, tactile, and other elements of the interactive experience. While the platform is not ideal for all topics of education, it does provide a useful framework for exploring topics that deal with groups of people (e.g., social or cultural education), groups of biological organisms (ecology, evolution, etc.), movements between those groups (invasive species, cultural migration, etc.), and potentially other topics (economics, math, languages) as well [29].

The EcoRaft Project was exhibited in the Emerging Technologies program at SIGGRAPH 2005, and was shown to approximately 2000 people over five days in summer 2005. In addition, the project was exhibited publicly at Discovery Science Center on two occasions, and was shown to a group of children in the research lab where it was developed (see Figure 2), with a total of approximately 200 participants.

4. IMPLEMENTATION

The hardware for both the Virtual Raft Project and the EcoRaft Project includes three stationary computers and three tablet PCs. All the computers have 3D graphics cards, infrared (IrDA) sensors, and Ethernet connectivity. Each stationary computer is connected to a webcam and each tablet PC has built-in accelerometers. Both the Virtual Raft Project and EcoRaft Project are written primarily in Java with a module written in C++ to interact with the IrDA sensors. The graphics are displayed using JOGL, the Java version of OpenGL graphics library. A number of factors combined to create an engaging moment of transition for the EMAs – animation, graphical effects, sound, autonomous behavior, networking, interface elements, and interaction context. Following are examples from both of these projects to demonstrate various ways to design engaging EMAs.

Animation is an important element of EMAs, as it represents the embodiment of the agent. It is also one of the distinct features and advantages of EMAs comparing to ordinary mobile agents. One of the challenges is that the animation must look similar on both sides of the transfer, so the users will believe that the character that appears on the source and the destination devices are actually the same character. To contribute to this believability and engagement, the EMAs on different sides of the transfer should also be consistent. The EMA before and after transfer should appear the same and the EMA should appear from the expected direction. In the Virtual Raft Project, the jump animation of a humanoid in the source computer and the landing animation in the destination computer are technically two separate animations and are played in two different machines, but the timing is precise so as to simulate a fluid and seamless transfer between the two devices.

Graphical effects can also augment the animations to make the experience more engaging and life-like. For example, the Virtual Raft includes water ripples near a moving raft, and the hummingbirds in the EcoRaft have iridescent feathers on their breast. However, the level of details of the graphical effects needs to be balanced to maintain the frame rate. For example, since the graphical capabilities of the tablet PCs did not match those of the desktop computers, the iridescence for the hummingbirds was disabled on the tablets to maintain an acceptable frame rate.

In addition, sound plays a significant role in enhancing the believability and engagement of EMAs. Sound serves as a transition indicator to convey a more convincing sense of continuity than animation alone is able to as agents move between platforms. For instance, in the EcoRaft Project when a seed rolls from one device to another, it emits a sound like a slide whistle. This slide whistle sound played an ascending tone from the device of origin followed by a descending tone on the destination device. Similarly, hummingbirds emit a distinctive set of sound effects when flying between devices. Using complementary pairs of sounds emitted from the two devices in sequence supports the illusion that a single agent leaves one platform and then appears on the other. They also provide confirmation for the user that a transfer is occurring. Furthermore, each type of agent has its own signature sound. For example, the each species of hummingbird sings a different tone, with the tones of the four species combining to form a chord. The signature sounds attached to each entity texture the soundscape much as the instrumental signatures of various characters do in Prokofiev's "Peter and the Wolf." Using sound in these various ways strengthens the connection between the embodied agent and its surroundings by giving an agent multiple avenues through which to contribute to its acoustic environment.

People often feel more engaged when they see directly visual feedback [14]. The EMAs in our implementation incorporate certain behaviors in response to human activities. Each stationary computer, which represents a virtual island, has a webcam mounted on top of the display. It allows the EMAs in the virtual world to sense the activities of users in the real world. When the webcam detects human motion, the humanoids in the Virtual Raft get up and approach the virtual camera, and the hummingbirds in the EcoRaft will have a higher probability of flying near the virtual camera. This response from the EMAs gives the impression that they have an awareness of the physical world. Also, the tablet PCs have built-in accelerometers, which can detect the tablet's motion in two axes. Our implementation allows the EMAs to respond to this data, thus reacting to the physical orientation of the tablet. In the Virtual Raft, humanoid EMAs try to balance on the raft; in the EcoRaft, hummingbirds fly to highest point of

the tablet, and seeds will roll inside the box. The added interactivity helps increase the engagement for the users interacting with the exhibit. Furthermore, these examples show ways in which EMAs are able to respond to their physical environment, blurring the line between the physical and virtual worlds.

In the EcoRaft installation, many people felt that they needed guidance to learn the ecological principles in the installation. Since this installation was designed as a science museum exhibit and it is not feasible to have a museum staff member available to demonstrate to the visitors at all times, our research team added a text overlay on the screen showing the status of the ecosystem. An accompanying voiceover also plays when the text overlay changes. This feedback loop allows visitors to try the installation and learn how to interact without needing someone to explain the system to them. It also provides an additional channel for educational content.

The transfer of EMAs from one computer to another computer also involves networking. With other mobile agents, the transfer of an agent involves packaging up the agent's behavior and memory, sending the agent data to the destination computer, and then reassembling the agent on the destination computer when it receives the data. Only one agent exists during the entirety of the transfer. The sequence of transferring an EMA is similar to the transfer of a mobile agent, involving packaging up data, sending to the destination, and reassembling the agent. In the case of an EMA, the transfer must occur during the animation so that the agent appears to transfer seamlessly; otherwise the time that it takes to transfer the agent would interrupt the animation and break the fluidity. The user will see the beginning of a transfer animation on the source device while the agent's data is being sent across. When the destination computer receives the agent data, it starts reassembling the agent and starts playing end of the transfer animation. At this point, two animated agents representing the same character exist in two different computers at the same time, but only one is visible to the user. When the whole animation sequence is complete, the agent in the source computer will be removed from the virtual world and the agent in the destination computer will continue to execute. Timing is very important in this process. If the timing is off, users may see both agents at the same time, or they may need to wait for a long time before the agent appears on the destination computer, thus decreasing the believability.

The tablet PCs have built-in infrared (IrDA) sensors and the stationary computers are connected to external IrDA dongles. The devices use the IrDA sensors to detect proximity and orientation of the other devices and TCP/IP to exchange animated characters. In this implementation, IrDA is used to detect proximity, as it is directional and the range is small. IrDA sensors require line of sight to detect each other and the detection angle is about thirty degrees, which make them good candidates to detect proximity. The sensing range of other technology, such as Bluetooth or GPS, is too broad and not precise enough for this purpose. This ensures that islands will only receive signals from tablet PCs that are in close proximity, in front of the island, and oriented correctly. When the infrared device detects that another device is in range, the two computers establish a network connection (TCP/IP) through Wireless Ethernet (802.11) and use this network connection for data transfer. TCP/IP was chosen for the communication protocol because it is faster than IrDA, especially for transferring all the attributes of the EMAs. If we had used IrDA to transfer the attributes of the EMA, it might have introduced a visible pause during the data transfer and interrupted the flow of the animation. While this method is somewhat more complex, it serves to minimize transfer delays, since wireless Ethernet allows for much faster transfer than IrDA.

5. EVALUATION

In all, approximately 3000 people have interacted with the two EMA implementations. At each exhibition of the projects, the research team observed people interacting with the system in various capacities, ranging from listening to an explanation of the system, to actually using the tablets, to watching people use the tablets, to explaining and demonstrating the system to others. During these exhibitions, groups of participants were successful in achieving basic tasks with EMAs such as distributing them evenly among specified devices with minimal introduction and in short periods of time.

In order to evaluate EMAs' capability to engage participants, the research team analyzed participants' interactions with one of the EMA-based installations described above, the EcoRaft project. A combination of qualitative techniques were employed, including non-interactive observation, participant observation, and semi-structured open-ended interviews [16]. Also, some of Druin's techniques for working with children [8] were employed. Non-interactive observation and participant observation were employed with nearly all of the 3000 participants. Interviews were done with 40 participants – 35 of them interviewed both before and after the interaction, and 5 interviewed only afterward. These interviews were conducted by one interviewer and one notetaker. The exact methodology differed slightly at the three deployment sites for these projects due to the exigencies of the various environments.

The first deployment site was in the Emerging Technologies program at the 2005 ACM SIGGRAPH Conference in Los Angeles, California. The researchers engaged in non-interactive observation; in addition, at SIGGRAPH the researchers conducted semi-structured interviews of one to two participants at a time. During set blocks of time, random participants were approached and asked if they would like to participate in an evaluation of the installation. The process began with a preliminary interview, which was done before participants had a chance to interact with the system. Participants were asked about their initial impressions regarding the EcoRaft exhibit, as well as their demographic information and technological background. A research member then gave a demonstration of the exhibit to the subject while the interviewer and notetaker observed the participants' interaction with the system. After finishing the interaction, the participants returned for a follow-up interview in which they were asked about their general impressions, their understanding of the ecological concept behind the game, the ease or difficulty in comprehending and using the system, their perspective on the realism of the exhibit, and whether or not the interface was natural and intuitive.

The second deployment site was at the Discovery Science Center, a hands-on science museum in Santa Ana, CA. The participants consisted of children, ages 7 to 13, visiting with family members or with day camps and summer schools. In addition to making non-interactive observations, the researchers asked some of the children to describe their experiences in an interview and give their impressions about

the system. Interviews were conducted with children randomly selected from those who interacted with the system for at least three to five minutes. Those children who agreed to participate in the interview were brought, with their parent or guardian accompanying, to a quiet room separate from the exhibit space for the interview. Interview questions were similar to those asked of other participants, as well as questions about whether they worked with other children and what they had learned.

The third setting was in a research lab on the campus of UC Irvine. The subject group consisted of local home-schooled children ages 5 through 9, all of whom were familiar with the Virtual Raft Project. The participants were split into focus groups of 4 to 5 children accompanied by parents or guardians. Before interacting with the system, participants were interviewed to determine their familiarity with rainforest ecology and restoration, as well as general familiarity with technology and what they remembered from the previous interaction. The focus groups were then combined and allowed to interact with the system together as one large group. After the interaction, follow-up interviews were conducted with the same small focus groups. The questions in these interviews were similar to the questions asked of the children at Discovery Science Center.

Researchers' observations and participants' responses indicated that participants of varying ages and levels of technical experience were able to quickly learn how to interact with the system. When asked about the interaction with the EMAs, participants indicated that "it seemed fairly intuitive" and that the interaction "was really easy, it's very accessible, that's definitely a very big plus." One of the most engaging aspects was what happened while the object being transferred was located on the participant's tablet. As described in the implementation section, accelerometers in the tablet PCs allowed virtual objects contained therein to move around as if affected by gravity, transforming these virtual objects into "quasi-physical objects," as one participant put it. According to another, "the ability to actually roll the seed around makes it an actual real object." This "quasi-physicality" made the EMAs seem so much a part of the physical world that many participants, especially the children, tried to interact with them by touching the screens of the tablet or the stationary computers. Also important was the seamless transfer between devices. "I am really impressed with how well images transfer from the tablet right to the screen," said one participant. "Everything seems to jump out." The coordination of animation, sound, networking, and other aspects of the implementation created a physicality that served to increase the engagement level for the large majority of participants.

However, the evaluations demonstrate that not every aspect of the EMAs was as successful at engaging participants. In the EcoRaft, very few participants noticed the effect of the webcams. When asked about them, one participant knew that they had "something to do with the recognition of something [in front of the screen]… it senses that basically there's something trying to interact." However, this individual thought that it was related to sensing the presence of the tablet, not the motion of participants, and had no idea that it affected the behavior of the hummingbirds. With the Virtual Raft, though, far more people noticed the change is the humanoids' behavior when a participant approached; many thought the humanoids were coming over to say hello when participants approached the screen. It could be that people are more willing to attribute that sort of intelligent behavior to humanoids rather than hummingbirds, but it is more likely due to the fact that the change in the humanoids' behavior is very sudden and obvious in comparison with that of the hummingbirds. The important point here is that people's effects on the behavior of EMAs should be obvious, such as that of the accelerometers, and less subtle, such as that of the webcam in EcoRaft.

The evaluations suggest that the EcoRaft is an effective tool in teaching about the process of restoration ecology. One of the core concepts behind the installation is that one cannot simply plant a bunch of seeds and expect them to grow, but rather specific species have to be brought to deforested areas in a specific order [4]. This concept is key to the process of restoration ecology. Many participants, when asked about the concept behind the installation, responded with comments such as "[the process] needs to be [done] in a specific order, that you need to put the tree first before you take the flower over there," or "you can't just put a bird there and expect it to stay without having a plant and the plant needs to be fed by the trees."

These evaluations demonstrate that EMAs are an effective approach to designing engaging autonomous agents. They point to some of the aspects of the design that act to increase that engagement, chiefly animation across heterogeneous devices and ways for the agents to react to the state of the surrounding physical world, as well as some other aspects that do not help as much to increase engagement. Furthermore, these evaluations show that EMAs can be and have been deployed as an effective educational tool, harnessing the increased levels of engagement they provide to teach concepts from restoration ecology.

6. DISCUSSION

This research anticipates the continued expansion of mobile devices into many aspects of life around the world. Many of these devices will have a variety of input and networking technologies such as high-bandwidth connections, global positioning, machine vision, and voice recognition. This is already the case in many places in the world. Software agents will operate on many of these machines, performing complex tasks both autonomously and at the explicit request of users. These agents will operate readily across devices, taking action fluidly on heterogeneous computational systems. The research described in this document examines the process by which these agents may be embodied on heterogeneous networks of devices. As the various aspects of embodied mobile agents begin to be addressed, this area may begin to develop as a platform for educational technology, new forms of computer gaming, and other work-related and social multi-device interactions.

EMAs could have applications in a number of different areas. As has been described above, EMAs could be applied to the area of ecology education by creating a participatory simulation in which several computers serve as virtual habitat niches populated by several EMA species, and mobile devices let children move EMAs from niche to niche. The EcoRaft Project seeks to serve this purpose. In addition, EMAs could be useful in teaching a wide range of complex topics, from economics (the flow of trade) to physiology (the transport of nutrients in the body) to chemistry (the reactions of chemicals as they mix) to global health issues (the transmission of disease). The project seeks to design EMA technologies that

are sufficiently general to be applicable in these and other domains, as well as to provide a specific example of the application of EMAs to education through the EcoRaft Project.

An industrial application could involve a system to support a group of people collaborating on a project, where the flows of information are embodied as animated information objects moving from device to device. For a person on the East Coast, EMA-based information objects coming in from the West Coast could appear on the west side of his or her PDA's screen; similarly EMAs coming from upstairs could drop from the top of a desktop monitor. Size, speed, appearance and behavioral characteristics could provide immediate feedback about the content of the information object, and interactions among EMAs could convey the relationships among them. This kind of contextual information could help each person to understand both the geographical and the conceptual dynamics of the project.

An entertainment application could involve a game in which several humanoid EMAs are jumping rapidly from device to device, and the goal is to catch certain EMAs, while passing off others to one's competitors. This game would be a kind of mixed reality "hot potato" game, incorporating aspects of both virtual world and real world engagement.

These potential applications point to the relevance of EMAs to several domains that currently exist or are not far into the future. In the longer term, EMA research could also help to create entirely new forms of human-machine interactions, where autonomous systems play a more significant role in computation. Currently, many people are skeptical of computational systems with too much autonomy, and there is a focus on interacting "through" computers rather than interacting "with" computers [24]. As computational systems grow more powerful, it may be relevant for paradigms to develop through which people can utilize this potential. EMAs may be of some use in designing systems where people interact effectively with multiple computing devices and autonomous computational entities.

7. FUTURE WORK

With regard to the EcoRaft Project, the research team is working on a number of topics that will enable the permanent installation of the exhibit at Discovery Science Center. Specifically, the tablet PCs need to be mounted on rolling platforms that will prevent them from being dropped by participants and will also store longer-lasting batteries. As an alternative, the research is also being done on implementing a version for smaller, more durable mobile devices such as PDA's, as discussed below. Also, the flat panel displays need to be covered with Plexiglas to protect them from interactions where participants assume that they must be touch screens, and poke them with their fingers or bump them with the tablet PCs. There are numerous robustness challenges beyond the software itself in an installation designed for 250,000 annual visitors.

In addition to designing the first museum exhibit, the team is beginning the development of a network of regionally specific ecological exhibits for science centers and museums around the world. Each exhibit will feature a collection of animal and plant species that are tailored to a specific regional ecosystem, to help visitors learn about the ecological issues of their region. For example, a Minnesota exhibit might focus on the reintroduction of wolves, while other exhibits might address the spread of invasive species such as snakehead fish or American bamboo. These exhibits will be built on the same basic Virtual Raft platform.

The authors are also currently working on the integration of Palm devices into the system, and anticipate a number of opportunities and challenges in this regard. The differences in input, output and networking capabilities will require additional effort to incorporate these devices. For example, the smaller size of the device will entail a variety of modifications to the graphical representations of the EMAs. In addition, the limited memory and processing capacity will also entail the implementation of new subsystems to enable EMAs to scale effectively in both memory footprint and processing needs.

With regard to developing EMAs for broader application, security becomes a significant concern. An autonomous agent capable of moving between devices could easily be seen as a kind of computer virus. There are at least two main potential security problems. First, an agent from one's own device could migrate to someone else's device, carrying with it potentially sensitive information. Second, an agent from an unknown or undesirable source could migrate to one's own device, carrying with it a harmful payload or at least being a new kind of spam. Therefore, having a substantial mechanism for preventing unwanted mobility of agents will be an important part of this project.

In addition to these topics, there are significant areas of future work in the integration of EMAs with existing software systems. For example, how can existing mobile agents be given an animated embodiment so that people can perceive their actions more clearly, and how could existing embodied agents be enhanced by allowing them to cross smoothly between devices? These areas of future work could be relevant to a great number of current agent-based systems.

8. CONCLUSION

This paper has proposed that embodied mobile agents (EMAs) can help to provide a coherent user experience across networks of multiple computational devices. Two interactive projects implemented with EMAs have been demonstrated at several major conferences (ACM CHI, ACM SIGGRAPH, CSCL), a public science center, and a number of other venues, to a total audience of approximately 3000 participants. Evaluations of the EMA system based on these participants' experiences were presented, demonstrating some of the advantages and challenges of EMA-based systems. While EMAs are not a solution to all multi-device interactions that people might wish to have, they could be an effective tool in the effort to create seamless interactions across the heterogeneous networks of devices that people increasingly use for learning, work, and play.

9. ACKNOWLEDGMENTS

The authors would like to acknowledge the support of the Donald Bren School of Information and Computer Science, The California Institute for Telecommunications and Information Technology (Calit2), the Nicholas Foundation, and the Emulex Corporation for their continuing support of this project. The authors also thank several anonymous reviewers for their contributions to the paper.

10. REFERENCES

[1] Blumberg, B., Downie, M., Ivanov, Y., Berlin, M., Johnson, M.P. and Tomlinson, B., Integrated learning for interactive synthetic characters. in Proceedings of the 29th annual conference on Computer graphics and interactive techniques, (San Antonio, Texas, USA, 2002), ACM Press, 417--426.

[2] Borovoy, R., Martin, F., Vemuri, S., Resnick, M., Silverman, B. and Hancock, C., Meme tags and community mirrors: moving from conferences to collaboration. in Proceedings of the 1998 ACM conference on Computer supported cooperative work, (1998), ACM Press, 159--168.

[3] Brooks, R.A., Human Level Cognition in Embodied Robots. in Proceedings of the International Joint Conference on Neural Networks (IJCNN '93), (Nagoya, Japan, 1993), 1079–1084.

[4] Carpenter, F.L., Nichols, J.D., Pratt, R.T. and Young, K.C. Methods of facilitating reforestation of tropical degraded land with the native timber tree, Terminalia amazonia. Forest Ecology and Management, 202. 2004. 281-291.

[5] Cassell, J., Bickmore, T., Billinghurst, M., Campbell, L., Chang, K., Vilhjálmsson, H. and Yan, H., Embodiment in Conversational Interfaces: Rea. in Proceedings of the CHI'99 Conference on Human Factors in Computing Systems, (Pittsburgh, PA, USA, 1999), 520-527.

[6] Cheok, A.D., Yang, X., Ying, Z.Z., Billinghurst, M. and Kato, H. Touch-Space: Mixed Reality Game Space Based on Ubiquitous, Tangible, and Social Computing. Personal Ubiquitous Comput., 6 (5-6). 2002. 430--442.

[7] Dourish, P. Seeking a Foundation for Context-Aware Computing. Human-Computer Interaction, 16 (2, 3 & 4). 2001. 229-241.

[8] Druin, A. Cooperative inquiry: developing new technologies for children with children. Conference on Human Factors in Computing Systems. 1999. 592-599.

[9] Fabre, Y. A framework for mobile-agents embodied in X3D networked virtual environment Proceeding of the eighth international conference on 3D Web technology, ACM Press, Saint Malo, France, 2003.

[10] Ferreira, A.G., Cerqueira, R.F.G., Celes, W. and Gattass, M. Multiple Display Viewing Architecture for Virtual Environments over Heterogeneous Networks ACM SIGGRAPH, ACM Press, 1999, 1-10.

[11] Gratch, J., Rickel, J., André, E., Badler, N., Cassell, J. and Petajan, E. Creating Interactive Virtual Humans: Some Assembly Required IEEE Intelligent Systems, 2002, 54-63.

[12] Gray, R., Kotz, D., Nog, S., Rus, D. and Cybenko, G., Mobile Agents: The Next Generation in Distributed Computing. in Proceedings of the Second Aizu International Symposium on Parallel Algorithms/Architectures Synthesis, (Fukushima, Japan, 1997), IEEE Computer Society Press, 8-24.

[13] Hutchings, D.R., Czerwinski, M., Meyers, B. and Stasko, J. Exploring the Use and Affordances of Multiple Display Environments. Workshop on Ubiquitous Display Environments at UbiComp. 2004.

[14] Hutchins, E.L., Hollan, J.D. and Norman, D.A. Direct Manipulation Interfaces. Human-Computer Interaction, 1 (4). 1985. 311-338.

[15] Ishii, H. and Ullmer, B. Tangible bits: towards seamless interfaces between people, bits and atoms. in Proceedings of the SIGCHI conference on Human factors in computing systems, ACM Press, 1997, 234--241.

[16] Lofland, J. and Lofland, L. Analyzing Social Settings: a guide to qualitative observation and analysis. Wadsworth Publishing Company, Belmont, CA, 1984.

[17] Lumsden, J. and Brewster, S. A paradigm shift: alternative interaction techniques for use with mobile & wearable devices. in Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research, IBM Press, Toronto, Ontario, Canada, 2003, 197-210.

[18] McIntyre, A., Steels, L. and Kaplan, F., Net-mobile embodied agents. in Proceedings of Sony Research Forum, (1999).

[19] Meunier, J., A Virtual Machine for a Functional Mobile Agent Architecture Supporting Distributed Medical Information. in 11th IEEE Computer Based Medical Systems conference, (Stamford, CT, 1999).

[20] O'Hare, G.M.P. and Duffy, B.R., Agent Chameleons: Migration and Mutation within and between Real and Virtual Spaces. in The Society for the Study of Artificial Intelligence and The Simulation of Behavior (AISB 02), (London, England, 2002).

[21] Perlin, K. and Goldberg, A., Improv: A System for Scripting Interactive Actors in Virtual Worlds. in Proceedings of ACM SIGGRAPH 96, (New Orleans, LA, USA, 1996), 205-216.

[22] Peysakhov, M., Artz, D., Sultanik, E. and Regli, W. Network Awareness for Mobile Agents on Ad Hoc Networks. Autonomous Agents & Multi Agent Systems. 2004.

[23] Rekimoto, J. Pick-and-drop: a direct manipulation technique for multiple computer environments. in UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology, ACM Press, 1997, 31--39.

[24] Roush, W. Social Machines. Technology Review, August. 2005.

[25] Sacher, H. and Loudon, G. Uncovering the new wireless interaction paradigm. interactions, 9 (1). 2002. 17-23.

[26] Stock, O. and Zancanaro, M. Intelligent Interactive Information Presentation for Cultural Tourism. International Workshop on Natural, Intelligent and Effective Interaction in Multimodal Dialogue Systems. 2002.

[27] Sumi, Y. and Mase, K., AgentSalon: Facilitating face-to-face knowledge exchange through conversations among personal agents. in International Conference on Autonomous Agents, (Canada, 2001), ACM Press, 393-400.

[28] To, H.H., Krishnaswamy, S. and Srinivasan, B. Mobile agents for network management: when and when not! Proceedings of the 2005 ACM symposium on Applied computing, ACM Press, Santa Fe, New Mexico, 2005.

[29] Tomlinson, B., A Heterogeneous Animated Platform for Educational Participatory Simulations. in Computer Supported Collaborative Learning (CSCL), (Taipei, Taiwan, 2005).

[30] Tomlinson, B., Yau, M.L., O'Connell, J., Williams, K. and Yamaoka, S., The Virtual Raft Project: A Mobile Interface for Interacting with Communities of Autonomous Characters. in Conference Abstracts and Applications, ACM Conference On Human Factors In Computing Systems (CHI 2005), (Portland, OR, 2005).

U-DIRECTOR: A Decision-Theoretic Narrative Planning Architecture for Storytelling Environments

Bradford W. Mott
Department of Computer Science
North Carolina State University
Raleigh, NC 27695 USA
bwmott@csc.ncsu.edu

James C. Lester
Department of Computer Science
North Carolina State University
Raleigh, NC 27695 USA
lester@csc.ncsu.edu

ABSTRACT
Recent years have seen significant growth in work on interactive storytelling environments. A key challenge posed by these environments is narrative planning, in which a *director agent* orchestrates all of the events in a storyworld to create an optimal experience for a user, who is herself an active participant in the unfolding story. To create effective stories, the director agent must cope with the task's inherent uncertainty, including uncertainty about the user's intentions and the absence of a complete theory of narrative. Director agents must be efficient so they can operate in real time. In this paper, we present U-DIRECTOR, a decision-theoretic narrative planning architecture that dynamically models narrative objectives (e.g., plot progress, narrative flow), storyworld state (e.g., plot focus), and user state (e.g., goals, beliefs) with a dynamic decision network that continually selects storyworld actions to maximize *narrative utility* on an ongoing basis. The U-DIRECTOR architecture has been implemented in a narrative planner for Crystal Island, an interactive storyworld in which users play the role of a medical detective solving a science mystery. Preliminary evaluations suggest that the U-DIRECTOR architecture satisfies the real-time constraints of interactive environments and creates engaging narrative experiences.

Categories and Subject Descriptors
H.5.1 [**Multimedia Information Systems**]: Artificial, augmented, and virtual realities.

General Terms
Algorithms, Design.

Keywords
Interactive Narrative, Synthetic Agents.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

1. INTRODUCTION
Narrative plays a central role in communication and cognition. With the demand for increasingly sophisticated entertainment and education applications, recent years have witnessed significant growth in research on interactive storytelling environments that create engaging narrative experiences [21, 22, 28, 33]. A critical component of this enterprise is devising effective computational models of narrative [1, 9, 20, 26, 30, 35]. These models support *director agents* that dynamically direct a cast of believable virtual characters in rich 3D storyworlds with coherent narrative structures that play out interactively in real time.

Director agents for interactive storytelling environments operate on at least two distinct but interacting levels: They craft the global story arc, typically by traversing a *plot graph* [2] that encodes a partial order of significant events in a story, and they plan the behaviors of the virtual characters and physical events in the world [3, 19]. To create an engaging experience for the user, director agents must carefully balance character believability and plot coherence [30], cope with deviations from a previously devised narrative structure caused by user actions [20, 29], provide fine-grained character control (both dialogue and actions) [22], all the while creating stories that obey the author's aesthetic, perhaps represented as an evaluation function that guides a search through a plot graph [2, 26, 35]. Director agents must also interface with believable agent functionalities supporting the story's virtual characters [14, 19, 32].

A key challenge posed by narrative planning is coping with the multiple sources of uncertainty inherent in the task. First, it is difficult to precisely infer users' goals, beliefs, and experiential attributes. It is difficult to accurately predict what effects changes in the storyworld may have on the user. Because it is important that users believe that their actions affect the ongoing story [24], predicting the effects of narrative planning actions is further complicated by the fact that users are active participants in the narrative. Second, we do not have—perhaps we *cannot* have—a complete theory of interactive narrative. We do not even have well defined theories of narrative for specific genres or particular storyworlds. Third, despite the episodic nature of narrative, however inaccurate our predictions might be over very short periods, they become even less accurate as we attempt to predict effects farther into the future.

The high degree of uncertainty, together with the multiple factors affecting narrative planning, call for a principled decision-making framework that enables director agents to rationally choose among candidate storyworld actions. The framework

must not only be able to effectively cope with uncertainty, it must be able to exploit the evidence available about the user and storyworld state to achieve the (possibly competing) narrative objectives, and it must be able to meet the real-time performance requirements of interactive environments.

In this paper we present U-DIRECTOR,[1] a decision-theoretic narrative planning architecture that uses a dynamic decision network [11] to model the narrative objectives, storyworld state, and user state. Rather than adopting an *ad hoc* approach to reasoning about the multiple sources of evidence, at each tick of the clock, U-DIRECTOR systematically evaluates the available evidence, updates its beliefs, and selects the storyworld action that maximizes expected narrative utility. The U-DIRECTOR architecture has been implemented in a narrative planner for Crystal Island, an interactive storyworld in which users play the role of a medical detective solving a science mystery. Preliminary evaluations suggest that the U-DIRECTOR architecture satisfies the real-time constraints of interactive environments and creates engaging narrative experiences.

2. NARRATIVE PLANNING

Narrative experiences are powerful, and there is growing awareness of narrative's foundational role in psychology [7], cognitive models of reading comprehension [13], and film theory [5]. In Gerrig's seminal work on comprehension [13], he identifies two properties that readers experience. First, they are *transported*, i.e., they are somehow taken to another place and time in a manner that is so compelling it seems real. Second, they *perform* the narrative. Like actors in a play, they actively draw inferences and experience emotions as if the experiences were somehow real. Because narrative is compelling on many levels, interactive applications in entertainment, education, and training increasingly leverage narrative to create engaging experiences through rich virtual storyworld environments.

Early interactive narrative-based media were either partially or entirely scripted. All events were either completely linear or were ordered in a pre-determined branching structure [6]. However, the simplicity of the tree-like representations severely limited the level of interactivity that users could experience, and the combinatorics of highly interactive storyworlds could not be accommodated. Interestingly, the pre-scripted tradition lives on and thrives today; it is the approach used by most educational software and video games.

Dynamic narrative generation, in which stories are created on the fly (perhaps in response to users' actions), offers a promising alternative to pre-defined branching structures. Work on text-based dynamic narrative generation has yielded computational models of narrative that manipulate characters' goals and actions [23], reason about the "virtual author's" goals [18, 34], and generate extended natural language prose for narratives [8]. Dynamic narrative generation is particularly well suited to interactive storytelling environments. If narratives can be created incrementally in response to users' actions, then users can (legitimately) be made to believe that they are empowered as active participants in the story, thereby increasing their sense of agency and immersion [24].

Narrative planning for interactive storytelling environments has been the subject of increasing interest [21, 22, 28, 33]. Recent work on narrative planning has investigated a broad range of issues for interactive story environments. The narrative community has devised techniques for tightly-coupled plot creation and character behavior in dialogue-oriented interactive stories [22], search paradigms for encoding author aesthetics with an evaluation function [26, 35], and monitoring users' actions to determine if they are threatening the plot and, if so, either accommodating the new development or intervening [20, 29]. Work on emergent narratives, in which highly believable synthetic agents are given initial goals in a simulated world, has been particularly active [1, 9]. For example, a promising technique for creating emergent narratives is providing virtual characters with hierarchical task networks to drive their interactions with one another and the user [9]. Recognizing the "authoring bottleneck," some have explored efficient techniques for designing virtual characters [19, 32].

3. DECISION-THEORETIC NARRATIVE PLANNING

A key challenge posed by narrative planning is coping with the significant uncertainty associated with the task. First, narrative planning must deal with unobservable aspects of the user. These include her beliefs about the storyworld, her goals, and her experiential state, such as her level of engagement. Being able to effectively reason about these user characteristics is essential for proper user-story mediation. For example, it is important to have an accurate picture of the user's state to determine when and how to intervene or to accommodate the user's actions [15, 20, 29]. Second, we do not have available to us a formally represented theory of interactive narrative that supports sound and complete inference about story construction and its impact on the user. Further complicating the problems posed by uncertainty is the multitude of factors that bear on narrative decision-making activities. For example, narrative objectives such as ensuring plot progress and maintaining narrative coherence are affected by factors associated with the user's state as well as activities in the storyworld, e.g., character behaviors. Finally, narrative planning must weight all of these factors as it drives towards the goal of creating the best possible narrative experience at each juncture of the unfolding story. In short, it should satisfy the following requirement:

> ***Narrative Rationality****:* Reasoning in a principled manner about narrative objectives, storyworld state, and user state, each with its own associated uncertainty, in the absence of a complete theory of interactive narrative, to rationally select actions that maximize expected narrative utility.

It is important to note that narrative rationality must be realized in real time to accommodate the demands of interactivity.

To address these requirements, we introduce U-DIRECTOR, a decision-theoretic *director agent* architecture that uses a dynamic decision network to achieve narrative rationality. Inspired by innovative decision-theoretic approaches to inference in intelligent tutoring systems [10, 25], U-DIRECTOR explicitly models the uncertainty in narrative objectives, storyworld state, and user state. In each decision-making cycle, it systematically evaluates the available evidence, updates its beliefs, and selects the storyworld action that maximizes expected narrative utility.

[1] U-DIRECTOR: Utility-based Director Agent

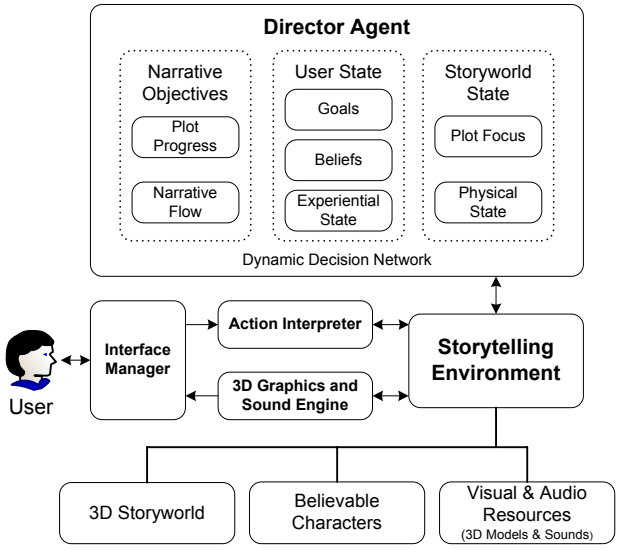


Figure 1. General Architecture.

3.1 Architecture

The U-DIRECTOR architecture is shown in Figure 1. All user activities in a U-DIRECTOR storytelling environment are mediated by the interface manager, which provides interaction and rendering functionalities. The storytelling environment, which drives the rendering and sound engines, employs three sets of resources: a representation of the 3D storyworld, a cast of semi-autonomous characters, and media libraries housing 3D models and sound effects. So that the director agent may carry out its primary tasks of plot creation and character behavior control, it monitors the user's activities and the actions of the semi-autonomous characters in the storytelling environment to craft the narrative.

The director agent has access to three principle knowledge sources: narrative objectives, storyworld state, and user state. The director agent continuously monitors and seeks to achieve the *narrative objectives* to ensure that steady progress is made through the plot (plot progress) and that global coherence is maintained (narrative flow) so that the story does not seem disjoint as a result of non-motivated actions occurring. Its *storyworld state* knowledge includes information about which plot points are currently active (plot focus), as well as basic knowledge of the storyworld and characters (physical state), e.g., the characters' goals, behaviors, and current locations. Its *user state* represents knowledge about the user's current goals, her beliefs about the storyworld, and experiential attributes, such as engagement, which are inferred from her actions in the storytelling environment. Details of knowledge source representation are discussed in Section 3.3. To cope with the uncertainty in narrative planning, the three sets of knowledge sources are integrated into a dynamic decision network (DDN) maintained by the director agent. The director agent evaluates the DDN to solve the decision problem in each cycle to select the next narrative action.

3.2 Dynamic Decision Networks

Dynamic decision networks (DDNs) extend decision networks, which in turn extend Bayesian networks. *Bayesian networks* [27] are composed of chance nodes with their associated conditional probabilities and influence arcs that collectively form a directed acyclic graph. Bayesian networks provide a compact representation of the full joint probability distribution and allow inferences to be made about any attribute within the network once priors, conditional probabilities, and available evidence have been specified.

Decision networks [16], also known as *influence diagrams*, extend Bayesian networks to provide a mechanism for making rational decisions by combining probability and utility theory. In decision networks, in addition to chance nodes, the network contains utility and decision nodes. The decision nodes represent the choices of the decision-maker while utility nodes model the decision-maker's preferences. In a decision cycle, a decision theoretic agent chooses the action with the maximum expected utility.

Dynamic decision networks [11] provide a principled approach for agents to make rational decisions in the face of uncertainty within changing environments. To cope with time varying attributes, DDNs maintain a series of time slices to represent attributes at successive moments in time. An arc connecting an attribute in a previous time slice to an attribute in a later time slice encodes an influence on the attribute's value from the previous attribute value. Dynamic decision networks provide a useful framework for modeling beliefs about the world, associating preferences with states of the world, and making decisions.

3.3 Director Agent DDNs

Narrative is fundamentally a time-based phenomenon. Director agents must therefore take into account the narrative history and be able to as accurately as possible predict (1) the effects of candidate actions on the user and (2) the effects of user's actions on possible future courses of the narrative. Therefore, in each decision cycle, U-DIRECTOR considers candidate narrative actions to project forward in time the effects of the actions being taken and their consequent effects on the user. To do so, it evaluates its narrative objectives in light of the current storyworld state and user state. Each decision cycle considers three distinct time slices (*narrative state*$_t$, *narrative state*$_{t+1}$, and *narrative state*$_{t+2}$), each of which consists of interconnected sub-networks containing chance nodes in the DDN (Figure 2). The three slices represent (1) the current narrative state, (2) the narrative state after the director agent's decision, and (3) the narrative state after the user's next action. The DDN's *director action* is a decision node, the DDN's *user action* is a chance node, and *utility*$_{t+2}$ is a utility node in the DDN. Each time slice encodes a probabilistic representation of the director's beliefs about the overall state of the narrative, represented with the following knowledge sources:

- *Plot Progress*: Models the storyworld's plot graph, a representation of temporal relations that hold between storyworld events; identifies which elements in the plot graph are waiting, ready, or completed.
- *Narrative Flow*: Models thought flow (coherence of actions to support a particular goal, e.g., searching a room after being asked to do so) and location flow (coherence of actions within spatial constraints of the storyworld, e.g., discovering multiple physical clues within the same room, one after another) associated with plot point completion; thought flow and location flow are both represented by annotations on

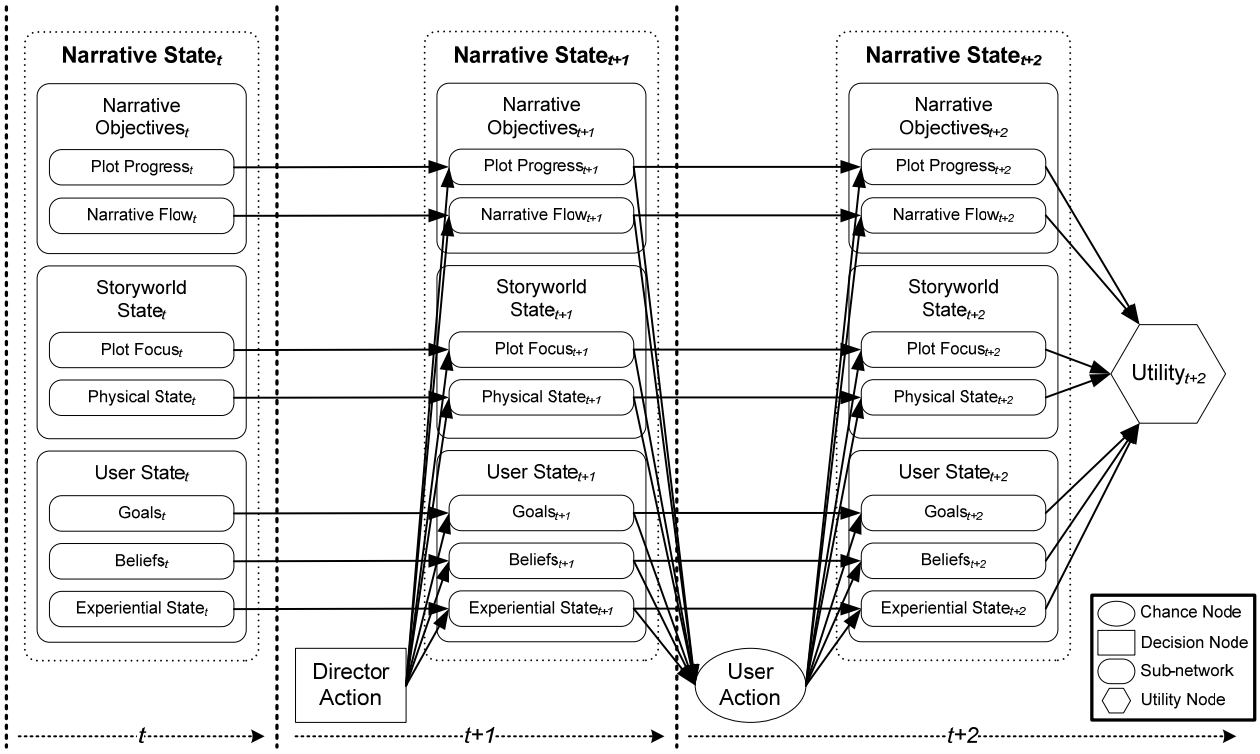


Figure 2. Director Agent Dynamic Decision Network.

user actions, indicating the relative importance for particular plot points.

- *Plot Focus*: Models the plot points that are currently active to which (it appears to the director agent) the user is currently attending.
- *Physical State*: Models the current location of the user and characters in the storyworld, and models the user's activity level as indicated by her character's interactions with objects and characters in the storyworld.
- *User Goals and Beliefs*: Models the user's knowledge about the salient facts of the story that she has learned through interactions with the environment and other characters, as well as her plot progress and focus (see above).
- *User Experiential State*: Models the user's independence (as indicated by how much manipulation the director agent has performed), her engagement (as indicated by how active she is in the environment – see physical state above), and her excitement (as indicated by changes in her knowledge about the facts of the world and pacing information).

Representations of each of these knowledge sources are integrated directly into U-DIRECTOR's DDN. It is important to note that each knowledge source is itself in fact a sub-network encoding many beliefs about particular components of the narrative state. For example, Figure 3 depicts a portion of a plot focus network and its influences by a plot progress network. Nodes in the plot progress network are used to predict likely user actions as they make their way through the narrative. The director agent takes them into account during each decision-making cycle as described below.

Computation begins by considering the current beliefs about the narrative objectives, storyworld state, and user state, represented in *narrative state*$_t$. U-DIRECTOR models candidate director actions and how they influence the story using links from the director action to *narrative state*$_{t+1}$. Example director actions include providing various levels of hints to the user to guide them through the plot and instructing characters to perform actions that the user is either neglecting or does not seem to be capable of performing. To constrain the number of candidate actions to evaluate, U-DIRECTOR models abstract director actions [26, 35] so that it does not in this step have to attend to the plethora of concrete storyworld actions. Next, it models how the user's actions depend on the possible worlds encoded in *narrative state*$_{t+1}$. In turn, it models how possible user actions influence the story in *narrative state*$_{t+2}$.

Finally, the director's preferences over potential narrative states are modeled with links from *narrative state*$_{t+2}$ to the utility node *utility*$_{t+2}$. Preferences provide a representation in which authors specify the relative importance of salient features of the narrative state. Narrative utility serves a similar function in the director agent's DDN as evaluation function serve in search-based narrative planning [35]. For example, the importance of "location flow" [26] can be appropriately weighted to suit an author's aesthetic when the DDN is constructed.

Once the director agent has fully updated the network, it selects the director action that maximizes the expected narrative utility, waits to see what action the user takes (if any) and updates its beliefs as necessary. It then begins the cycle over again, performing a "rollup" operation (also known as "filtering"), which usually involves the use of approximation techniques [4], to reduce the number of slices needed in memory at a given moment in time.

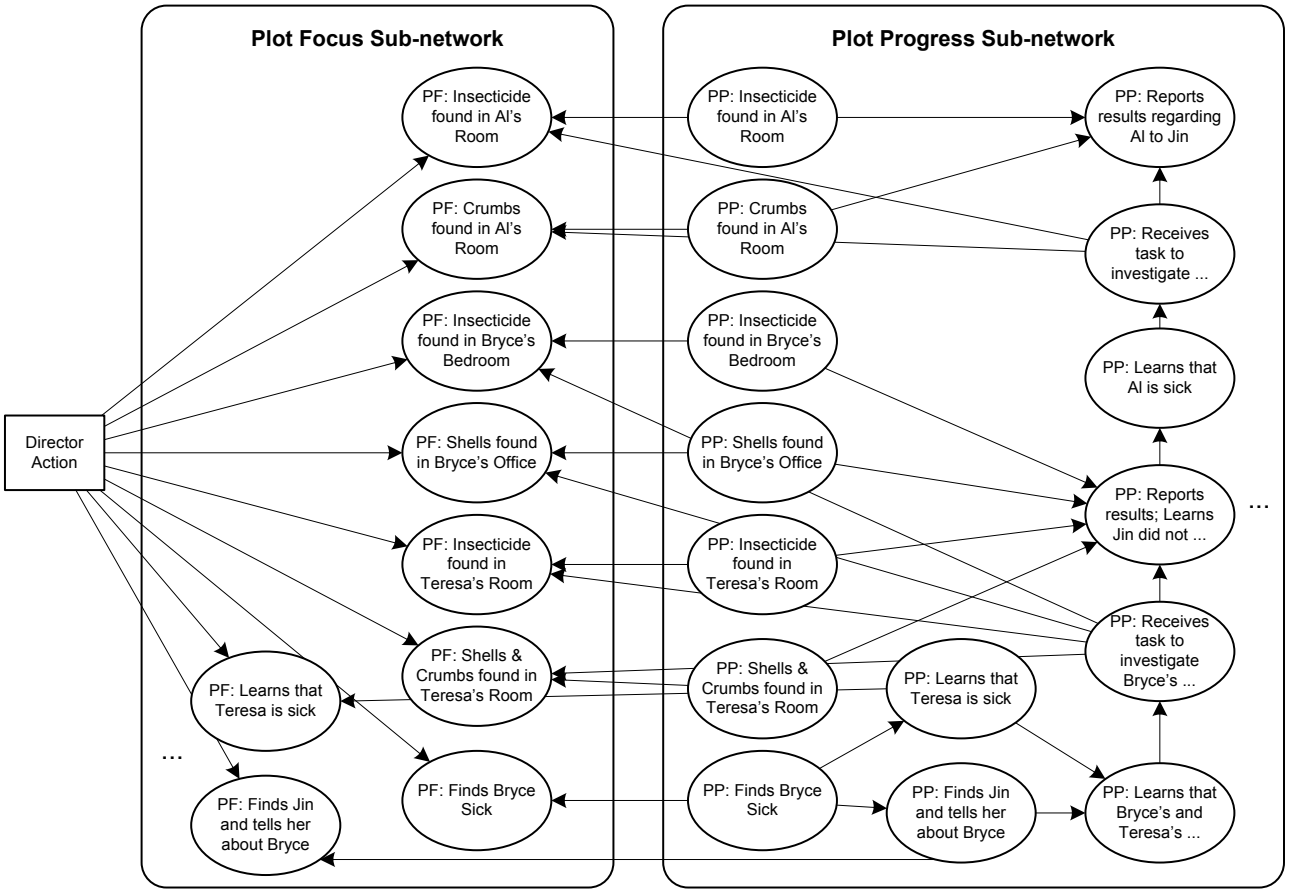


Figure 3. Selected Elements of the Plot Focus and Plot Progress Sub-networks.

4. EXAMPLE DOMAIN

The U-DIRECTOR architecture has been implemented and evaluated in a narrative planner for Crystal Island, an interactive storyworld in which users play the role of a medical detective solving a science mystery. After introducing the Crystal Island storyworld, we describe the implementation and present a sample interaction illustrating U-DIRECTOR's behavior.

4.1 Crystal Island Storyworld

Crystal Island is a prototype interactive storytelling environment featuring a science mystery set on a recently discovered volcanic island where a research station has been established to study the unique flora and fauna. The user plays the protagonist attempting to discover the origins of an unidentified outbreak at the research station. The story opens by introducing her to the island and the members of the research team for which her father serves as the lead scientist. She is immediately confronted with her father's sudden paralysis (Figure 4(a)). As other members of the team fall ill, it is her task to discover the cause of the outbreak as quickly as possible. She is free to explore the world to collect physical evidence and interact with other characters (Figure 4(b)). She must quickly gather enough evidence to correctly choose among candidate diagnoses including botulism, Guillain-Barré syndrome (a disorder caused by nerve inflammation), paralytic shellfish poisoning, stroke, and tick paralysis to solve the mystery and save the research group.

4.2 Implementation

U-DIRECTOR and the Crystal Island storytelling environment are implemented with a dynamic decision network modeled with an efficient Bayesian inference engine—a discussion of runtime performance follows in Section 5—and a high-performance 3D game platform. The director agent consists of a dynamic decision network containing approximately 200 chance nodes connected by over 400 links representing the narrative objectives, storyworld state, and user state. Collectively, the sub-networks explicitly encode over 7,000 conditional probabilities.

The director agent's narrative utility preferences are represented in a separate structured utility network with over 50 utility nodes. The dynamic decision network for Crystal Island is modeled and implemented with the SMILE Bayesian inference engine [12], while the storytelling environment, semi-autonomous characters, and presentation layer are implemented with Valve Software's Source™ engine, the 3D game platform for Half-Life 2.

4.3 Example Scenario

To illustrate U-DIRECTOR's behavior, consider the following situation as a story unfolds on Crystal Island. The user's character awakens from a good night's sleep and the adventure begins. At this juncture, several elements in the plot graph are available for her to address. For example, she could find her father paralyzed in his bedroom, she might notice that a plate with

leftover food is in his office, or she might discover other facts such as that her good friend, Teresa, has also been stricken with the mysterious illness.

Although many possible actions may be taken by the user, it is the task of the director agent to determine if there is any action it might select to ensure that she is making steady progress through the plot. As the director agent assesses the situation, it notices that (1) the user is currently exhibiting high independence (since the director agent has not yet had to intervene in the story), (2) the user has not discovered any facts about the case, (3) the user is currently located in her father's house, and (4) her father is in his room. These observations, combined with a number of other factors (e.g., the fact that the user is somewhat active in the world), lead the director to make the decision to provide a hint that might lead her to find her father sick in his room since this action yields good location flow.

Once this decision is made, the system can realize the hint in many ways. For example, if the user were nearby, her father might make a moaning noise; however, if she were farther away, he might blink the lights in his room or have another character contact his daughter. The director agent then awaits the user's reaction to see how the events play out. In this case, the user hears her father's call for help, walks to his room, learns of his illness, and quickly departs to find the camp nurse. Observing that the user has found her father sick, the director agent updates its plot graph by marking the corresponding node complete and marking subsequent nodes ready for execution.

At this point the director agent must once again decide if there is anything it should do. As it notices that there are a number of plot graph elements (such as finding the partially eaten plate of food) which could be executed at the user's current location (i.e., to maintain location flow), it also takes into account the fact that these plot elements do not coincide with the user's current train of thought, as evidenced by the fact that she should be attempting to locate the camp nurse. It combines this evidence with the fact that it recently presented a hint and decides to take no action at this juncture and await the user's next action.

After some time has passed and the user has made limited progress toward completing the available plot graph elements, the director agent decides to provide a hint to the user to get her back on track. After reviewing all of the available evidence, the director agent decides that the user should be attempting to find the camp nurse and therefore directs the camp nurse to come to the user. The mystery continues with the director agent continuously monitoring the unfolding situation as the user rules out all but one of the possible diagnoses. The user solves the mystery by determining that the source of all the illnesses was botulism from last night's dinner and the research team is saved.

While the artifacts and character behaviors in the 3D world of Crystal Island and their connections to the director agent's network are the subject of continued work, the narrative reasoning described in this scenario is fully implemented. Each of the knowledge sources in the architecture (plot progress, narrative flow, plot focus, physical state, user goals, user beliefs, and user experiential state) are fully functioning and currently support this and many other scenarios.

5. EVALUATION

Two key aspects of narrative planning that must be addressed in a viable solution are tractability and effectiveness. Section 5.1 discusses a performance analysis of the U-DIRECTOR implementation for Crystal Island, and Section 5.2 examines U-DIRECTOR's decision-making.



(a)



(b)

Figure 4. The Crystal Island Storyworld.

5.1 Performance Analysis

Real-time performance of narrative planning is critical for interactive storytelling environments. While probabilistic reasoning has much to offer, tractability in probabilistic reasoning is always a challenge. Because exact inference in Bayesian networks is known to be extraordinarily inefficient (in the worst case *NP*-hard), U-DIRECTOR exploits recent advances in approximate Bayesian inference via stochastic sampling. The accuracy of these methods depends on the number of samples used. Moreover, stochastic sampling methods typically have an "anytime" property which is particularly attractive for real-time applications.

Considering the activity level in Crystal Island and the level of abstraction at which directions are modeled, it was determined that the director agent should be able to perform a complete network update at least five times per minute. To ensure that the response time of the director agent was adequate, a performance analysis was conducted to measure the network update time using an exact Bayesian inference algorithm (*Clustering* [17]) and two

approximate Bayesian inference algorithms (*EPIS-BN* [36] and *Likelihood weighting* [31]).

Results of updating the director agent's network using the clustering, EPIS-BN, and likelihood weighting algorithms implemented in SMILE are presented in Table 1. These results were obtained on a 2.4 GHz AMD Mobile Athlon 64 PC with 1 GB of RAM running Windows XP.

Table 1. Response times for Bayesian inference algorithms

	Samples	Mean (seconds)	Standard Deviation
Clustering	N/A	18.41	8.48
EPIS-BN	1,000	2.32	0.18
	5,000	6.02	0.12
	10,000	10.70	0.16
Likelihood	1,000	1.59	0.19
	5,000	7.31	0.06
	10,000	14.65	0.03

Results of the study indicate that the clustering algorithm's running time has the greatest variability and may not satisfy the performance requirements of interactive narrative. EPIS-BN appears to provide acceptable response times while utilizing larger sample sizes than likelihood weighting and therefore yielding better approximations; however, the errors introduced by these approximation techniques can be problematic. Future work involves exploring alternative approximation techniques (e.g., [4]).

5.2 Simulated User Experiments

While the quality of narrative is inherently subjective, it is nonetheless important to attempt to gauge the overall effectiveness of a director agent's decision-making activities. To systematically investigate U-DIRECTOR's narrative planning, two families of simulated users were created. Simulated c*ooperative* users followed the director agent's guidance 60% of the time for hints and 80% of the time for strong suggestions. In contrast, simulated *uncooperative* users followed the director agent's guidance 10% of the time for hints and 25% of the time for strong suggestions. Six simulated users (3 cooperative and 3 uncooperative) interacted with Crystal Island director agent to create six different narrative experiences.

Analyses of the traces indicate that the director agent took appropriate action to guide the user through the narrative. As was desired, the director agent tended to adopt a hint-centered approach for more cooperative users and was more heavy-handed with users who were less cooperative. The simulated user approach appears to offer a promising means for establishing baseline performance prior to conducting extensive focus group studies with human users.

6. CONCLUSION AND FUTURE WORK

Decision-theoretic narrative planning offers a unified approach to dynamically guiding narratives in a storytelling environment. This paper describes a decision-theoretic director agent that has been implemented in a narrative planner for an interactive storytelling environment in which the user plays the role of a medical detective. By providing a principled approach to coping with the intrinsic uncertainty of interactive narrative, decision-theoretic director agents can systematically draw inferences about the broad range of factors affecting an unfolding story. It can thereby reason about narrative objectives, the state of the storyworld, and the state of the user to direct the course of the plot and the behaviors of the semi-autonomous characters, all with the result of creating engaging interactive narrative experiences.

Three areas of future work are particularly intriguing. First, in the current approach to constructing DDNs, devising networks by hand is very labor intensive. For example, the DDNs for the prototype interactive storytelling environment described in this paper required several person-weeks of effort. The prospect of learning the conditional probabilities in the DDNs holds much appeal.

Second, developing pedagogically oriented narrative environments that support education via narrative-based learning experiences is a promising direction for future work. However, integrating pedagogical goals into the narrative planner undoubtedly poses significant challenges because the resulting system would need to reason about (possibly competing) pedagogical and narrative goals. Nonetheless, utility-based computing could contribute to balancing these goals.

Finally, affect plays an important role in narrative. However, the current director agent includes only an impoverished representation of the user's affective state. Introducing more sophisticated techniques for affective reasoning could significantly extend the director agent's ability to accurately assess the user's engagement and motivation, which could directly contribute to its ability to create more engaging experiences.

7. ACKNOWLEDGMENTS

The authors wish to thank the members of the IntelliMedia Center for Intelligent Systems at NC State University for their assistance in implementing U-DIRECTOR, the Decision Systems Laboratory of the University of Pittsburgh for access to the SMILE Bayesian inference engine, Omer Sturlovich and Pavel Turzo for use of their 3D model libraries, and Valve Software for access to the Source™ engine and SDK. Thanks also to Michael Young and the members of the Liquid Narrative Group at NC State University for insightful discussions on narrative.

8. REFERENCES

[1] Aylett, R. Narrative in virtual environments: towards emergent narrative. *Working notes of the Narrative Intelligence Symposium*, AAAI Spring Symposium Series. Menlo Park: AAAI Press, 1999.

[2] Bates, J. Virtual reality, art, and entertainment. *Presence: The Journal of Teleoperators and Virtual Environments* 1(1):133-138, 1992.

[3] Blumberg, B. and Galyean, T. Multi-level direction of autonomous creatures for real-time virtual environments. In *Proceedings of SIGGRAPH '95*, 47-54, 1995.

[4] Boyen, X. and Koller, D. Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, 33-42, 1998.

[5] Branigan, E. *Narrative Comprehension and Film*. London: Routledge, 1992.

[6] Bruckman, A. The combinatorics of storytelling: Mystery train revisited. Unpublished manuscript, 1990.

[7] Bruner, J. *Acts of Meaning*. Cambridge, MA: Harvard University Press, 1990.

[8] Callaway, C. and Lester, J. Narrative prose generation. *Artificial Intelligence*, 139(2), 213-252, 2002.

[9] Cavazza, M., Charles, F., and Mead, S. Interacting with virtual characters in interactive storytelling. In *Proceedings of the First International Joint Conferences on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, 2002.

[10] Conati, C. Probabilistic assessment of user's emotions in educational games. *Journal of Applied Artificial Intelligence*, Special Issue on Merging Cognition and Affect in HCI, 16(7-8), 555-575, 2002.

[11] Dean, T. and Kanazawa, K. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3), 142-150, 1989.

[12] Druzdzel, M. SMILE: structural modeling, inference, and learning engine and GeNIE: a development environment for graphical decision-theoretic models. In *Proceedings of the Eleventh Conference on Innovative Applications of Artificial Intelligence*, 902-903, 1999.

[13] Gerrig, R. *Experiencing Narrative Worlds: On the Psychological Activities of Reading*. New Haven: Yale University Press, 1993.

[14] Gratch, J. and Marsella, S. A domain-independent framework for modeling emotion. *Journal of Cognitive Systems Research*, 5(4), 2004.

[15] Harris, J. and Young, M. Proactive mediation in plan-based narrative environments. In *Proceedings of the International Conference on Intelligent Virtual Agents*, 2005.

[16] Howard, R. and Matheson, J. Influence diagrams. In R. Howard and J. Matheson (Eds.), *Readings on the Principles and Applications of Decision Analysis*, 721-762. Menlo Park: Strategic Decisions Group, 1984.

[17] Huang, C. and Darwiche, A. Inference in belief networks: a procedural guide. *International Journal of Approximate Reasoning*, 15(3), 225-263, 1996.

[18] Lebowitz, M. Story-telling as planning and learning. *Poetics*, 14, 483-502, 1985.

[19] Loyall, B., Reilly, S., Bates, J., and Weyhrauch, P. System for authoring highly interactive, personality-rich interactive characters. In *Proceedings of the 2004 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2004.

[20] Magerko, B., Laird, J., Assanie, M., Kerfoot, A., and Stokes, D. AI characters and directors for interactive computer games. In *Proceedings of the 2004 Innovative Applications of Artificial Intelligence Conference*, San Jose, CA, 2004.

[21] Marsella, S., Johnson, L., and LaBore, C. Interactive pedagogical drama for health interventions. In *Proceeding of the 11th International Conference on Artificial Intelligence in Education*, Sydney, Australia, 2003.

[22] Mateas, M. and Stern, A. Structuring content in the façade interactive drama architecture. In *Proceedings of Artificial Intelligence and Interactive Digital Entertainment*, Marina del Rey, 2005.

[23] Meehan, J. Tale-spin, an interactive program that writes stories. In *Proceedings of the 5th International Joint Conferences on Artificial Intelligence*, 91-98, 1977.

[24] Murray, J. *Hamlet on the Holodeck*. Cambridge, MA: MIT Press, 1998.

[25] Murray, R. and VanLehn, K. DT Tutor: a dynamic, decision-theoretic approach for optimal selection of tutorial actions. In *Proceedings of the 5th International Conference on Intelligent Tutoring Systems*, Springer-Verlag, 153-162, 2000.

[26] Nelson, M. and Mateas, M. Search-based drama management in the interactive fiction anchorhead. In *Proceedings of Artificial Intelligence and Interactive Digital Entertainment,* Marina del Rey, 2005.

[27] Pearl J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kauffman, USA. 1988.

[28] Prada, R., Machado, I., and Paiva, A. TEATRIX: virtual environment for story creation. In *Proceedings of the 5th International Conference on Intelligent Tutoring Systems*, Springer-Verlag, 464-473, 2000.

[29] Riedl, M., Saretto, C., and Young, M. Managing interaction between users and agents in a multiagent storytelling environment. In *Proceedings of the Second International Conference on Autonomous Agents and Multi-Agent Systems*, June, 2003.

[30] Riedl, M. and Young, M. An intent-driven planner for multi-agent story generation. In *Proceedings of the Third International Joint Conferences on Autonomous Agents and Multi-Agent Systems*, New York, 2004.

[31] Shachter, R. and Peot, M. Simulation approaches to general probabilistic inference on belief networks. In *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, 221-234, 1990.

[32] Si, M., Marsella, S., and Pynadath, D. Thespian: using multi-agent fitting to craft interactive drama. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems*, 2005.

[33] Swartout, W., Hill, R., Gratch, J., Johnson, L., Kyriakakis, C., Labore, C., Lindheim, R., Marsella, S., Miraglia, D., Moore, B., Morie, J., Rickel, J., Thiebaux, M., Tuch, L., Whitney, R., and Douglas, J. Towards the holodeck: integrating graphics, sound, character and story. In *Proceedings of the Fifth International Conference on Autonomous Agents*, 2001.

[34] Turner, S. *MINSTREL: A computer model of creativity and storytelling*. Ph.D. Thesis, University of California, Los Angeles, CA, 1992.

[35] Weyhrauch, P. *Guiding interactive ficiton*. Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, PA, 1997.

[36] Yuan, C. and Druzdzel, M. An importance sampling algorithm based on evidence pre-propagation. In *Proceedings of the Nineteenth Annual Conference on Uncertainty in Artificial Intelligence*, 624-631, 2003.

Evaluating a Computational Model of Social Causality and Responsibility

Wenji Mao
University of Southern California
Institute for Creative Technologies
13274 Fiji Way, Marina del Rey, CA 90292
mao@ict.usc.edu

Jonathan Gratch
University of Southern California
Institute for Creative Technologies
13274 Fiji Way, Marina del Rey, CA 90292
gratch@ict.usc.edu

ABSTRACT
Intelligent agents are typically situated in a social environment and must reason about social cause and effect. Such reasoning is qualitatively different from physical causal reasoning that underlies most intelligent systems. Modeling social causal reasoning can enrich the capabilities of multi-agent systems and intelligent user interfaces. In this paper, we empirically evaluate a computational model of social causality and responsibility against human social judgments. Results from our experimental studies show that in general, the model's predictions of internal variables and inference process are consistent with human responses, though they also suggest some possible refinement to the computational model.

Categories and Subject Descriptors
I.2 [**Computing Methodologies**]: Artificial Intelligence; J.4 [**Computer Applications**]: Social and Behavioral Sciences

General Terms
Algorithms, Experimentation

Keywords
Intelligent Agents, Cognitive Modeling, Causality, Commonsense Reasoning

1. INTRODUCTION
A growing number of applications seek to incorporate automatic reasoning techniques into intelligent agents. Many intelligent systems incorporate planning and reasoning techniques designed to reason about *physical* causality. Since intelligent agents are typically situated in a multiagent environment and multiagent interactions are inherently *social*, physical causes and effects are simply inadequate for explaining social phenomena. In contrast, social causality, both in theory and as practiced in everyday folk judgments, emphasizes multiple causal dimensions, involves epistemic variables, and distinguishes between physical cause, responsibility and blame.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005…$5.00.

The multiagent research community has considered many aspects of social reasoning (e.g., [1, 20]), but has largely ignored this crucial distinction between physical and social causal reasoning. A model of the process and inferences underlying social causality can enrich the cognitive and social functionality of intelligent agents. Such a model can help an agent to interpret the observed social behavior of others and impact the way an agent acts on the world, which is crucial for successful interactions among intelligent entities. With the advance of multi-agent systems and systems that socially interact with people, it is increasingly important to model this central form of human social inference. Social causal reasoning can inform the design of human-like agents, guide conversation strategies and help modeling and understanding social emotions [6].

We have developed a general computational model of social causality and responsibility [10, 11] that formalizes the factors people use in reasoning about social events. Psychological and philosophical theories identify intermediate constructs that determine social causality. In these theories, social causality involves not only physical causality, but also epistemic variables such as freedom of choice, intention and foreknowledge [19, 22, 24]. As a result, an actor may physically cause an event, but be absolved of responsibility and blame. Conversely, a person may be held responsible and blameworthy for what she did not physically cause. Our model serves as a bridge between such theoretical distinctions and their computational realization, by inferring these additional factors from a representation of the physical and social context.

To evaluate this computational model, we need to assess the model's consistency with human judgments of social cause and responsibility. This is challenging given that people's judgment results often vary, and sometimes, there might be even no consensus among people. But further, we are interested in the more challenging task of testing whether the inferential mechanism of the model is consistent with the intuition of human inference, that is, does our model infer judgment results *in the way* people actually do?

In the rest of the paper, we first review the psychological theory for social causality and responsibility, and the computational model we develop. We then discuss the details of the experimental evaluation of the model, including the methodology, results and some empirical findings.

2. ATTRIBUTION THEORY

Our work is based on the influential attribution theories of Shaver [19] and Weiner [22] of social causality and responsibility (we adopt the terminology of Shaver in this paper). Their theories argue that physical causality and coercion identify *who* is responsible for some outcome under judgment, whereas epistemic factors, intention and foreseeability, determine *how much* responsibility and blame/credit are assigned. Below we summarize their theories.

The assessments of physical causality and coercion identify the responsible party. *Physical causality* (including personal causality and environmental causality) refers to the connection between events and the outcomes they produce. Only when human agency is involved, does an event become relevant to the investigation of responsibility and blame/credit. In the *absence of coercion*, the actor whose action directly produces the outcome is regarded as responsible. However, in the *presence of coercion* (as when some external force, such as a more powerful individual or a socially sanctioned authority, limits an agent's freedom of choice) some or all of the responsibility may be deflected to the coercive force.

Intention and forseeability determine the degree of responsibility. *Intention* is generally conceived as the commitment to work towards a certain act or outcome. Most theories view intention as the major determinant of the degree of responsibility. If an agent intends an action to achieve an outcome, then the agent must have the foreknowledge that the action brings about the outcome. The higher the degree of intention, the greater the responsibility assigned. *Foreseeability* refers to an agent's foreknowledge about actions and their consequences. The lower the degree of foreseeability, the less the responsibility assigned.

An agent may intentionally perform an action, but may not intend all the action effects. It is *outcome intention* (i.e., intended action effect), rather than *act intention* (i.e., intended action) that are key in responsibility judgment [23]. Similar difference exists in *outcome coercion* (i.e., coerced action effect) and *act coercion* (i.e., coerced action). The result of the judgment process is the assignment of certain blame or credit to the responsible agent(s). The intensity of blame or credit is determined by the severity or positivity of the outcome as well as the degree of responsibility. The latter is based on the assessed values of attribution variables.

3. THE COMPUTATIONAL MODEL (A REVIEW)

To model the process of social causality and responsibility attribution, we have constructed a computational model that can automatically derive the judgments that underlying attributions of responsibility and blame from observations and knowledge about social acts. Two important sources of information facilitate this inference process. One source is the actions performed by the observed agents (including physical acts and speech acts). The other is the causal knowledge about actions and their effects. To represent causal knowledge, we have adopted a hierarchical plan representation used by many intelligent systems. This representation provides a concise description of the causal relationship between events and states. It also provides a clear structure for exploring alternative courses of actions, and plan interactions. The computational model is described in detail elsewhere [10, 11, 6]. Here we briefly review the inference techniques.

Figure 1 illustrates an overview of the computational model. The inference process infers beliefs from dialogue evidence and causal evidence. Both dialogue inference and causal inference make use of commonsense heuristics, and derive beliefs about the attribution values.

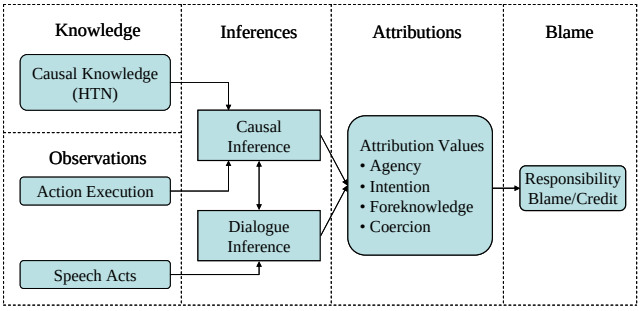


Figure 1. Computational model of responsibility attribution.

Dialogue inference reasons about beliefs of participating agents' desires, intentions, foreknowledge and choices from dialogue communication, represented as a sequence of speech acts. For example, *inform* gives evidence that both the speaker and the hearer know the content of the act, given the conversation between agents is *grounded* [21]. A *request* shows the speaker's desire (or want). An *order* shows the speaker's intent, and if it is successfully issued, it creates an obligation for the hearer to perform the content of the act. The hearer may *accept*, *reject* or *counter-propose* the request or order. Various inferences can be made depending on current beliefs and the response of the hearer [10].

Causal inference adopts a plan-based approach to reason about agency, coercion and intentions. To infer *agency*, the approach first identifies the performing agent whose action directly causes the outcome. Other agents who assist the performer by enabling action preconditions are viewed as indirect agency.

Determinations of *coercion* are informed by dialogue and causal inference. Two concepts are important in assessing coercion. One is social *obligation*, created by utterance, role assignment, etc. The other is *unwillingness*. For example, if an authorizing agent commands another agent to perform a certain action, then the latter agent has an obligation to do so. If there is no clear evidence that an agent intends beforehand, and the agent accepts the obligation, there is evidence of *weak* coercion. Note, intend(x, p, $t1$) denotes that agent x intends that proposition p at time $t1$, obligation(x, p, y, $t2$) represents that x has an obligation p by agent y at time $t2$, accept(x, p, $t3$) represents that x accepts that p at time $t3$, and coerce(y, x, p, $t4$) represents that y coerces x that p at time $t4$.

$$\neg(\exists t1)(t1<t3 \wedge \text{intend}(x, p, t1)) \wedge \text{obligation}(x, p, y, t2) \wedge \text{accept}(x, p, t3) \wedge t2<t3<t4 \Rightarrow \text{coerce}(y, x, p, t4)$$

If there is clear evidence of the unwillingness (i.e., intend(x, p, $t1$) is false), there is evidence of *strong* coercion.

$$\neg\text{intend}(x, p, t1) \wedge \text{obligation}(x, p, y, t2) \wedge \text{accept}(x, p, t3) \wedge t1<t3 \wedge t2<t3<t4 \Rightarrow \text{coerce}(y, x, p, t4)$$

Causal inference helps infer outcome coercion from evidence of act coercion, by examining the plan structure and action alternatives. For example, assume an agent is coerced to perform an ab-

stract action. If there is only one way to realize (i.e., decompose) this act, the model concludes that the agent must perform all the subsequent actions within this realization and the occurrences of all the action effects are unaviodable. If, instead, the coerced action has multiple realizations (i.e., decompositions), only the effects shared across all of these alternatives are necessarily coerced. Non-shared effects only occur if the coerced agent chooses that particular realization of the coerced act. Outcomes can also be coerced indirectly, for example, some agents can block other alternatives by disabling action preconditions; or they can enable a conditional effect. Similarly, indirect coercion can be inferred from plan knowledge and action alternatives [13].

Intentions (i.e., act intention and outcome intention) can be inferred from dialogue evidence. For example, rejecting what the speaker wants or intends shows no intention. Besides, causal inference helps partially infer outcome intention from act intention. For example, if an agent intends an action voluntarily, the agent must intend at least one action effect. If there is only one action effect (significant to the agent), we can exactly infer which effect the agent intends. As plans provide context in evaluating intention, with association to the goals and reasons of an agent's behavior, in the absence of clear evidence from dialogue inference, we employ a general plan-based algorithm to recognize intentions [12]. If a plan is intended by agents, then the actions and effects that are *relevant* to goal achievement (i.e., in the path from initial states to goal states of the plan) are intended. Other action effects are viewed as *side effects* in goal attainment and thus are not intended by the agents.

Foreknowledge refers to an agent's epistemic state before action execution. It is inferred from dialogue evidence (e.g., speech act inform, tell or assert). Intention recognition also helps infer an agent's foreknowledge, as intentions entail foreknowledge (*Axiom 4* in [11]).

The attribution values derived from causal inference and dialogue inference serve as inputs of the attribution process. The algorithm is detailed in [11]. As the last step, this algorithm determines responsibility and assigns proper blame or credit to the responsible agent(s) based on the implications of attribution theory.

4. RELATED WORK

Recent computational approaches have addressed social causality and responsibility judgment by extending causal models [7, 3]. Halpern and Pearl [7] proposed a definition of *actual cause* within the framework of structural causal models. As their approach can extract more complex causal relationships from simple ones, their model is capable of inferring indirect causal factors including social causes. Take the "firing squad" example [17]: there is a two-man firing squad; on their captain's order, both riflemen shoot simultaneously and accurately, and the prisoner dies. Besides the two riflemen who physically cause the death, Halpern & Pearl's model can find the captain as one actual cause for the death. Chockler and Halpern [3] extended this notion of causality, to account for degree of *responsibility*. They gave a definition of responsibility. For example, if a person wins an election 11-0, then each voter who votes for her is a cause for the victory, but each voter is less responsible for the victory than each of the voters in a 6-5 victory. Based on this definition of responsibility, they then defined the *degree of blame*, using the expected degree of responsibility weighed by the epistemic state of an agent.

Chockler & Halpern's extended definition of responsibility accounts better for multiple causes and the extent to which each cause contributes to the occurrence of a specific outcome. Another advantage of their model is that their definition of degree of blame takes an agent's epistemic state into consideration. However, they only consider one epistemic variable, that is, an agent's knowledge prior to action performance (corresponding to foreseeability in Shaver's term). Important concepts in moral responsibility, such as intention and freedom of choice are excluded in their definition. As a result, their model uses foreknowledge as the only determinant for blame assignment, which is inconsistent with psychological theories. As their model is the extension of counterfactual reasoning within the structural-model framework, and structural-model approach represents all the events as random variables and causal information as equations over the random variables, this brings about other limitations in their model. For instance, causal equations do not have direct correspondence in computational systems, so it is hard to obtain them for practical applications. As communicative events are also represented as random variables in their model (which is propositional), it is difficult to construct equations for communicative acts and infer intermediate beliefs (e.g., beliefs about desires, intentions, etc) that are important for social causal reasoning.

In contrast, our approach is built on general plan representation commonly used in many intelligent systems. Causal inference is a plan-based evaluation over this representation. Our model takes different forms of interactions into account, and makes use of commonsense reasoning to infer beliefs from dialogue communication. In Mao & Gratch [13], we use four variants of the original firing squad scenario in the related work [3], to empirically compare our model with Chockler & Halpern's model and two other models (i.e., simple cause model and simple authority model). The results show that for responsibility and blame assignments, our model better approximates human judgments than these alternative models [13].

Social psychological studies show that people consider intentions, coercion and foreknowledge in their judgments. In our work, we have come up with computational account of all these variables. We would like to directly assess our model's ability in predicting the internal variables with respect to human results, and the veracity of the inference process that leads to the corresponding results.

5. EXPERIMENTAL EVALUATION

To evaluate the model, first we need to assess its consistency with human responses (i.e., given the same inputs, do people and the model produce the same outputs). Rather than simply viewing the model as a black box, however, we are also interested in assessing the consistency of the model's internal structure and processes underlying human attributions of responsibility and blame (i.e., do people use the same sources of evidence and generate the same intermediate conclusions). The results for the first task were already demonstrated in [13], though here we seek to extend these finding to additional scenarios. The second task is the focus of this paper.

5.1 Method

Our model embodies the theoretical view that people will judge social cause and responsibility differently based on their perception of the key variables such as intentions and coercion. Thus, a good test is to see how the model performs when the evidence for

such judgments is systematically varied. In this study, we take a description of a single social situation and systematically generate several variants, using the inference rules of our model as a guide. For example, if our model suggests that a particular line of evidence is necessary to infer coercion, than an obvious variation would be to eliminate that line.

As a starting point, we adopt the well-known "company program scenario" (*Figure 3*: *Scenario 2*) in experimental philosophy research [5, 8]. In our study, descriptions of each scenario are organized into separate labeled statements of evidence (e.g., *E1-E6*). In *Scenario 1* (*Figure 2*), we manipulate evidence related to foreknowledge of the outcome:

Scenario 1:

E1 The vice president of Beta Corporation goes to the chairman of the board and requests, "Can we start a new program?"

E2 The vice president continues, "The new program will help us increase profits,

E3 and according to our investigation report, it has no harm to the environment."

E4 The chairman answers, "Very well."

E5 The vice president executes the new program.

E6 However, the environment is harmed by the new program.

Questions:

1. Does the vice president want to start the new program?
   Your answer: Yes No
   Based on which information (circle all that apply)?
   E1 E2 E3 E4 E5 E6
2. Does the chairman intend to start the new program?
   Your answer: Yes No
   Based on which information (circle all that apply)?
   E1 E2 E3 E4 E5 E6
3. Is it the chairman's intention to increase profits?
   Your answer: Yes No
   Based on which information (circle all that apply)?
   E1 E2 E3 E4 E5 E6
4. Does the vice president know that the new program will harm the environment?
   Your answer: Yes No
   Based on which information (circle all that apply)?
   E1 E2 E3 E4 E5 E6
5. Is it the vice president's intention to harm the environment by starting the new program?
   Your answer: Yes No
   Based on which information (circle all that apply)?
   E1 E2 E3 E4 E5 E6
6. How much would you blame the individuals for harming the environment?
   Blame the chairman: 1 2 3 4 5 6
   Blame the vice president: 1 2 3 4 5 6
   Little Lots

Figure 2. Scenario 1 and the questionnaire

Each scenario is followed by a questionnaire. The questions in the questionnaires are designed to test the beliefs about different variables. *Figure 2* shows the wording of the questions after *Scenario 1*.

Scenario 2:

E1 The chairman of Beta Corporation is discussing a new program with the vice president of the corporation.

E2 The vice president says, "The new program will help us increase profits,

E3 but according to our investigation report, it will also harm the environment."

E4 The chairman answers, "I only want to make as much profit as I can. Start the new program!"

E5 The vice president says, "Ok," and executes the new program.

E6 The environment is harmed by the new program.

Figure 3. Scenario 2

In *Scenario 3*, we manipulate the degree of perceived coercion and willingness of the coerced agent by introducing an alternative course of action that will not harm the environment and which the vice president prefers. Specifically, we add one line to Scenario 2 between *E3* and *E4*:

Scenario 3:

E1 The chairman of Beta Corporation is discussing a new program with the vice president of the corporation.

E2 The vice president says, "The new program will help us increase profits,

E3 but according to our investigation report, it will also harm the environment.

E4 Instead, we should run an alternative program, that will gain us fewer profits than this new program, but it has no harm to the environment."

E5 The chairman answers, "I only want to make as much profit as I can. Start the new program!"

E6 The vice president says, "Ok," and executes the new program.

E7 The environment is harmed by the new program.

Figure 4. Scenario 3

In *Scenario 4*, we manipulate freedom of choice. We introduce an alternative, but the preference of the vice president is based on a feature unrelated to the environment and the vice president is allowed to choose the option:

Scenario 4:

E1 The chairman of Beta Corporation is discussing a new program with the vice president of the corporation.

E2 The vice president says, "There are two ways to run this new program, a simple way and a complex way.

E3 Both will equally help us increase profits, but according to our investigation report, the simple way will also harm the environment."

E4 The chairman answers, "I only want to make as much profit as I can. Start the new program either way!"

E5 The vice president says, "Ok," and chooses the simple way to execute the new program.

E6 The environment is harmed.

Figure 5. Scenario 4

Table 1. Model predictions and people responses for sample scenarios

		Question 1		Question 2		Question 3		Question 4		Question 5		Question 6	
		Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Chair	VP
S1	Model	√		√		√			√		√		√
	People	30	0	27	3	29	1	2	28	0	30	3.00	3.73
S2	Model	√		√		√			√	√		√	
	People	30	0	30	0	30	0	10	20	22	8	5.63	3.77
S3	Model	√			√	√		√		N/A		√	
	People	21	9	2	28	29	1	21	9			5.63	3.23
S4	Model	√			√		√	N/A		N/A			√
	People	21	9	5	25	5	25					4.13	5.20

5.2 Results

At the level of assessing internal variables, we have no computational alternative to compare with (as the related work only infers part of the variables). In our experiment, we directly compare the predictions of the model with human data.

5.2.1 Assessing Inferred Beliefs

Table 1 shows the number of responses to each question in the sample scenarios. The values for the last questions are the averages of people's answers (on a 6-point scale). The model's predictions are checked with '√'. The data show that for most questions, people agree with each other quite well. But certain disagreement exists on some of the questions.

Though people sometimes may disagree with each other on specific questions, our purpose is to assess the model's general agreement with people. We measure the agreement of the model and each subject using *Kappa statistic*. *Kappa* coefficient is the de facto standard to evaluate inter-rater agreement for classification tasks [1]. It corrects the raters' proportional agreement $P(A)$ due to chance agreement $P(E)$:

$$K = \frac{P(A) - P(E)}{1 - P(E)}$$

Di Eugenio & Glass [4] argue that the computation of K coefficient is sensitive to the skewed distribution of categories (i.e., prevalence). In our treatment, we account for prevalence and construct contingency tables for the calculation. We average the results of *Kappa* agreement of the model's predictions with each subject's answers. *Table 2* gives the results of *Kappa* agreement of the model's predictions with each subject's answers.

Table 2. The *Kappa* agreement of the model and subjects

Subjects	*P*(*A*)	*P*(*E*)	*K*
1	.824	.526	.628
2	.882	.543	.742
3	.706	.491	.422
4	.882	.509	.761
5	.941	.526	.876
6	.882	.543	.742
7	.941	.561	.866
8	.941	.561	.866
9	.882	.543	.742
10	.941	.526	.876
11	1	.543	1
12	.882	.543	.742
13	1	.543	1
14	.941	.526	.876
15	.765	.543	.485
16	.824	.491	.667
17	.824	.561	.598
18	.765	.543	.485
19	.882	.509	.761
20	.941	.526	.876
21	.824	.561	.598
22	.882	.543	.742
23	.765	.543	.485
24	.941	.526	.876
25	.941	.561	.866
26	.882	.509	.761
27	.765	.578	.443
28	.882	.543	.742
29	.824	.491	.667
30	.882	.509	.761
Average			.732

The average *Kappa* agreement between the model and subjects is *0.732*. According to [18], $0.61<K<0.80$ indicates substantial agreement. The empirical results show good consistency between the model's generation of intermediate beliefs and human data.

5.2.2 Assessing Inference Rules

In our model, each belief is derived by a specific inference rule, so each question in the questionnaire corresponds to the firing of one rule (with the exception of three questions in the questionnaire designed to test two rules each). Currently, we have 37 dialogue and causal inference rules in the model. This survey study covers 19 of them. To assess the inference rules, we compare the conditions of each rule with the evidence people use in forming each answer. For every subject and every question, we build a confusion matrix [9] to compute the numbers of true positive *TP* (i.e., evidence both the rule and the subject use), true negative *TN* (i.e., evidence both the rule and the subject ignore), false positive (i.e., evidence the rule incorrectly uses) and false negative (i.e., evidence the rule incorrectly ignores).

For each question Q_i, the correct prediction of the corresponding rule with respect to people's evidence choice is measured by accuracy (*AC*), where N_s is the total number of subjects and N_e is the total number of evidence for Q_i.

$$AC(Q_i) = \frac{\sum_{j \in Subjects} AC(j, Q_i)}{N_S} = \frac{\sum_{j \in Subjects} (TP(j, Q_i) + TN(j, Q_i))}{N_S * N_e}$$

Table 3 gives the accuracy of each tested rules (*R**). Given that each question contains 6 or 7 lines of evidence and people choose multiple lines in most cases, the accuracy results are fairly good (well above those generated by a random model). The empirical results show that the sources of evidence the model uses for inference are consistent with human data.

Table 3. Accuracies of inference rules

	Question/Rule	Average Accuracy
Scenario 1	Q1/R7	0.76
	Q2/R13	0.96
	Q3/R2	0.85
	Q4/R5	0.94
	Q5/R1	0.91
Scenario 2	Q1/R6	0.92
	Q2/R10	0.96
	Q3/R24	0.86
	Q4/R25	0.70
	Q5/R11&R14	0.84
Scenario 3	Q1/R17	0.94
	Q2/R18&R19	0.88
	Q3/R11&R16	0.80
	Q4/R27	0.74
Scenario 4	Q1/R29	0.71
	Q2/R28	0.84
	Q3/R30	0.75

In the next section, we discuss how our model appraises each scenario and some experimental findings.

5.3 Discussions

5.3.1 Scenario 1

In *Scenario 1*, we manipulate the variable foreseeability to be false (from evidence *E3*). Evidence is encoded into the model as follows (*VP* and *CH* refer to the vice president and the chairman, respectively):

(E1) request(*VP*, *CH*, do(*VP*, new-program))
(E2) inform(*VP*, *CH*, profit-increase∈effect(new-program))
(E3) inform(*VP*, *CH*, env-harm∉effect(new-program))
(E4) accept(*CH*, do(*VP*, new-program))
(E5) do(*VP*, new-program)
(E6) occur(environmental-harm)

The evaluation shows that the model and people draw similar intermediate conclusions from this evidence, suggesting both that our representation of the evidence and the underlying inference rules are largely correct. For example, the questionnaire specifically queries the perceived desire, foreknowledge and intentions of the characters (see *Figure 2*). The belief that the vice president desires the new program can be inferred from speech act *request* in *E1*. The chairman's intention to start the new program can be inferred from speech act *accept* in *E4*. As starting *new program* has only one action effect (*E2*), we can infer outcome intention from act intention. The chairman must intend the only effect (i.e., *increase profits*). The vice president has no foreknowledge of the environmental harm can be inferred from the content of *inform* in *E3* (note that our approach assumes Grice's maxims of *Quality* and *Relevance*). According to a causal inference rule, no foreknowledge entails no intention.

As the goal of agent(s) was not explicitly expressed in this scenario, intention recognition method is not involved. However, from dialogue evidence and from causal connection of foreknowledge, act intention and outcome intention, the beliefs about intentions are properly inferred.

Subjects gave quite consistent answers to the questions in *Scenario 1*. Their answers to the last question show that blameworthiness is mitigated by no foreknowledge. This result is consistent with psychological findings. Though people assigned relatively more blame to the vice president, the data also suggest that the chairman should share blame with the vice president.

The accuracies of inference rules are also good in general. The accuracy of the rule tested in *Question 1* is lower than the others because besides evidence *E1*, many people chose *E2* as well. Post-experiment interviews with the subjects uncovered that many subjects had assumed that making profits should be desirable to the vice president (because of his role), and therefore, he should desire to start the new program to increase profits (which is supported by *E2*).

5.3.2 Scenarios 2&3

Scenarios 2 & 3 manipulate the degree of perceived coercion and willingness of the coerced agent. The agents have clear foreknowledge about the harm (*E3*). The chairman's goal of making *more profits* is also clearly informed (*E4* in *Scenario 2*, *E5* in *Scenario 3*).

The following beliefs can be inferred from *Scenario 2*:

(B1) know(*CH*, profit-increase∈effect(new-program))
(derived from *E2*: inform)
(B2) know(*CH*, env-harm∈effect(new-program))
(derived from *E3*: inform)
(B3) intend(*CH*, do(*VP*, new-program))
(derived from *E4*: order)
(B4) obligation(*VP*, do(*VP*, new-program), *CH*)
(derived from *E4*: order)
(B5) coerce(*CH*, *VP*, do(*VP*, new-program))
(derived from *B4* & *E5*: accept)
… …

As there is no evidence of unwillingness, perceived coercion (*B5*) is in a weak sense. The inferred beliefs *B2*, *B3*, *B4* and *B5* give predictions to *Questions 1, 2 & 5* in *Scenario 2*. As there is only one plan in this scenario and the chairman intends the action (i.e., starting *new program*) in the plan (*B3*), intention recognition is trivial[1]. Making *more profits* is the goal of the plan, so it is intended by the

[1] Note that our intention recognition method is generally applied to a plan library with multiple plans and sequences of actions. In this oversimplified example, intention recognition becomes trivial.

chairman (*Question 3*). *Environmental harm* is a side effect of goal attainment, so it is not intended by the chairman (*Question 4*).

In *Scenario 3*, the vice president's counter-proposal provides additional information (*E4*). More beliefs can be derived:

(B3) know(*CH*, alternative(new-program, alternative-program))
(from *E4*: counter-propose)
(B4) want(*VP*, do(*VP*, alternative-program))
(from *E4*: counter-propose)
(B5) ¬intend(*VP*, do(*VP*, new-program))
(from *E4*: counter-propose)
(B6) intend(*CH*, do(*VP*, new-program))
(from *E5*: order)
(B7) obligation(*VP*, do(*VP*, new-program), *CH*)
(from *E5*: order)
(B8) coerce(*CH*, *VP*, do(*VP*, new-program))
(from *B5*, *B7* and *E6*: accept)
(B9) coerce(*CH*, *VP*, achieve(environmental-harm))
(from *B2* and *B8*)
… …

Beliefs *B3*, *B4*, *B8* and *B9* give predictions to *Questions 1, 2, 3 & 4* in *Scenario 3*, respectively. Belief *B5* gives strong evidence of coercion.

There are several disagreements among the subjects in *Scenarios 2 & 3*. In *Question 4* of *Scenario 2*, one-third of the subjects think it the chairman's intention to harm the environment. Whether side-effects are intended is controversial in philosophy, and other empirical studies show similar results [16]. Also in *Question 5* of *Scenario 2*, some subjects think the vice president is not coerced to start the new program by the chairman, as evidence is weaker than in *Scenario 3*. Half of them referred to evidence *E5*, indicating that they expect the vice president to negotiate with the chairman rather than directly accept the order. This suggests a limitation in our current model. In contrast, when asking the same question in *Scenario 3* (*Question 3*), almost all the subjects agreed that the vice president is coerced to start the new program.

In the first question of *Scenario 3*, some subjects think the chairman does not know the alternative program, though the vice president clearly informs this in the scenario. Most of these subjects (80%) referred to evidence *E5*, showing that they looked for grounding information. As our model infers grounded information from conversation, it is our mistake to omit this information in scenario design. Last, in *Question 4* of *Scenario 3*, some subjects seemed reluctant to infer outcome coercion from evidence of act coercion. Nonetheless, they still assigned high degree of blame to the chairman. Comparing the blame assignments in *Scenarios 2 & 3*, it shows that on one hand, the higher the degree of coercion, the less blame is assigned to the actor. This is consistent with psychological findings. On the other hand, even when perceived coercion is not strong, people still assign high degree of blame to the coercer, as in *Scenario 2*.

The accuracies of two tested rules are relatively lower than the others. One is the rule used in *Question 4* of *Scenario 2*. In our model, there are three evidence needed for the inference, *E2*, *E3* and *E4*. Almost all subjects chose evidence *E4*, but most ignored *E2* (except two subjects). One reason is that *E2* as knowledge (i.e., the new program helps increase profits), seems implied in *E4* (otherwise people will have difficulty understanding *E4*). Similarly, for the rule used in *Question 4* of *Scenario 3*, most subjects did not choose knowledge *E3*. However, we think this knowledge is necessary for the inference.

5.3.3 Scenario 4

In *Scenarios 2 & 3*, action and alternative are coerced by authority, whereas in *Scenario 4*, the vice president has some freedom of choice. While the high-level plan (i.e., starting the new program) is still coerced (*E4*), the agent can choose to execute either alternative (simple way or complex way in *E2*). As both ways will increase profits (*E3*), increasing profit is unavoidable in either way: the vice president is coerced to achieve this effect (*Question 1*). However, he is neither coerced to choose the simple way (*Question 2*), nor is he coerced to achieve the specific effect *environmental harm* that only occurs in the simple way (*Question 3*).

In *Question 1*, some subjects think that the vice president is not coerced to increase profits, for the same reason mentioned earlier. They think it the vice president's job to increase profits, so he must be willing to do so. People assigned more blame to the vice president, as he could have done otherwise. This result is consistent with psychological findings. However, people still assigned considerable blame to the chairman, though it was the vice president's choice to harm the environment.

The inference rules in *Question 1* and *Question 3* are based on evidence *E3*, *E4* and *E5*. Many subjects ignore knowledge *E3*. This lowers the accuracies of the two rules.

5.3.4 Discussion on limitations

Although results show general support for the model, they reveal some limitations of the approach. Subjects tended to assign shared blame to the individuals involved. Though our model is able to handle joint activity and multiagent plan, one limitation of the model is that it always blames one most deserving agent (or group of agents).

It is clear that people made assumptions about the scenarios that were not explicitly represented in the model. For example, people assumed the vice president had the goal of increasing profits even though this was not explicitly stated. This relates to the more general issue of ensuring the correspondence between the model's encoding of the scenario and the subjects' interpretation of the scenario. Currently, we construct this mapping by hand. This has the disadvantage that, as designers of the scenarios, we may unintentionally introduce discrepancies. Alternatives would be to explore ways to automatically generate descriptions from their representation in the model, or at least to use an independent set of coders to characterize the textual encoding.

It is well known that responsibility judgment is influenced by the observer's emotional state, interpersonal goals such as impression management [15], and dispositional differences such as personality. Further, our model of dialogue inference assumes that parties faithfully articulate their actions and beliefs, whereas people are notoriously biased when describing their involvement in creditworthy/blameworthy events. Although we have not accounted for these biases, our current model provides a framework for both generating and recognizing such influences [14].

6. CONCLUSION

A growing number of applications seek to incorporate automatic reasoning techniques into intelligent agents. In contrast to physical causal reasoning that underlies most intelligent systems, social causal reasoning emphasizes multiple causal dimensions, involves

epistemic variables, and distinguishes between physical cause, responsibility and blame. In this paper, we empirically evaluate a computational model of social causality and responsibility using human data. Results from our experimental studies are encouraging in general, though they also indicate some limitations and possible refinement to the computational model.

Our future work needs to better model shared responsibility and blame, and consider the influences of individual difference and the perceiver's subjective bias on the judgment process. We also need to improve experimental design, using objective device to minimize the discrepancies of information encoding.

7. ACKNOWLEDGEMENTS

This work was sponsored by the U. S. Army Research, Development, and Engineering Command (RDECOM). The first author benefits from the USC Knowledge Representation Group seminars led by Jerry Hobbs and Andrew Gordon. Thanks to Anya Okhmatovskaia for the helpful discussions. The content does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

8. REFERENCES

[1] Carletta, J. Assessing Agreement on Classification Tasks: the Kappa Statistic. *Computational Intelligence*, 22(2):249-254, 1996.

[2] Castelfranchi, C. Social Power. *Proceedings of the First European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, 1990.

[3] Chockler, H., and Halpern, J. Y. Responsibility and Blame: A Structural-Model Approach. *Journal of Artificial Intelligence Research*, 22:93-115, 2004.

[4] Di Eugenio, B., and Glass, M. The Kappa Statistic: A second Look. *Computational Linguistics*, 30(1):95-101, 2004.

[5] Experimental Philosophy Website (http://experimentalphilosophy.typepad.com/).

[6] Gratch, J., Mao, W., and Marsella, S. Modeling Social Emotions and Social Attributions. In: R. Sun (Ed.), *Cognition and Multi-Agent Interaction*. Cambridge University Press, 2006.

[7] Halpern, J. Y., and Pearl, J. Causes and Explanations: A Structural-Model Approach – Part I: Causes. *Proceedings of the Seventeenth Conference in Uncertainty in Artificial Intelligence*, 2001.

[8] Knobe, J. Intentional Action and Side-Effects in Ordinary Language. *Analysis*, 63:190-193, 2003.

[9] Kohavi, R., and Provost, F. Glossary of Terms. *Machine Learning*, 30(2/3):271-274, 1998.

[10] Mao, W., and Gratch, J. The Social Credit Assignment Problem (Extended Version). *USC/ICT Technical Report* (http://www.ict.usc.edu/publications/ICT-TR-02-2003.pdf), 2003.

[11] Mao, W., and Gratch, J. Social Judgment in Multiagent Interactions. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004a.

[12] Mao, W., and Gratch, J. A Utility-Based Approach to Intention Recognition. *AAMAS 2004 Workshop on Agent Tracking: Modeling Other Agents from Observations*, 2004b.

[13] Mao, W., and Gratch, J. Social Causality and Responsibility: Modeling and Evaluation. *Proceedings of the Fifth International Conference on Intelligent Virtual Agents*, 2005.

[14] Martinovski, B., Mao, W., Gratch, J., and Marsella, S. Mitigation Theory: An Integrated Approach. *Proceedings of the Twenty-Seventh Annual Conference of the Cognitive Science Society*, 2005.

[15] Mele, A. R. *Self-Deception Unmasked*. Princeton University Press, 2001.

[16] Nadelhoffer, T. On saving the Simple View. *Mind and Language*, forthcoming.

[17] Pearl, J. Reasoning with Cause and Effect. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

[18] Rietveld, T., and van Hout, R. *Statistical Techniques for the Study of Language and Language Behavior*. Mouton de Gruyter, 1993.

[19] Shaver, K. G. *The Attribution Theory of Blame: Causality, Responsibility and Blameworthiness*. Springer-Verlag, 1985.

[20] Sichman, J. S., Conte, R., Demazeau, Y., and Castelfranchi, C. A Social Reasoning Mechanism Based on Dependence Networks. *Proceedings of the Eleventh European Conference on AI*, 1994.

[21] Traum, D. *A Computational Theory of Grounding in Natural Language Conversation*. Ph.D. Thesis, University of Rochester, 1994.

[22] Weiner, B. *The Judgment of Responsibility: A Foundation for a Theory of Social Conduct*. The Guilford Press, 1995.

[23] Weiner, B. Responsibility for Social Transgressions: An Attributional Analysis. In: B. F. Malle, L. J. Moses and D. A. Baldwin (Eds.), *Intentions and Intentionality: Foundations of Social Cognition*, pp. 331-344. The MIT Press, 2001.

[24] Zimmerman, M. J. *An Essay on Moral Responsibility*. Rowman & Littlefield, 1988.

A Model of Emotions for Situated Agents

H. Van Dyke Parunak, Robert Bisson, Sven Brueckner, Robert Matthews, John Sauter
Altarum Institute
3520 Green Court, Suite 300
Ann Arbor, MI 48105-1579 USA
1-734-302-4600

{van.parunak ,robert.bisson, sven.brueckner, robert.matthews, john.sauter }@altarum.org

ABSTRACT

In stressful situations such as combat, emotion is at least as important as rational analysis in determining behavior. DETT (Disposition, Emotion, Trigger, Tendency) is a model of emotion for situated agents that captures the essential features of the OCC (Ortony, Clore, Collins) model in a computationally efficient framework. We motivate and describe this architecture, and report preliminary experiments in simulating combat scenarios.

Categories and Subject Descriptors

I.2.0 [**Artificial Intelligence**]: General– *cognitive simulation.*
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence– *multiagent systems.*
J.4 **[Social and Behavioral Sciences]**: Psychology

General Terms

Algorithms, Experimentation, Theory.

Keywords

Emotion modeling, pheromones, swarm intelligence, BDI.

1. INTRODUCTION

Most computational models of emotion [13] require extensive symbolic reasoning, and are not practical in highly populous real-time domains. DETT (Disposition, Emotion, Trigger, Tendency) captures the essential features of the OCC (Ortony, Clore, Collins) emotion model efficiently, using digital pheromones as the agents' main source of perceptions. It extends current emotional models by defining a Disposition parameter that distinguishes agents' differing susceptibility to various emotions. DETT is designed for situated agents. An agent's emotions are triggered by its percepts, not just by internal reasoning.

Section 2 reviews previous work in computational emotions, particularly in the context of combat modeling. Section 3 describes the DETT model. Section 4 concludes.

2. PREVIOUS WORK

Our work rests on previous research in emotional modeling, and two existing agent-based combat models that can be interpreted as supporting elements of our approach.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005…$5.00.

2.1 Emotional Modeling

Research in agent-based modeling and human interfaces (including gaming) has increased interest in computational models of emotion [13], each drawing on different segments of a long psychological tradition [3]. For our purposes, a cognitive perspective on emotions (rather than an expressive or physiological one) is most appropriate, in particular the OCC model [8], where emotions are "valenced reactions to events, agents or objects." That is, the strength of an emotion depends on stimuli in the agent's environment. Their presence is mapped to a "valence," a positive or negative score, by a process called "appraisal."

Emotion can focus attention, increase the prominence of memories, affect cognitive style and performance, and influence judgments [1]. In OCC, "behavior is a response to an emotional state in conjunction with a particular initiating event." We focus on the impact of emotion on an agent's analysis, the process by which it selects intentions from desires. The left side of Figure 1 summarizes the Belief-Desire-Intention [14] model. Beliefs (derived from the environment by Perception) and Desires (constant over the time horizon of our model) feed Analysis, which produces Intentions, which in turn drive actions that change the environment. The right side enhances this model with OCC emotion. Beliefs feed not only Analysis, but also Appraisal, which generates Emotions. These Emotions in turn influence Analysis and Perception. (We do not implement the latter link in our current system).

Gratch and Marsella [4] offer one of the more mature current computational models of agent emotion. Table 1 summarizes the correspondence between the two models.

2.2 Emotional Combat Models

Explicit emotional modeling is rare in combat models, but elements of two recent models anticipate DETT.

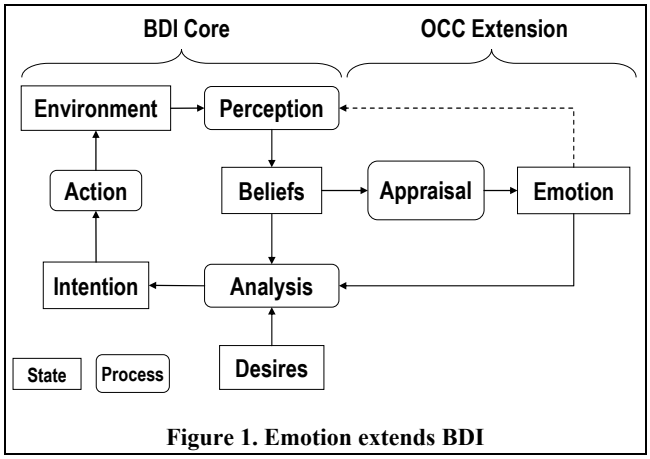


Figure 1. Emotion extends BDI

EINSTein [5] represents an agent's personality as six scalars in [-1, 1], describing the agent's response to six percepts: the number of alive friendly, alive enemy, injured friendly, and injured enemy troops within the agent's sensor rang, and the agent's distance from its own and its adversary's flag. (Each team seeks to capture the adversary's flag while protecting its own.) A positive weight indicates that the agent is attracted to the entity described by the weight, while a negative weight indicates that it is repelled.

Table 1. Comparison of Models

BDI + OCC	Gratch-Marsella
Environment	Environment
Perception	Causal Interpretation
Beliefs	Causal Interpretation
Appraisal	Appraisal
Emotion	Affective State
Analysis	Coping
Desires	???
Intention	Control Signals
Action	Action

MANA [7] extends EINSTein. Friendly and enemy flags are replaced by the waypoints being pursued by each side. MANA includes four additional components: low, medium, and high threat enemies. It also defines triggers (e.g., reaching a waypoint, being shot at, making contact with the enemy, being injured) that shift the agent from one personality vector to another. A default state defines the personality vector when no trigger state is active.

The notion of being attracted or repelled by friendly or adversarial forces in various states of health is an important component of what we informally think of as emotion (e.g., fear, compassion, aggression), and the use of the term "personality" in both EINSTein and MANA suggests that the system designers are thinking anthropomorphically, though they do not use "emotion" to describe the effect they are trying to achieve.

EINSTein and MANA implement subsets of Figure 1. The personality corresponds to the agent's Desires. EINSTein's personality vector guides the agent's decisions, but does not change in response to the agent's beliefs about the events, objects, or agents in its environment. Thus it is not a "valenced reaction." EINSTein does not capture OCC emotion. MANA's personality vectors depend on a trigger state, and implement a valenced reaction.

Both EINSTein and MANA do analysis by multiplying a vector of percepts by the personality, yielding a movement vector. In both models, the agent has perfect knowledge within a vision radius. These processes are much simpler than those in [4]. The differences reflect the systems' objectives. Gratch-Marsella supports a training environment with few agents. Regular human interaction slows the pace, permitting significant computation. EINSTein and MANA manipulate dozens or hundreds of agents in non-interactive simulations, and must minimize execution time.

3. The DETT Model

DETT supports the DARPA RAID program [6], whose objective is to anticipate enemy actions and deceptions to provide real-time support to a tactical commander. Our module [11] evolves agents against observations to learn their characteristics and determine which ones are most likely to reflect future behavior. These agents must execute faster than real time, so they cannot conduct complex symbolic reasoning, but use numerical computation via DETT. We also use the model in another project to automate the reactions of non-combatants with combatants. This capability requires us to recognize that non-combatants will have a range of personality types and to incorporate these differences in their behavior. Details are in the full paper.[1]

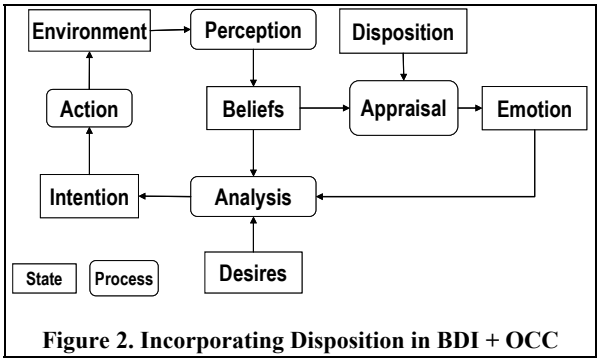


Figure 2. Incorporating Disposition in BDI + OCC

Both applications require an efficient way to include emotions in modeling combat. This reasoning takes place at two locations in Figure 1: Appraisal and Analysis.

3.1 Appraisal

MANA's triggered personality vectors are a useful model for appraisal, but have two limitations. First, MANA defines vectors and triggers for the squad, all of whose members share the same values. Individual combatants (e.g., new recruits vs. seasoned veterans) will differ widely in their susceptibility to different emotions. To use evolution to learn the characteristics of entities, we must parameterize this kind of difference. Second, MANA assumes that emotions arise and disappear immediately with the appropriate trigger. Empirically, emotions rise rapidly but not instantaneously, and persist after the trigger is removed.

To address the first concern we add a new component, Dispositions, to the model (Figure 2). Emotions and Dispositions correspond one to one. Like Desires, Dispositions are constant over each simulation. A Disposition modulates Appraisal to determine how much a Belief triggers the corresponding Emotion. The Emotion then modulates Analysis to impose a Tendency on the resulting Intention. The main elements of this model are thus the Disposition, Emotion, Trigger, and Tendency (the effect on intentions) (DETT). Table 2 gives two examples.

Agents sense one another through digital pheromones [2], labeled scalars that they deposit in the environment and that diffuse and evaporate over time. An agent believes what it senses in the form of pheromones. Our pheromone vocabulary for RAID includes flavors for live and dead friendly, hostile, and neutral individuals, weapons fire, key sites, and topographical features that provide cover from fire or that enhance mobility.

An agent's desires are values in [-1, +1]: ProtectRed (the adversary), ProtectBlue (friendly forces), ProtectGreen (civilians), ProtectKeySites, AvoidCombat, AvoidDetection, and Survive. Negative values reverse the sense: a negative ProtectRed is a desire to harm Red. Table 2 in [9] shows which pheromones attract or repel an agent with a given desire, and how that tendency translates into action. For example, an agent with a high positive ProtectRed will be attracted to RED-ALIVE, RED-CASUALTY, and MOBILITY pheromone, and will move at maximum speed.

Let $\boldsymbol{P}$ be the vector of pheromone strengths at an agent's location. The agent's Disposition is a matrix $\boldsymbol{D}$. $D[i,j] \in [0,1]$ is the relevance of the ith pheromone to the jth emotion. The jth emotion depends (nonlinearly) on the jth element of $\boldsymbol{P}^{\mathrm{T}}\boldsymbol{D}$.

A digital hormone [12], one per

[1] www.altarum.net/~vparunak/AAMAS06EmotionModel.pdf

agent per Emotion, models change in emotion with time. $\boldsymbol{P}^{\mathrm{T}}\boldsymbol{D}$ determines the current deposit to the hormone vector E, so the longer an agent is exposed to a trigger pheromone, the higher the level of the associated emotions. When the trigger is removed, its emotions decay exponentially. The higher the disposition, the more quickly the emotion grows in the presence of a trigger.

Table 2. Sample DETT Semantics

Disposition	Emotion	Trigger	Tendency
Cowardice	Fear	Armed enemy Incoming attack	Less attention to orders Tend to avoid threat
Irritability	Anger	Presence of enemy	More likely to fight Tend to move toward threat

3.2 Analysis

Analysis draws on the same pheromone vector $\boldsymbol{P}$ of beliefs as does Appraisal, and takes as input the current state of the emotion vector $\boldsymbol{E}$. In addition, it considers the values of the agent's vector of Desires or Wants $\boldsymbol{W}$. The desires we are modeling are Protect Red, Protect Blue, Protect Green, Protect Key Sites, Avoid Combat, Avoid Detection, and Survive. Each has a real value in the range [-1,1], where a negative value indicates that the agent wants the opposite state of affairs described by the desire. A movement matrix $\boldsymbol{M}$ indicates whether a given Desire tends to attract or repel the agent toward a given flavor of pheromone: $\boldsymbol{M}$[i,j] is 1 if desire j is attracted to pheromone i, -1 if it is repelled, and 0 if the pheromone is irrelevant to the desire.

In the absence of emotions, the agent's behavior is a nonlinear function of $\boldsymbol{P}^{\mathrm{T}}\boldsymbol{MW}$. Emotions modulate these behaviors. Elevated Anger increases movement likelihood, weapon firing likelihood, and tendency toward an exposed posture, while elevated Fear decreasse these likelihoods. Level of a particular emotion models the extent to which the emotion modulates the agent's behavior. Someone who experiences high fear, but continues to behave as if he were not afraid, would be modeled as having low fear. We are not trying to model emotion as experienced by an agent, only as perceived by its impact on the agent's behavior.

4. CONCLUSION

Emotion is critical in modeling agent behavior in stressful environments such as combat. The Gratch-Marsella model is a sophisticated implementation of current psychological theories of emotion, but is computationally too expensive for large populations of combatant agents. EINSTein and MANA are efficient, but their notion of personality is psychologically naïve. DETT combines the theoretical richness of Gratch-Marsella with the computational efficiency of EINSTein and MANA, and adds dispositions, accounting for differences in the emotional susceptibility of various agents by reasoning about the reaction of an internal agent characteristic to the external environment in which the agent is situated. We have demonstrated the basic computational cycle in implemented software and in actual wargame experiments.

The model is still an approximation. It does not implement the known effect of emotion on perception, and does not consider other possible linkages (e.g., between emotion and desire). Such simplifications are in the nature of simulation, and are justified empirically by the notion of "universality": the dynamics of a multi-agent simulation often depend more on the interactions of the agents than on the details of individual agents' reasoning [10].

5. Acknowledgements

This material is based on work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHC040153. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DARPA or the Department of Interior-National Business Center (DOI-NBC). Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

6. REFERENCES

[1] S. Brave and C. Nass. Emotion in Human-Computer Interaction. In J. A. Jacko and A. Sears, Editors, *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, pages 81-96. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 2003.

[2] S. Brueckner. *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. Thesis at Humboldt University Berlin, Department of Computer Science, 2000.

[3] C. Darwin. *The expression of the emotions in man and animals*. London, John Murray, 1872.

[4] J. Gratch and S. Marsella. A Domain-independent Framework for Modeling Emotion. *Journal of Cognitive Systems Research*, 5(4):269-306, 2004.

[5] A. Ilachinski. *Artificial War: Multiagent-based Simulation of Combat*. Singapore, World Scientific, 2004.

[6] A. Kott. Real-Time Adversarial Intelligence & Decision Making (RAID). 2004. http://dtsn.darpa.mil/ixo/programdetail.asp?progid=57.

[7] M. K. Lauren and R. T. Stephen. Map-Aware Non-uniform Automata (MANA)—A New Zealand Approach to Scenario Modelling. *Journal of Battlefield Technology*, 5(1 (March)):27ff, 2002.

[8] A. Ortony, G. L. Clore, and A. Collins. *The cognitive structure of emotions*. Cambridge, UK, Cambridge University Press, 1988.

[9] H. V. D. Parunak, S. Brueckner, R. Matthews, J. Sauter, and S. Brophy. Characterizing and Predicting Agents via Multi-Agent Evolution. Altarum Institute, Ann Arbor, MI, 2005. http://www.altarum.net/~vparunak/BEE.pdf.

[10] H. V. D. Parunak, S. Brueckner, and R. Savit. Universality in Multi-Agent Systems. In *Proceedings of Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*, pages 930-937, IEEE, 2004.

[11] H. V. D. Parunak and S. A. Brueckner. Extrapolation of the Opponent's Past Behaviors. In A. Kott and W. McEneany, Editors, *Adversarial Reasoning: Computational Approaches to Reading the Opponent's Mind*, Chapman and Hall/CRC, 2006.

[12] H. V. D. Parunak, S. A. Brueckner, R. Matthews, and J. Sauter. Pheromone Learning for Self-Organizing Agents. *IEEE SMC*, 35(3 (May)):316-326, 2005.

[13] R. W. Picard. *Affective Computing*. Cambridge, MA, MIT Press, 2000.

[14] A. S. Rao and M. P. Georgeff. Modeling Rational Agents within a BDI Architecture. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR-91)*, pages 473-484, Morgan Kaufman, 1991.

Balancing the needs of players and spectators in agent-based commentary systems

Daniel Fielding
University of Nottingham
School of Computer Science
Notingham, UK
dgf@cs.nott.ac.uk

Brian Logan
University of Nottingham
School of Computer Science
Nottingham, UK
bsl@cs.nott.ac.uk

Steve Benford
University of Nottingham
School of Computer Science
Nottingham, UK
sdb@cs.nott.ac.uk

ABSTRACT

We describe an agent-based commentary system for computer games which produces real-time and post-game commentary for spectators. Our system adopts an embodied approach in which reporter agents are directly embodied within the game and are subject to the same constraints as players in terms of perception and movement. We analyse the results of an initial evaluation of the system in terms of the conflicting needs of players and spectators, and show that there is an 'optimal' number of reporter agents that is small enough to be tolerated and even welcomed by the players while at the same time producing commentary which is judged to be sufficiently complete, accurate and interesting by the spectators.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents, Multiagent Systems*; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Artificial, augmented, and virtual realities, Evaluation/methodology*

General Terms

Human Factors, Performance

Keywords

Synthetic and Embodied Agents, Games, MMOGs, Reporting

1. INTRODUCTION

The increasing availability of broadband Internet connections has led to a rise in popularity of networked games. These range from large, complex, persistent online environments supporting a large number of simultaneous participants, e.g., Massively-Multiplayer Online Games (MMOGs), to more conventional, first person shooter games supporting a relatively small number of players. More recently, games have started to emerge as a spectator sport, with professional players competing for prestige and prizes [2, 5], and semi-professional commentators, players who takes part in a game in order to report on events in the game [1].

Commentating on games and online environments involves balancing two sets of potentially conflicting interests: spectators who want to know what is going on and players (or more generally, participants) who may wish to reveal some events while concealing others—either because an element of surprise is important in the game, or simply because there is a natural expectation of a degree of privacy, even in a shared virtual environment. At the same time, players may benefit from commentary in some situations, e.g., in-game commentary may enhance the enjoyment of the game, and appearing in reports may be a way of building a reputation and progressing in the community of players.

In previous work we have presented a framework for agent-based commentary systems based on embodied reporting agents which can produce real-time and post-game commentary which is individually tailored to the interests of the spectators and provided preliminary results on objective measures of system performance in terms of the number of events reported [3, 4]. In this paper we focus on the players' perspective. We analyse the results of an initial evaluation of our system in terms of the conflicting interests of players and spectators, and show that there is an 'optimal' number of agents which balances the needs of both groups, i.e., that is small enough to be tolerated and even welcomed by the players while at the same time producing commentary which is judged to be complete, accurate and interesting by the spectators.

2. CONTROLLING COMMENTARY

Spectators and players interact with a commentary system in different ways. Spectators want "good" commentary. This could be defined in many ways, e.g., how interesting or engagingly the commentary is, or how well presented. However, at a minimum good commentary requires that certain events be reported. Precisely which events, depends on the type of spectator and their situation. We wish to support different kinds of spectators, from those who don't play but may perhaps become spectators, to those who are actively playing. Some spectators will require real-time commentary covering key events, perhaps delivered via SMS, while others need detailed, post game commentary delivered by an animated presenter, while yet others require commentary in the game itself. A key question is how can spectators control the system to get the commentary they want in the format they want it.

Players on the other hand may wish certain events reported (e.g., when they are playing well) and certain events not reported (e.g., when they are playing badly or when an element of surprise is necessary to the game). What should or shouldn't be reported depends on the nature of the game. For example, in games in which the play-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

ers are always on display (e.g., gladiatorial combat) a lower level of privacy may be expected or even desired than in games where the demands of tactical or social play require privacy. In more unstructured virtual environments, such as SecondLife [6] the expectations of privacy may be higher still.

A key question is how can the players control the commentary, or at a minimum, the events which are recorded by the commentary system. The players' main focus is on playing the game; a commentary system which is too intrusive, e.g., one which explicitly requires a player to define before or after an event whether it is to be reported, will interfere with the gameplay. However simply recording every event in the environment is equally undesirable in environments where either a degree of privacy is required or where recording everything introduces scalability issues (e.g., MMOGs). Ideally, a commentary system needs to achieve a balance between providing a useful and engaging commentary, while at the same time minimising the degree of intrusion into the gameplay.

In our approach commentary is generated by agents that are directly embodied within the game. Being embodied means that the agents are subject to the same constraints as human players in terms of what they can sense. It also means that agents are visible to players and can be affected by their actions. Players can therefore choose whether to try to hide from the agents, or conversely approach them, perhaps 'acting up' for the camera (e.g., celebrating a victory). Our hypothesis was that embodiment and the familiar metaphor of reporters in the field would provide players with the affordances they need to control the commentary system.

3. THE STUDIO SYSTEM

Our solution, the *Studio System*, is a MAS framework in which responsibility for generating commentary is distributed between three kinds of agents: reporters, editors and presenters, which cooperate to produce commentary [3, 4].

The role of the reporters is to collect information about "interesting" events in the game world. "Interesting" events are those which are of particular interest to groups of spectators (e.g., which determine the outcome of the game or involve a particular team or player). Reporters are embodied in the virtual environment, but unlike the players cannot influence it directly. They are subject to the same sensory limitations as human players, and to obtain information about events taking place in the game world, the reporter must move to the relevant location within it. Events the reporters judge interesting are forwarded to the editors. Since reporters are not infallible, the editors attempt to verify the data produced by the reporters, e.g., by clarifying conflicting reports or by requiring multiple reporters to detect the same event. In addition, the editors attempt to maximise the collection of relevant and interesting information by assigning reporters in such a way as to provide good coverage of the events in the environment, e.g. by directing idle reporters to under-reported or interesting regions within the environment. Editors pass confirmed reports to the presenter(s). The role of the presenters is to deliver commentary to players and spectators at an appropriate time and in an appropriate format. Different presenters may have differing temporal relationships to reported events, e.g. real-time versus retrospective commentary, or may report differing degrees of detail to different output devices.

By interacting with the presenters, spectators can influence the importance the editors and reporters attach to different events, and so indirectly control what gets presented by the presenters. Editors direct the activities of reporters based on the spectators' expressed preferences and what they know of coverage and the activities of the other reporters. However, reporters are autonomous, and will follow the most interesting event they see (according to spectators' interests and assumptions about which events are interesting), notifying the editor if they abandon their allocated task.

We implemented a prototype *Studio System* based on "Capture the Flag", one of the game types provided by Unreal Tournament (UT). This is a "first person shooter" game, in which two teams of players (Red and Blue), each attempt to collect the flag from the opposing team's flag base and carry it back to their own team's flag base to score a point, while at the same time preventing the opposing team from doing the same to their flag (typically by shooting them). In a first person shooter game like UT it can be difficult for players to avoid the reporters. However the embodiment of the reporters means that they are subject to the same rules as other players; in particular they can be killed. Killing a reporter removes it from the game temporarily, and causes it to forget any information it has not passed to an editor. Killing reporters therefore gives the players a way of preventing the current state of the game being reported. To discourage players from killing reporters indiscriminately, a (small) number of points are deducted from their score for each reporter killed.

A more detailed description of the system can be found in [3, 4].

4. EVALUATION

We ran a series of online trials of our extended CTF game to determine whether the system can generate 'good enough' commentary (especially whether it can provide sufficient coverage of game events), and the impact of the system on playing the game (especially whether reporters interfere with play or otherwise change its feel). The UT server was set to six player slots; connecting players occupied these slots, and any slots unused by such connections were filled by UT's built-in bots. We varied the number of reporters present between 1, 2, 4 and 6, as we expected this to be a major factor affecting both questions. We used the relatively small CTF-Simple game map which we felt should offer the possibility of crowding even with a relatively small number of reporters present.

We invited all players to complete an online questionnaire after playing the game that probed their attitudes to the in-game commentary and the impact of reporters being present. We received 26 completed questionnaires from an estimated 45 players who took part, distributed evenly across the test conditions. We also captured logs from the UT server and from the Studio System.

4.1 Accuracy and Coverage

We assessed coverage by comparing the number of game events (flag captures, pick-ups, takes, drops and returns) that were present in the studio logs compared to the full game logs. For each event, we awarded the Studio one point if its report completely matched the UT server log in all four respects; half a point for a 'partial match' where the player was incorrect or not specified but the report was otherwise correct; and zero points otherwise. Coverage was then computed as the percentage of points awarded out of those that were potentially available. Figure 1 shows the average coverage level for each event type across all games logged on the server.[1]

With the exception of flag captures, coverage for all event types improves as more reporters are introduced into the game environment. However, this improvement becomes smaller as more reporters are added. The greatest increase is seen when going from 1 reporter to 2, with a slightly smaller increase when going from 2 to 4, and the smallest increase when going from 4 to 6. For flag-capture events, even a single reporter yields a high coverage (over 90%) because reporters can reliably deduce that a capture has taken

[1]Note that parts of the system have been reimplemented and the coverage results presented here differ from that in [3].

place (though not which player performed the flag capture) by observing the team scores.

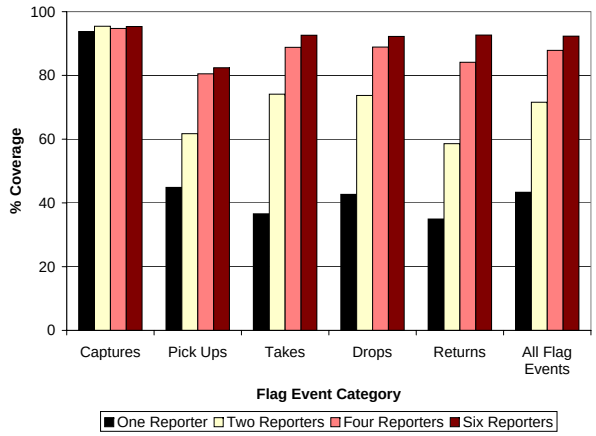


Figure 1: System coverage during live tests.

To supplement the coverage data, our exit questionnaire asked the players to assess the quality of the commentary. Our prototype system supports a number of presenters. To separate the question of coverage and accuracy from the details of presentation as much as possible, we used the "in game" presenter, which produces brief, topical reports of events in the game in the form of regular player chat. We asked players to rate whether reported events were described accurately. Responses indicate that the accuracy of the system improved with more than one reporter. We also asked them to rate whether the commentary missed out important events. Only one player felt that this was the case with only one reporter present and no players felt this was the case with six reporters present.

In summary, player feedback also suggests that the perceived coverage of the system improves as the number of reporters increases, which is in line with our objective measure of coverage. However, even with only two reporters, only one player agreed with the suggestion that the system missed a large number of events. In other words, players may find a relatively low coverage of events acceptable.

4.2 Intrusiveness

Our questionnaire also asked players to rate a number of statements about the impact of reporters on playing the game, along with free-form comments. Here we focus on the statements concerning whether reporters got in the way of movement and combat; and whether they were intrusive or annoying.

Adding more reporters results in them increasingly obstructing combat and, to a lesser extent, movement. With six reporters, four of the seven players who responded indicated strong agreement that they obstructed combat. However, one observed that they could enhance combat: "Oddly enough, they offered good meatshields. When I was carrying the flag, their presence not only took away enemy points, but helped me escape". Players did not appear to find the system particularly intrusive regardless of the number of reporters. However, some commented that the game did feel different, though not necessarily in a negative way: "It was interesting to have actual, physical spectators, much like any other 'extreme' sport much like what UT tires to portray itself as. I liked the idea of a cameraman or spectator following the action or when I grabbed the flag".

On balance, we would argue that our study shows that deploying a framework of reporter, editor and presenter agents is a viable approach to generating commentary from games. Objectively, two or more reporters were able to cover 50% or more of the events in this particular game, and players appear to have found the resulting in-game commentary to be generally appropriate and reasonably accurate. At the same time, the presence of small numbers of reporters does not appear to have had a negative impact on game play, and may even have enhanced the experience for some due to the feeling of being watched and new opportunities for tactical play.

In terms of the numbers of reporters required, there appear to be rapidly diminishing returns from using more than two in for the test map, both in terms of objective coverage and player perception, while there is increasing cost as we move towards four and six reporters in terms of negative impact on gameplay. In this particular case, we would suggest that two reporters is the most appropriate number. Of course, other maps and other games may have their own 'sweet-spots' in terms of an ideal number of reporters, but we anticipate that the general principle of diminishing returns and increasing costs should hold true.

5. CONCLUSIONS AND FUTURE WORK

We have presented a framework for reporting on computer games, the Studio System, which attempts to balance the requirements of spectators and players. The system is novel in its use of embodied reporters, agents which are directly embodied in the game world and subject to the same constraints as human players. We have described the implementation of the system and summarised the results of a preliminary study of how players experienced the system, showing that there is an 'optimal' number of agents which is small enough to be tolerated and even welcomed by the players while at the same time producing commentary which is judged to be complete, accurate and interesting by spectators.

In future we plan to extend our current basic teamwork model to approaches that can scale to larger worlds, building on previous work on teamwork, e.g., [7]. We also plan to develop embodied presenter agents that can provide engaging news feeds within virtual worlds. A final direction is further validating our framework through more in depth studies with more complex games.

6. REFERENCES

[1] BBC News. Meet the John Motsons of online gaming. February 2005. `news.bbc.co.uk/1/hi/technology/4242871.stm` (verified February 2006).

[2] Cyberathlete Professional League. `www.thecpl.com` (verified February 2006).

[3] D. Fielding, M. Fraser, B. Logan, and S. Benford. Extending game participation with embodied reporting agents. In *Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE 2004)*, pages 100–108, New York, June 2004. ACM Press.

[4] D. Fielding, M. Fraser, B. Logan, and S. Benford. Reporters, editors and presenters: Using embodied agents to report on online computer games. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*, pages 1530–1531, New York, July 2004. IEEE.

[5] National E-Sports League. `www.nationalesports.com` (verified February 2006).

[6] SecondLife. `secondlife.com` (verified February 2006).

[7] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

Architectures: BDI and MDPs

Hierarchical Planning in BDI Agent Programming Languages: A Formal Approach*

Sebastian Sardina
RMIT University
Melbourne, Australia
ssardina@cs.rmit.edu.au

Lavindra de Silva
RMIT University
Melbourne, Australia
ldesilva@cs.rmit.edu.au

Lin Padgham
RMIT University
Melbourne, Australia
linpa@cs.rmit.edu.au

ABSTRACT

This paper provides a general mechanism and a solid theoretical basis for performing planning within Belief-Desire-Intention (BDI) agents. BDI agent systems have emerged as one of the most widely used approaches to implementing intelligent behaviour in complex dynamic domains, in addition to which they have a strong theoretical background. However, these systems either do not include any built-in capacity for "lookahead" type of planning or they do it only at the implementation level without any precise defined semantics. In some situations, the ability to plan ahead is clearly desirable or even mandatory for ensuring success. Also, a precise definition of how planning can be integrated into a BDI system is highly desirable. By building on the underlying similarities between BDI systems and Hierarchical Task Network (HTN) planners, we present a formal semantics for a BDI agent programming language which cleanly incorporates HTN-style planning as a built-in feature. We argue that the resulting integrated agent programming language combines the advantages of both BDI agent systems and hierarchical offline planners.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents,Languages and structures*

Keywords

BDI agent-oriented programming, HTN planning

1. INTRODUCTION

The BDI (Belief-Desire-Intention) model is a popular and well-studied architecture of agency for intelligent agents situated in complex and dynamic environments. The model has its roots in philosophy with Bratman's [2] theory of practical reasoning and Dennett's theory of intentional systems [9]. There are a number of agent programming languages in the BDI tradition, such as PRS [13], AGENTSPEAK [19], 3APL [12], JACK [3], and CAN [24].

*We would like to acknowledge the support of Agent Oriented Software and of the Australian Research Council under the grant "Learning and Planning in BDI Agents" (number LP0560702).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

BDI agent-oriented systems are extremely flexible and responsive to the environment, and as a result, well suited for complex applications with real-time reasoning and control requirements. However, a limitation of these systems is that they normally do no *lookahead* or *planning* in the traditional sense; execution is based on a user-provided "plan library" to achieve goals. BDI frameworks rely entirely on context sensitive subgoal expansion, acting as they go. In some circumstances, however, lookahead deliberation (i.e., hypothetical reasoning) about the effects of one choice of expansion over another is clearly desirable, or even mandatory in order to guarantee goal achievability and to avoid undesired situations. In general, this is the case when (a) important resources may be used in taking actions that do not lead to a successful outcome; (b) actions are not always reversible and may lead to states from which there is no successful outcome; (c) execution of actions take substantially longer than "thinking" (or planning); and (d) actions have side effects which are undesirable if they turn out not to be useful.

In this paper, we develop a traditional BDI-style agent programming language that includes an *on-demand planning mechanism* in the style of Hierarchical Task Networks (HTN), whose semantics and implementations are well understood in the planning community [11]. The language we propose, named CANPLAN, provides a flexible approach regarding when to perform full lookahead and is provably more expressive than either BDI or HTN systems alone. CANPLAN is based on CAN [24] and AGENTSPEAK [19]. One could argue, of course, that it is always possible, in critical situations, to explicitly program lookahead within existing BDI systems. However, such code would generally be domain dependent, can be fairly complex, and would lie outside the infrastructure support provided by the BDI agent platform. Alternatively, there are many frameworks that attempt to interleave BDI-type execution with offline planning (e.g., [1, 23, 10, 17, 14]). Still, these are mostly implemented systems with no precise semantics and with little programmer control over when to plan. Our approach, instead, is to provide a formal specification of planning as a *built-in feature* of the BDI infrastructure that the programmer can use as appropriate.

The contributions of this paper are threefold. Firstly, a precise account of planning within a typical BDI agent programming language is provided. Secondly, the intrinsic relationship between lookahead planning in the context of BDI agents and the HTN approach to planning is formally explored. Lastly, the semantics of CAN given in [24] is substantially improved and simplified.

The rest of the paper is organised as follows. In section 2, we provide a brief overview of BDI agent programming languages and HTN planners; we also provide an informal discussion on their similarities. In section 3, we describe the basic BDI agent language we will use, namely the CAN notation described in [24], but with some modifications to include actions with preconditions and ef-

fects, multiple variable bindings, and a simpler, though equivalent, account of declarative goals. We chose CAN from the numerous available options because it has the desirable features of (a) combining a declarative and procedural view of *goals*, and (b) capturing the semantics of BDI failure recovery and goal persistence. In section 4, we develop CANPLAN, our new integrated account of planning and BDI execution. Besides showing some intuitively expected properties for the combined framework, we prove that, under suitable assumptions, CANPLAN's planning module reduces to HTN planning. In section 5, a brief discussion on a prototype implementation is given. Section 6 discusses related work. Finally, in section 7, we draw conclusions and outline future lines of research.

2. BDI AND HTN SYSTEMS

There are a number of BDI agent languages and HTN systems. In this section, we provide a brief abstract overview of these and comment on their similarities.

2.1 BDI Agent Programming Languages

Generally speaking, BDI agent-oriented programming languages are built around an explicit representation of beliefs, desires, and intentions. A BDI architecture addresses how these components are represented, updated, and processed to determine the agent's actions. There are a substantial number of implemented BDI systems, as well as a number of formally specified languages.

An agent consists, basically, of a belief base $\mathcal{B}$, a set of recorded pending events (goals), a plan library Π, and an intention base Γ. The *belief base* encodes the agent's knowledge about the world. The *plan library* contains *plan rules* of the form $e : \psi \leftarrow P$ encoding a *plan-body program* P for handling an event-goal e when *context* condition ψ is believed to hold. The *intention base* contains the current, partially instantiated, plans the agent has already committed to in order to handle or achieve some event-goal.

A BDI system responds to *events*, the inputs to the system, by committing to handle one pending event-goal, selecting a plan rule from the library, and placing its plan-body program into the intention base. The execution of this program may, in turn, post new sub-goal events to be achieved. If at any point a program fails, then an alternative plan rule is found and its plan-body is placed into the intention base for execution. This process repeats until a plan succeeds completely or until there are no more applicable plans, in which case failure is propagated to the event-goal.

In section 3, we shall discuss in detail one formal BDI language of this sort, namely, the CAN language [24].

2.2 HTN Planning

Hierarchical Task Network (HTN) planning is an approach to planning based on the decomposition of (high-level) *tasks* in order to accomplish an (initial) *task network*. Two examples of HTN systems include SHOP [15] and its successor SHOP2 [16]. Below, we mostly follow the definitions of HTN-planning from [11].

Tasks can be of two types. A *primitive task* is an action $act(\vec{x})$ that can be directly executed by the agent (e.g., $drive(x_1, x_2)$). A (high-level) *compound task* $e(\vec{x})$ is one that cannot be executed directly (e.g., $build\_trip(origin, dest)$). A *task network* $d = [T, \phi]$ is a collection of tasks T that need to be accomplished and a boolean formula of constraints ϕ. Constraints impose restrictions on the ordering of the tasks ($e \prec e'$), on the binding of variables ($x = x'$) and ($x = c$) (c is a constant), and on what literals must be true before or after each task (l, e), (e, l), and (e, l, e'). A *method* (e, ψ, d) encodes a way of decomposing a high-level compound task e into lower-level tasks using task network d when ψ holds. Methods provide the procedural knowledge of the domain.

An HTN *planning domain* $\mathcal{D} = (\Pi, \Lambda)$ consists of a library Π of methods and a library Λ of primitive tasks. Each primitive task in Λ is a STRIPS style action with corresponding preconditions and effects in the form of *add* and *delete* lists. An HTN *planning problem* **P** is a triple $\langle d, \mathcal{B}, \mathcal{D}\rangle$ where d is the task network to accomplish, $\mathcal{B}$ is the initial state (i.e., a set of all ground atoms that are true in $\mathcal{B}$), and $\mathcal{D}$ is a planning domain. A *plan* σ is a sequence $act_1 \cdot \ldots \cdot act_n$ of ground actions (that is, ground primitive tasks).

Given a planning problem instance **P**, the planning process involves selecting and applying an applicable reduction method from $\mathcal{D}$ to some compound task in d. This results in a new, and typically more "primitive," task network d'. This reduction process is repeated until only primitive tasks (i.e., actions) remain. If no applicable reduction can be found for a compound task at any stage, the planner "backtracks" and tries an alternative reduction for a compound task previously reduced. If all compound tasks can eventually be reduced, a plan solution σ is obtained

In [11], a clear operational semantics for HTN planning was given. The set of plans $sol(d, \mathcal{B}, \mathcal{D})$ that solves a planning instance $\mathbf{P} = \langle d, \mathcal{B}, \mathcal{D}\rangle$ is defined as $sol(d, \mathcal{B}, \mathcal{D}) = \bigcup_{n<\omega} sol_n(d, \mathcal{B}, \mathcal{D})$, where $sol_n(d, \mathcal{B}, \mathcal{D})$ is, in turn, defined as follows:

$$\begin{aligned} sol_1(d, \mathcal{B}, \mathcal{D}) &= comp(d, \mathcal{B}, \mathcal{D}), \\ sol_{n+1}(d, \mathcal{B}, \mathcal{D}) &= sol_n(d, \mathcal{B}, \mathcal{D}) \cup \bigcup_{d' \in red(d, \mathcal{B}, \mathcal{D})} sol_n(d', \mathcal{B}, \mathcal{D}). \end{aligned}$$

Intuitively, $comp(d, \mathcal{B}, \mathcal{D})$ is the set of all plan *completion* of a network d containing only primitive tasks (i.e., plans for which the constraint formula ϕ in d is satisfied), and $red(d, \mathcal{B}, \mathcal{D})$ is the set of all *reductions* of d in $\mathcal{B}$ by methods in $\mathcal{D}$. We refer to [11] for more details on HTN and its formal semantics.

2.3 Similarities Between HTN and BDI

As stated in [7], BDI agent programming languages and HTN planners share many similarities despite their different purposes. The similarities come from the knowledge used by both systems as well as from how this knowledge is manipulated to create solutions.

First of all, HTN systems and BDI languages assume an explicit representation of the agent's knowledge (i.e., the belief base or state) and a set of primitive tasks or actions that the agent can directly execute in the world. Secondly, procedural knowledge about the domain is available in both HTN and BDI systems in the form of reduction methods and plan rules, respectively. Thirdly, and most importantly, both systems create solutions by reducing higher-level entities into lower-level ones by appealing to a given set of reduction recipes. Whereas a BDI system "reduces" an event into an plan-body program using a plan rule from the plan library, an HTN planner reduces a compound task into a task network using a reduction method from the method library.

The following table gives an indication of the mapping between HTN and BDI entities.

BDI SYSTEMS	HTN SYSTEMS
belief base	state
plan library	method library
event	compound task
action	primitive task
plan-body/program	network task
plan rule	method
plan rule context	method precondition
test $?l$ in plan-body	state constraints
sequence in plan-body	ordering constraint $\prec$
parallelism in plan-body	no ordering constraint

The above table is not complete—whereas some entities have a straightforward mapping some others require a more elaborate translation (we refer to [7, 22] for a more detailed mapping).

BDI agent systems and HTN planners, despite their close relationship, differ fundamentally in their objectives. The former are focused on the *execution* of agent programs where "backtracking" can only happen in the real world. The latter, in contrast, are concerned with *hypothetical reasoning* about actions and their potential interactions within a whole plan for achieving a goal or task.

3. THE BASIC BDI LANGUAGE

The CAN (**C**onceptual **A**gent **N**otation) notation [24] is a high-level plan language in the style of typical agent languages, both in the BDI tradition and elsewhere (e.g., AGENTSPEAK [19], 3APL [12, 21], and even CONGOLOG [5, 6]). Its syntax and semantics attempt to extract the essence of a class of implementable agent platforms and could be considered as a superset of AGENTSPEAK. Unlike AGENTSPEAK, though, the semantics for CAN includes both *failure handling* and *declarative goals*—two appealing features for our planning agents.

An agent is created by the specification of a set of base beliefs $\mathcal{B}$ and a set of plans Π. The *belief base* of an agent is a set of formulas from some (knowledge representation) logical language. The programmer may choose any logical language; all that is required is for operations to exist that check whether a condition ϕ—a logical formula over the agent's beliefs—follows from a belief set (i.e., $\mathcal{B} \models \phi$), and to add and delete a belief b to and from a belief base (i.e., $\mathcal{B} \cup \{b\}$ and $\mathcal{B} \setminus \{b\}$, respectively). In practice, however, the belief base contains ground belief *atoms* in a first-order language.

As explained in section 2, an agent *plan library* Π consists of a collection of plan rules of the form $e : \psi \leftarrow P$, where e is an event and ψ is the context condition which must be true in order for the plan-body P to be applicable.[1] The *plan-body* or *program* P is built from primitive actions act that the agent can execute directly, operations to add $+b$ and delete $-b$ beliefs, tests for conditions $?\phi$, and events or (internal) achievement goals $!e$. Complex plans can be specified using sequencing $P_1; P_2$, parallelism $P_1 \| P_2$, and declarative goals $\mathsf{Goal}(\phi_s, P, \phi_f)$ (explained later). Hence, the *user language* is described by the following grammar:

$$P ::= act \mid +b \mid -b \mid ?\phi \mid !e \mid P_1; P_2 \mid P_1 \| P_2 \mid \mathsf{Goal}(\phi_s, P_1, \phi_f).$$

There are also a number of auxiliary plan forms which are used internally when assigning semantics to constructs: basic (terminating) program *nil*; and compound plans $P_1 \triangleright P_2$, which executes P_1 and then executes P_2 only if P_1 failed; and $(\!|\psi_1 : P_1, \ldots, \psi_n : P_n|\!)$, which is used to encode a set of (relevant) guarded plans. The *full language* is therefore described by the following grammar:

$$P ::= \mathit{nil} \mid act \mid ?\phi \mid +b \mid -b \mid !e \mid P_1; P_2 \mid P_1 \triangleright P_2 \mid P_1 \| P_2 \mid \mathsf{Goal}(\phi_s, P_1, \phi_f) \mid (\!|\psi_1 : P_1, \ldots, \psi_n : P_n|\!).$$

In contrast with [19, 24], we take actions as the usual basic means of the agent to change its environment and, hence, actions may have preconditions and effects. One possibility would be to follow [12] and assume that a *partial* function $\mathcal{T}$ specifying the update semantics of basic actions is given: if $\mathcal{T}(act, \mathcal{B})$ is defined, it yields the new updated belief base $\mathcal{B}'$; otherwise, we say that the action's precondition is not met in $\mathcal{B}$. However, for simplicity, we shall restrict ourselves to agents that are equipped with a (simple) STRIPS-like *action description library* Λ containing rules of the form $act : \psi_{act} \leftarrow \Phi^-_{act}; \Phi^+_{act}$, one for each action type in the domain. Formula ψ_{act} corresponds to the action's precondition, and Φ^+_{act} and Φ^-_{act} stand for the add and delete lists of atoms, respectively.[2] For example, action $move(x, y, z)$, which moves object x from y to z, could be represented in Λ as follows:

$$move(x, y, z) : Free(z) \wedge At(x, y) \longleftarrow \{Free(z), At(x, y)\}; \{Free(y), At(x, z)\}.$$

Next, we show the operational semantics for the above language along the lines of [24]. A *transition relation* $\longrightarrow$ on so-called *configurations* is defined by a set of derivation rules. A transition $C \longrightarrow C'$ specifies that executing configuration C a *single step* yields configuration C'. We write $C \longrightarrow$ to state that there exists C' such that $C \longrightarrow C'$, and $\xrightarrow{*}$ to denote the usual reflexive transitive closure of $\longrightarrow$. A derivation rule consists of a, possibly empty, set of premises, which are transitions together with some auxiliary conditions, and a single transition conclusion derivable from these premises. (see [18] for more on operational semantics).

Two types of transitions will be used to define the semantics of our agents. The first type defines what it means to execute a single intention and is defined in terms of *basic* configurations. The second type of transition is defined in terms of the first type and defines what it means to execute an agent. A *basic configuration* is a tuple $\langle \mathcal{B}, \mathcal{A}, P \rangle$ consisting of the current belief base $\mathcal{B}$ of the agent, the sequence $\mathcal{A}$ of primitive actions executed so far, and the plan-body program P being executed (i.e., the current intention).[3]

Here are some of the core derivation rules for the language:

$$\frac{\Delta = \{\psi_i\theta : P_i\theta \mid e' : \psi_i \leftarrow P_i \in \Pi \wedge \theta = \mathsf{mgu}(e, e')\}}{\langle \mathcal{B}, \mathcal{A}, !e \rangle \longrightarrow \langle \mathcal{B}, \mathcal{A}, (\!|\Delta|\!) \rangle} \; Event$$

$$\frac{\psi_i : P_i \in \Delta \quad \mathcal{B} \models \psi_i\theta}{\langle \mathcal{B}, \mathcal{A}, (\!|\Delta|\!) \rangle \longrightarrow \langle \mathcal{B}, \mathcal{A}, P_i\theta \triangleright (\!|\Delta \setminus P_i|\!) \rangle} \; Sel$$

$$\frac{\langle \mathcal{B}, \mathcal{A}, P_1 \rangle \overset{\mathsf{bdi}}{\not\longrightarrow}}{\langle \mathcal{B}, \mathcal{A}, (P_1 \triangleright P_2) \rangle \overset{\mathsf{bdi}}{\longrightarrow} \langle \mathcal{B}, \mathcal{A}, P_2 \rangle} \; \triangleright_f \qquad \frac{\mathcal{B} \models \phi\theta}{\langle \mathcal{B}, \mathcal{A}, ?\phi \rangle \longrightarrow \langle \mathcal{B}, \mathcal{A}, \mathit{nil} \rangle} \; ?$$

$$\frac{a : \psi \leftarrow \Phi^-; \Phi^+ \in \Lambda \quad a\theta = act \quad \mathcal{B} \models \psi\theta}{\langle \mathcal{B}, \mathcal{A}, act \rangle \longrightarrow \langle (\mathcal{B} \setminus \Phi^-\theta) \cup \Phi^+\theta, \mathcal{A} \cdot act, \mathit{nil} \rangle} \; act$$

$$\frac{\langle \mathcal{B}, \mathcal{A}, P_1 \rangle \longrightarrow \langle \mathcal{B}', \mathcal{A}', P' \rangle}{\langle \mathcal{B}, \mathcal{A}, (P_1; P_2) \rangle \longrightarrow \langle \mathcal{B}', \mathcal{A}', (P'; P_2) \rangle} \; Seq$$

Rule $Event$ handles achievement goal events by collecting all *relevant* plans for the event in question. Rule Sel selects one *applicable* plan from a set of (remaining) relevant plans: program $P \triangleright (\!|\Delta|\!)$ states that program P should be tried first, falling back to the remaining alternatives in Δ if necessary. Notice that plan rules' context conditions are handled in a lazy manner. Rule ? deals with test goals by checking that the condition follows from the current belief base, whereas rule act handles the case of primitive actions by using the domain action description library Λ. Rule Seq handles sequencing of programs in the usual way. Rule $\triangleright_f$ is used along with rule Sel for *failure handling*: if the current plan $P_i\theta$ for a goal fails (e.g., at some point the precondition of an action or a test goal is not met), rule $\triangleright_f$ applies first, and eventually, rule Sel may select another *applicable* alternative for the event-goal, if any.

A central distinguishing feature of CAN is its $\mathsf{Goal}(\phi_s, P, \phi_f)$ goal construct, which provides a mechanism for representing both declarative and procedural aspects of goals. Intuitively, a *goal-program* $\mathsf{Goal}(\phi_s, P, \phi_f)$ states that we should achieve the (declarative) goal ϕ_s by using (procedural) program P; failing if ϕ_f becomes true. The execution of a goal-program is consistent with

[1] An omitted ψ is equivalent to *true*. Notice that e, ψ, and P may contain free variables; a plan rule is of the form $e(\vec{x}) : \psi(\vec{x}, \vec{y}) \leftarrow P(\vec{x}, \vec{y}, \vec{z})$, where $\vec{x}, \vec{y}$ and $\vec{z}$ are vectors of (distinct) variables.

[2] Free variables in ψ_{act}, Φ^-_{act} and Φ^+_{act} are free in act too.

[3] Strictly speaking, the plan and action libraries Π and Λ should also be part of basic configurations. For legibility purposes, we omit them as they are assumed to be static entities. Configurations must also include a variable substitution θ for keeping track of all bindings done so far during the execution of a plan-body. Again, for legibility, we keep substitutions implicit in places where they need to be carried across multiple rules. See [12] on how substitutions are propagated across derivation rules for 3APL.

some desired properties of declarative goals (namely, persistent, possible, and unachieved). For instance, if P is fully executed but ϕ_s is still not true, P will be re-tried; and if ϕ_s becomes true during P's execution, the whole goal will succeed immediately.

In order to capture the desired behaviour of goal-programs, a sophisticated operational semantics was given in [24], based on explicit exceptions, a set of conditions being "watched," and derivation rules with priorities. Here, we provide an alternative much *simpler* semantics that is equivalent to the original one. The following is the new set of rules for goal-programs:

$$\frac{P \neq P_1 \rhd P_2 \quad \mathcal{B} \not\models \phi_s \vee \phi_f}{\langle \mathcal{B}, \mathcal{A}, \mathsf{Goal}(\phi_s, P, \phi_f)\rangle \longrightarrow \langle \mathcal{B}, \mathcal{A}, \mathsf{Goal}(\phi_s, P \rhd P, \phi_f)\rangle}\ \mathsf{G}_I$$

$$\frac{\mathcal{B} \models \phi_s}{\langle \mathcal{B}, \mathcal{A}, \mathsf{Goal}(\phi_s, P, \phi_f)\rangle \longrightarrow \langle \mathcal{B}, \mathcal{A}, \mathit{nil}\rangle}\ \mathsf{G}_s$$

$$\frac{\mathcal{B} \models \phi_f}{\langle \mathcal{B}, \mathcal{A}, \mathsf{Goal}(\phi_s, P, \phi_f)\rangle \longrightarrow \langle \mathcal{B}, \mathcal{A}, ?\mathit{false}\rangle}\ \mathsf{G}_f$$

$$\frac{P = P_1 \rhd P_2 \quad \mathcal{B} \not\models \phi_s \vee \phi_f \quad \langle \mathcal{B}, \mathcal{A}, P_1\rangle \longrightarrow \langle \mathcal{B}', \mathcal{A}', P'\rangle}{\langle \mathcal{B}, \mathcal{A}, \mathsf{Goal}(\phi_s, P, \phi_f)\rangle \longrightarrow \langle \mathcal{B}', \mathcal{A}', \mathsf{Goal}(\phi_s, P' \rhd P_2, \phi_f)\rangle}\ \mathsf{G}_S$$

$$\frac{P = P_1 \rhd P_2 \quad \mathcal{B} \not\models \phi_s \vee \phi_f \quad \langle \mathcal{B}, \mathcal{A}, P_1\rangle \not\longrightarrow}{\langle \mathcal{B}, \mathcal{A}, \mathsf{Goal}(\phi_s, P, \phi_f)\rangle \longrightarrow \langle \mathcal{B}, \mathcal{A}, \mathsf{Goal}(\phi_s, P_2 \rhd P_2, \phi_f)\rangle}\ \mathsf{G}_R$$

When a goal-program is first encountered during execution, rule G_I applies: G_I "initialises" the execution of a goal-program by setting the program in the goal to $P \rhd P$, where the first P is to be executed and the second P is just used to store and carry the *original* program P in case rule G_R may eventually apply later on. The second and third rules handle the cases where either the success condition ϕ_s or the failure condition ϕ_f become true. The fourth rule G_S is the one responsible for performing a single step on an already initialised goal-program. Notice that the second part in the pair $P_1 \rhd P_2$ remains constant. Finally, rule G_R restarts the original program (stored as the second program in pair $P_1 \rhd P_2$) whenever the current program has finished or got stuck, but the desired, and still possible, goal has not been achieved yet.

The above semantics for Goal is substantially simpler than the original one in [24] in that we do not appeal to explicit exceptions, "watched" conditions, or special prioritised derivation rules. Although it is not hard to prove that this alternative semantics is equivalent to the original one, due to lack of space, we do not do that here. Finally, we point out that, in the original semantics of CAN, an agent also included a goal base $\mathcal{G}$ to account for the declarative goals the agent has already committed to via goal-programs. Although not done in CAN, the goal base could potentially be used to perform (meta)reasoning about goals at the agent level execution, such as goal conflict detection and resolution ([20]). Since we are also not concerned in this paper with this type of reasoning, we completely omit the goal base from our agents.

Agent Level Execution

On top of the above basic rules, we define the evolution of an agent. An *agent configuration*, or just an agent, is a tuple of the form $\langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \Gamma\rangle$ where $\mathcal{N}$ is the agent name, Λ is an action description library, Π is a plan library, $\mathcal{B}$ is a belief base, $\mathcal{A}$ is the sequence of actions already performed by the agent, and Γ is the set of current intentions (that is, plan-body programs). Transitions between agent configurations are dictated by the following three rules:

$$\frac{P \in \Gamma \quad \langle \mathcal{B}, \mathcal{A}, P\rangle \longrightarrow \langle \mathcal{B}', \mathcal{A}', P'\rangle}{\langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \Gamma\rangle \Longrightarrow \langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}', \mathcal{A}', (\Gamma \setminus \{P\}) \cup \{P'\}\rangle}\ A_{step}$$

$$\frac{e \text{ is a new external event}}{\langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \Gamma\rangle \Longrightarrow \langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \Gamma \cup \{!e\}\rangle}\ A_{event}$$

$$\frac{P \in \Gamma \quad \langle \mathcal{B}, \mathcal{A}, P\rangle \not\longrightarrow}{\langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \Gamma\rangle \Longrightarrow \langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \Gamma \setminus \{P\}\rangle}\ A_{clean}$$

The first rule performs a single step in one intention; the second rule creates a new intention from an external event; and the last rule removes a completed intention from the intention base, that is, an intention *nil* or one that is blocked and cannot make a transition.

Next, we define the meaning of an agent execution and two related notions that will be used later in the paper.

DEFINITION 1 (BDI EXECUTION). *A BDI execution E of an agent $C_0 = \langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}_0, \mathcal{A}_0, \Gamma_0\rangle$ is a, possibly infinite, sequence of agent configurations $C_0 \cdot C_1 \cdot \ldots \cdot C_n \cdot \ldots$ such that $C_i \Longrightarrow C_{i+1}$, for every $i \geq 0$. A terminating execution is a finite execution $C_0 \cdot \ldots \cdot C_n$ where $C_n = \langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}_n, \mathcal{A}_n, \{\}\rangle$. An environment-free execution is one in which rule $\mathcal{A}_{event}$ has not been used.*

Sometimes we will be only interested in those steps of an execution where changes occur in either the executed actions or the belief of the agent—agent steps where the belief base and the executed actions remain unchanged can be disregarded. So, if $E = C_0 \cdot \ldots \cdot C_n$ is a (finite) execution, then the *derived execution* $\overline{E}$ is the sequence of configurations obtained from E by deleting all configurations C_j in the sequence such that $\mathcal{B}_j = \mathcal{B}_{j+1}$ and $\mathcal{A}_j = \mathcal{A}_{j+1}$.

In addition, we give the following notation to track an intention during an execution. If $C_0 \cdot \ldots \cdot C_n$ is a normal or derived execution and P is an intention in C_0 (i.e., $P \in \Gamma_0$), then the sequence $P_0 = P, P_1, \ldots, P_n$ denotes P's evolution within the execution and either (i) $P_i \in \Gamma_i$; or (ii) $P_i = \epsilon$, if the intention has already been removed from the intention base at some C_j, where $j \leq i$.

DEFINITION 2. *Two, possibly derived, agent executions $C_0 \cdot \ldots \cdot C_n$ and $C'_0 \cdot \ldots \cdot C'_n$ are equivalent modulo intentions iff $C'_i = \langle \mathcal{N}_i, \Lambda_i, \Pi_i, \mathcal{B}_i, \mathcal{A}_i, \Gamma'_i\rangle$, for every $0 \leq i \leq n$. Also, the two executions are equivalent modulo intentions $P_0 \in \Gamma_0$ and $P'_0 \in \Gamma'_0$ if they are equivalent modulo intentions and for every $0 \leq i \leq n$, $(\Gamma'_i \setminus \{P'_i\}) = (\Gamma_i \setminus \{P_i\})$ (where P_i (P'_i) is P_0's (P'_0's) evolution in configuration C_i (C'_i)).*

Lastly, we define what we mean by the execution of an intention and by a program (weakly) simulating another program.

DEFINITION 3 (INTENTION EXECUTION). *Let E be a BDI execution $C_0 \cdot C_1 \cdot \ldots \cdot C_n$ for an agent $C_0 = \langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}_0, \mathcal{A}_0, \Gamma_0\rangle$, where $\Gamma_0 = \Gamma'_0 \cup \{P_0\}$. Intention P_0 in C_0 has been fully executed in E if $P_n = \epsilon$; otherwise P_0 is currently executing in E. In addition, intention P_0 in C_0 has been successfully executed in E if $P_i = nil$, for some $i \leq n$; intention P_0 has failed in E if it has been fully but not successfully executed in E.*

DEFINITION 4 (PROGRAM SIMULATION). *Let E be an execution of $C = \langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \Gamma \cup \{P\}\rangle$. Program P' simulates program P in execution E iff there is an execution E' of configuration $C' = \langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \Gamma \cup \{P'\}\rangle$ such that (a) $\overline{E}$ and $\overline{E'}$ are equivalent modulo P and P'; and (b) if P has been successfully executed in E, so has P' in E'. We say that P' simulates P iff P' simulates P in every execution of any configuration.*

We have, so far, defined the necessary technical foundations for adding HTN-style planning into the CAN BDI agent language, including substantially polishing and simplifying the original CAN's semantics from [24], incorporating extra representation for actions, and providing the necessary definitions of agent execution that were not addressed in [24]. Let us now move on to the core of the paper.

4. PLANNING IN BDI SYSTEMS

In this section, we shall integrate hierarchical planning into the BDI architecture of section 3. To do so, several issues need to

be addressed. Firstly, we want to keep the language as *uniform* as possible. Secondly, we allow control over when and on what planning is to be performed within the BDI architecture. Thirdly, we need to decide what domain information the planner will use—we want the planner to re-use as much information as possible from an existing BDI specification. Lastly, the result of the planning process ought to be carried on, and possibly monitored, within the BDI execution cycle in a uniform manner.

To address the above issues, we extend the CAN language by introducing a new language construct Plan for offline lookahead planning, so that $\mathsf{Plan}(P)$, where P is a plan-body program, is intended to mean "*plan for P offline, searching for a complete hierarchical decomposition.*" In this way, the BDI agent on Plan does a full lookahead search before committing to even the first step.

As with other constructs in the language, we need to provide the operational rules for the Plan construct. To do this, we shall distinguish, from now on, between two types of transitions on basic configurations, namely, bdi and plan (labelled) transitions. We write $C \xrightarrow{\mathsf{t}} C'$ to specify a single step transition of type t (when no label is stated, both types apply). Intuitively, bdi-type steps will be used to model the normal BDI execution cycle, whereas plan-type transitions will be used to model (internal) deliberation steps within a *planning* context.

Following [6], the main operational rule states that configuration $\langle \mathcal{B}, \mathcal{A}, \mathsf{Plan}(P)\rangle$ can evolve to $\langle \mathcal{B}', \mathcal{A}', \mathsf{Plan}(P')\rangle$ provided that $\langle \mathcal{B}, \mathcal{A}, P\rangle$ can evolve to $\langle \mathcal{B}', \mathcal{A}', P'\rangle$ from where it is possible to reach a *final* configuration in a finite number of *planning* steps:

$$\frac{\langle \mathcal{B}, \mathcal{A}, P\rangle \xrightarrow{\mathsf{plan}} \langle \mathcal{B}', \mathcal{A}', P'\rangle \quad \langle \mathcal{B}', \mathcal{A}', P'\rangle \xrightarrow{\mathsf{plan}_*} \langle \mathcal{B}'', \mathcal{A}'', nil\rangle}{\langle \mathcal{B}, \mathcal{A}, \mathsf{Plan}(P)\rangle \xrightarrow{\mathsf{bdi}} \langle \mathcal{B}', \mathcal{A}', \mathsf{Plan}(P')\rangle}\ \mathsf{Plan}$$

There are also two more simple rules associated with construct Plan that are shown in Figure 1. Rule Plan_t deals with the trivial case of planning on program *nil*; and, lastly, Plan_P handles the Plan construct within a planning context.

In addition to these three derivation rules for Plan, we need to restrict the two derivation rules $\rhd_f$ and G_R from section 3 to the bdi context only. This is because failure handling and goal restarting should not be made available during planning—they are features of the BDI execution cycle only. Hence, planning is *not* merely doing lookahead on the BDI execution cycle. We refer to the new versions of the rules as $\rhd_f^{bdi}$ and G_R^{bdi}, respectively. Also, since we now have two types of transition for basic configurations, we need to slightly modify the top-level agent rules A_{step} and A_{clean} to be defined in terms of bdi-type transitions. We only show here rules $\rhd_f^{bdi}$ and A_{step} (rules G_R^{bdi} and A_{clean} should be obvious):

$$\frac{\langle \mathcal{B}, \mathcal{A}, P_1\rangle \xrightarrow{\mathsf{bdi}}\!\!\!\!\!\!/\ \ }{\langle \mathcal{B}, \mathcal{A}, (P_1 \rhd P_2)\rangle \xrightarrow{\mathsf{bdi}} \langle \mathcal{B}, \mathcal{A}, P_2\rangle}\ \rhd_f^{bdi}$$

$$\frac{P \in \Gamma \quad \langle \mathcal{B}, \mathcal{A}, P\rangle \xrightarrow{\mathsf{bdi}} \langle \mathcal{B}', \mathcal{A}', P'\rangle}{\langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \Gamma\rangle \Longrightarrow \langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}', \mathcal{A}', (\Gamma \setminus \{P\}) \cup \{P'\}\rangle}\ A_{step}$$

Observe that, with the alternative rule $\rhd_f^{bdi}$, only the BDI execution cycle would be allowed to re-try alternative plans for an event upon the failure of some failed alternative. Indeed, a program of the form $(?\mathit{false} \rhd (\!|\Delta|\!))$ has no transition within a plan context, whereas program $(\!|\Delta|\!)$ would be tried within a bdi context.

In [24], it was required that the success and failure conditions in a goal-program be mutually exclusive. There is also another sensible restriction on goal-programs, namely, that the program P provided as a method for achieving a (declarative) goal ϕ_s does not make the failure condition ϕ_f true by itself.

DEFINITION 5. *A goal-program $\mathsf{Goal}(\phi_s, P, \phi_f)$ is coherent (relative to a plan library and an action library) if for every belief bases $\mathcal{B}, \mathcal{B}', \mathcal{B}''$ and sequences of actions $\mathcal{A}, \mathcal{A}', \mathcal{A}''$ such that $\langle \mathcal{B}, \mathcal{A}, P\rangle \xrightarrow{\mathsf{bdi}_*} \langle \mathcal{B}', \mathcal{A}', P'\rangle \xrightarrow{\mathsf{bdi}_*} \langle \mathcal{B}'', \mathcal{A}'', nil\rangle$, it is the case that $\mathcal{B}' \not\models \phi_f$. An agent is coherent if every goal-program mentioned in its plan library is coherent.*

From now on, we assume that agents are *coherent*—only the environment or other concurrent intentions may make the failure condition of a goal-program true.[4] As expected, if the agent's only intention $\mathsf{Plan}(P)$ is able to start executing, then there is at least one full successful BDI execution for such intention, provided there is no intervention from the outside environment. Equally important, under the same provisions, no execution of the agent will end up failing the intention.

THEOREM 1. *Let $C = \langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \{\mathsf{Plan}(P)\}\rangle$ such that $\langle \mathcal{B}, \mathcal{A}, \mathsf{Plan}(P)\rangle \xrightarrow{\mathsf{bdi}}$. If E is an environment-free agent execution of C, then intention $\mathsf{Plan}(P)$ is either executing or has been successfully executed in E. Moreover, there is an execution E^s of C in which intention $\mathsf{Plan}(P)$ has been successfully executed in E^s.*

PROOF. *This relies on the following lemma: if $\langle \mathcal{B}, \mathcal{A}, P\rangle \xrightarrow{\mathsf{plan}_*} \langle \mathcal{B}_f, \mathcal{A}_f, nil\rangle$, then $\langle \mathcal{B}, \mathcal{A}, \mathsf{Plan}(P)\rangle \xrightarrow{\mathsf{bdi}_*} \langle \mathcal{B}_f, \mathcal{A}_f, nil\rangle$.*

On the contrary, suppose there is an environment-free execution E of the form $C_0 = C \cdot \ldots \cdot C_k$ such that $\langle \mathcal{B}_k, \mathcal{A}_k, \mathsf{Plan}(P_k)\rangle \xrightarrow{\mathsf{bdi}}\!\!\!\!\!\!/\ \ $. Observe, though, that $\langle \mathcal{B}, \mathcal{A}, \mathsf{Plan}(P)\rangle \xrightarrow{\mathsf{bdi}_k} \langle \mathcal{B}_k, \mathcal{A}_k, \mathsf{Plan}(P_k)\rangle$. By the rule Plan, $\langle \mathcal{B}, \mathcal{A}, P\rangle \xrightarrow{\mathsf{plan}_k} \langle \mathcal{B}_k, \mathcal{A}_k, P_k\rangle \xrightarrow{\mathsf{plan}_*} \langle \mathcal{B}_f, \mathcal{A}_f, nil\rangle$ and $\langle \mathcal{B}, \mathcal{A}, P\rangle \xrightarrow{\mathsf{plan}_*} \langle \mathcal{B}_f, \mathcal{A}_f, nil\rangle$ applies. By using the above lemma, we get that $\langle \mathcal{B}, \mathcal{A}, \mathsf{Plan}(P)\rangle \xrightarrow{\mathsf{bdi}_*} \langle \mathcal{B}_f, \mathcal{A}_f, nil\rangle$. Next, since $\langle \mathcal{B}, \mathcal{A}, \mathsf{Plan}(P)\rangle \xrightarrow{\mathsf{bdi}}\!\!\!\!\!\!/\ \ \langle \mathcal{B}_f, \mathcal{A}_f, nil\rangle$, there exist $\mathcal{B}'', \mathcal{A}'', P''$ such that $\langle \mathcal{B}, \mathcal{A}, \mathsf{Plan}(P)\rangle \xrightarrow{\mathsf{bdi}} \langle \mathcal{B}'', \mathcal{A}'', P''\rangle \xrightarrow{\mathsf{bdi}_*} \langle \mathcal{B}_f, \mathcal{A}_f, nil\rangle$. Thus, $\langle \mathcal{B}, \mathcal{A}, \mathsf{Plan}(P)\rangle \xrightarrow{\mathsf{bdi}}$ and the execution E cannot exist.

The second part follows easily from the fact that $\xrightarrow{\mathsf{plan}_*}$ stands for a finite chain of transitions: if the agent follows those exact transitions, P will eventually terminate successfully. □

Thus, by using the new lookahead construct $\mathsf{Plan}(P)$, the programmer can make sure—to some extent—that *failing* executions of program P will be avoided. This contrasts with the usual (default) BDI execution of P which may potentially *fail* program P due to wrong decisions at choice points. Nonetheless, it should be clear that the proposed deliberation module is *local* in the sense that it does not take into account the potential interactions with the external environment and other concurrent intentions.

Let us now focus on the relationship between our planning construct Plan and existing HTN planners. To that end, we say that a CANPLAN agent is a *bounded agent* if its belief base and all belief conditions are defined in a language which follows the same constraints as those imposed by HTN planners [11] (e.g., first-order atoms, finite domains, close world assumption). It is worth pointing out that, in practice, most existing BDI programming language implementations do actualise such constraints and deal only with bounded agents. We also assume, without loss of generality, that bounded agents do not make use of $+b$ and $-b$ statements in their plans—only primitive actions can change the belief base. ($+b$ and $-b$ statements can always be represented using special actions.)

[4] This definition is a bit too strong in that it requires a goal-program to be "sound" w.r.t. the failure condition for *every* possible belief base and *every* chain of bdi-transitions, including failed recovered executions. Even though a weaker version could be obtained with a more involved definition, we stick, for simplicity, to the above one.

$$\frac{}{\langle \mathcal{B},\mathcal{A},(nil \rhd P')\rangle \longrightarrow \langle \mathcal{B},\mathcal{A},nil\rangle}\ \rhd_t \qquad \frac{}{\langle \mathcal{B},\mathcal{A},+b\rangle \longrightarrow \langle \mathcal{B}\cup\{b\},\mathcal{A},nil\rangle}\ +b \qquad \frac{}{\langle \mathcal{B},\mathcal{A},-b\rangle \longrightarrow \langle \mathcal{B}\setminus\{b\},\mathcal{A},nil\rangle}\ -b$$

$$\frac{\langle \mathcal{B},\mathcal{A},P_1\rangle \longrightarrow \langle \mathcal{B}',\mathcal{A}',P'\rangle}{\langle \mathcal{B},\mathcal{A},(P_1 \rhd P_2)\rangle \longrightarrow \langle \mathcal{B}',\mathcal{A}',(P' \rhd P_2)\rangle}\ \rhd \qquad \frac{}{\langle \mathcal{B},\mathcal{A},(nil\,;P)\rangle \longrightarrow \langle \mathcal{B},\mathcal{A},P\rangle}\ Seq_t \qquad \frac{}{\langle \mathcal{B},\mathcal{A},(P \parallel nil)\rangle \longrightarrow \langle \mathcal{B},\mathcal{A},P\rangle}\ \|_{t_2}$$

$$\frac{\langle \mathcal{B},\mathcal{A},P_1\rangle \longrightarrow \langle \mathcal{B}',\mathcal{A}',P'\rangle}{\langle \mathcal{B},\mathcal{A},(P_1 \parallel P_2)\rangle \longrightarrow \langle \mathcal{B}',\mathcal{A}',(P' \parallel P_2)\rangle}\ \|_1 \qquad \frac{\langle \mathcal{B},\mathcal{A},P_2\rangle \longrightarrow \langle \mathcal{B}',\mathcal{A}',P'\rangle}{\langle \mathcal{B},\mathcal{A},(P_1 \parallel P_2)\rangle \longrightarrow \langle \mathcal{B}',\mathcal{A}',(P_1 \parallel P')\rangle}\ \|_2 \qquad \frac{}{\langle \mathcal{B},\mathcal{A},(nil \parallel P)\rangle \longrightarrow \langle \mathcal{B},\mathcal{A},P\rangle}\ \|_{t_1}$$

$$\frac{}{\langle \mathcal{B},\mathcal{A},\mathsf{Plan}(nil)\rangle \longrightarrow \langle \mathcal{B},\mathcal{A},nil\rangle}\ \mathsf{Plan}_t \qquad \frac{\langle \mathcal{B},\mathcal{A},P\rangle \xrightarrow{\mathsf{plan}} \langle \mathcal{B}',\mathcal{A}',P'\rangle}{\langle \mathcal{B},\mathcal{A},\mathsf{Plan}(P)\rangle \xrightarrow{\mathsf{plan}} \langle \mathcal{B}',\mathcal{A}',\mathsf{Plan}(P')\rangle}\ \mathsf{Plan}_\mathsf{P}$$

$$\Delta = \{A_{step}, A_{clean}, A_{event}\} \cup \{Event, Sel, +b, -b, act, ?, Seq, Seq_t, \rhd, \rhd_t, \rhd_f^{bdi}, \|_1, \|_2, \|_{t_1}, \|_{t_2}, \mathsf{G}_I, \mathsf{G}_s, \mathsf{G}_S, \mathsf{G}_R^{bdi}, \mathsf{Plan}, \mathsf{Plan}_t, \mathsf{Plan}_\mathsf{P}\}.$$

Figure 1: CANPLAN's complete set of rules Δ is built from the rules described in the text plus the ones shown here.

The next theorem establishes, formally, the link between the Plan construct and HTN planning. First, we prove that the new construct Plan could indeed be seen as an HTN planner. Second, we show that executions of program $\mathsf{Plan}(P)$ encode HTN plan solutions. Lastly, and not so surprisingly, we demonstrate that a straight-line HTN plan solution could be successfully executed by the BDI execution cycle. For clarity, we keep the translation between the BDI domain knowledge (i.e., libraries Π and Λ, and programs) and the HTN procedural knowledge (i.e., planning domain and task networks) implicit. (the theorem's proof is based on the relationship between the BDI's and HTN's entities as discussed in section 2.3.)

THEOREM 2. *For any bounded agent,*

1. $\langle \mathcal{B},\mathcal{A},\mathsf{Plan}(P)\rangle \xrightarrow{\mathsf{bdi}}$ *iff* $sol(P,\mathcal{B},\Pi\cup\Lambda)\neq\emptyset$.
2. $\langle \mathcal{B},\mathcal{A},\mathsf{Plan}(P)\rangle \xrightarrow{\mathsf{bdi}_*} \langle \mathcal{B}',\mathcal{A}\cdot act_1\cdot\ldots\cdot act_k,\mathsf{Plan}(P')\rangle$ *with* $k\geq 1$ *iff there exists a plan* $\sigma\in sol(P,\mathcal{B},\Pi\cup\Lambda)$*, such that* $\sigma = act_1\cdot\ldots\cdot act_k\cdot\ldots\cdot act_n$*, for some* $n\geq k$.
3. *If there exists a plan* $\sigma = act_1\cdot\ldots\cdot act_n\in sol(P,\mathcal{B},\Pi\cup\Lambda)$*, then* $\langle \mathcal{B},\mathcal{A},(act_1;\ldots;act_n)\rangle \xrightarrow{\mathsf{bdi}_*} \langle \mathcal{B}',\mathcal{A}\cdot\sigma,nil\rangle$.

Therefore, provided we restrict to the language of HTN [11], our deliberator construct Plan provides a built-in HTN planner within the whole BDI framework. The above theorem is an important practical result as it gives us the rationale for using existing HTN planner systems, such as SHOP and SHOP2 within current BDI implementations, such as AGENTSPEAK and JACK.

4.1 Planning for Declarative Goals

So far we have seen how lookahead planning can be performed on (procedural) programs. Let us now discuss how (classical) planning for a *declarative* goal ϕ_s using a procedural program P can be done. There are a few choices for this and the following five properties that we may be interested in satisfying:

(A) P is used towards the eventual satisfaction of goal ϕ_s.
(B) P may execute partially if goal ϕ_s is achieved before P completion. That is, P need not be executed completely.
(C) There is a commitment to the goal ϕ_s so that P is reinstantiated and retried until the goal in question is established.
(D) There exists a mechanism for dropping the goal when a failure condition ϕ_f becomes true.
(E) At planning time, P is solved up to the point where the goal is met. That is, it may not be required to solve P completely.

The different alternatives that we shall consider, together with the properties satisfied by each one, are depicted in the following table:

ALTERNATIVES	A	B	C	D	E
$\mathsf{Plan}(P;?\phi_s)$	$\surd$				
$\mathsf{Plan}(\mathsf{Goal}(\phi_s,P,\phi_f))$	$\surd$	$\surd$		$\surd$	
$\mathsf{Goal}(\phi_s,\mathsf{Plan}(P),\phi_f)$		$\surd$	$\surd$	$\surd$	
$\mathsf{Goal}(\phi_s,\mathsf{Plan}(P;?\phi_s),\phi_f)$	$\surd$	$\surd$	$\surd$	$\surd$	
$\mathsf{Goal}(\phi_s,\mathsf{Plan}(\mathsf{Goal}(\phi_s,P,\phi_f)),\phi_f)$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

Interestingly, a *first-principles* account of planning can easily be obtained by using the first alternative $\mathsf{Plan}(P;?\phi)$ and taking $P = !seqActions$, where the distinguished event $seqActions$ can be solved with any sequence of primitive actions.

Notice that the last four alternatives make use of the special Goal construct available in CANPLAN to handle declarative goals within the BDI execution cycle. Observe also that the last option is the only one satisfying all five properties, combining then the advantages from the BDI execution cycle and the planning module. Consequently, it is sensible to define a new language construct $\mathsf{Plan}(\phi_s,P,\phi_f)$ that the user may use, as follows:

$$\mathsf{Plan}(\phi_s,P,\phi_f) \stackrel{\text{def}}{=} \mathsf{Goal}(\phi_s,\mathsf{Plan}(\mathsf{Goal}(\phi_s,P,\phi_f)),\phi_f).$$

Among other results, it can be shown that $\mathsf{Plan}(\phi_s,P,\phi_f)$ subsumes all the executions of $\mathsf{Plan}(\mathsf{Goal}(\phi_s,P,\phi_f))$.

THEOREM 3. *For every* ϕ_s,ϕ_f *and* P*, program* $\mathsf{Plan}(\phi_s,P,\phi_f)$ *simulates program* $\mathsf{Plan}(\mathsf{Goal}(\phi_s,P,\phi_f))$.

To recap: combinations of the Plan and Goal constructs suggest an interesting range of programs for declarative goals. We believe that $\mathsf{Plan}(\phi_s,P,\phi_f)$ provides a convenient mechanism for dealing with declarative goals at both planning and execution time.

4.2 Planning vs BDI Execution

We conclude this section by exploring the differences between the execution of a planning program and the normal BDI execution. A CANPLAN$^-$ agent is a CANPLAN agent whose plan language does not include the $\|$ and Goal constructs. This restriction corresponds to classical BDI agent programming languages like AGENTSPEAK and to *total-order* HTN planners like SHOP; neither system include concurrency and goals natively. Under such restricted CANPLAN agents, the planning module is no more than a lookahead mechanism on top of the BDI execution cycle.

THEOREM 4. *Program* P *simulates program* $\mathsf{Plan}(P)$ *in every* CANPLAN$^-$ *agent.*

On the other hand, when concurrency or goal-programs are considered, performing planning may result in extra executions. In fact, it can be shown that executing $\mathsf{Plan}(\mathsf{Plan}(P_1)\|P_2)$ is equivalent to executing $\mathsf{Plan}(P_1\|P_2)$, which in turn, is very different

from executing $(\mathsf{Plan}(P_1)\|P_2)$.[5] A similar situation arises with program $\mathsf{Plan}(\mathsf{Goal}(\phi_s, \mathsf{Plan}(P), \phi_f))$. The reason, informally, is that a Plan construct is ignored within the context of another Plan construct—there is no notion of planning within planning.

Surprisingly, also, the BDI execution engine may obtain successful executions that the planner cannot produce.

THEOREM 5. *There exists an agent configuration C of the form $\langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \Gamma \cup \{P\}\rangle$ for which there is an execution where P is successfully executed, but such that no execution of $C' = \langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \Gamma \cup \{\mathsf{Plan}(P)\}\rangle$ can successfully execute* $\mathsf{Plan}(P)$.

PROOF. *Let us build a counter-example. Suppose that all actions are possible and that action act_1 just makes p true, that p and q are both false initially, and that there are only two plan rules in the plan library Π for handling event e: (i) $e : true \leftarrow act_1; ?q; act_2$; and (ii) $e : p \leftarrow act_3; act_2$. There is no solution for* $\mathsf{Plan}(!e)$*, but a BDI execution that would successfully execute $!e$ can be obtained by partially executing plan rule (i) (action act_1) and then, upon failure, fully executing plan rule (ii).* □

As one can observe, the proof's counter-example relies on both the plan failure handling mechanism built into the BDI execution cycle and the programmer not having provided a full set of plans. In fact, if the plan library in the above proof's counter-example had included a third rule of the form $e : true \leftarrow act_1; ?p; act_3; act_2$, then the planner would have found a full execution. Still, as agent programs—that is, plan libraries—are often developed incrementally and in modules, the above situation could well arise.

It follows then that the combined framework of (default) BDI execution plus local hierarchical planning is strictly more general than hierarchical planning alone. Furthermore, as discussed after Theorem 1, by using the new local planning mechanism the programmer can rule out BDI executions that are bound to fail.

5. IMPLEMENTATION ISSUES

In earlier work [8], we presented an implementation that combined BDI reasoning with HTN planning. We used JACK[6] BDI system [3] and JSHOP[7] HTN planner, a Java version of SHOP [15]. Although the integrated framework does not fully realise the operational semantics presented here, it does incorporate some important concepts from it. In particular, it allows the programmer to specify from within a JACK program the points at which JSHOP should be called, in a manner similar to the Plan construct. Consistent with the semantics of Plan, JSHOP uses the same domain representation as JACK does (i.e., the plan library Π and belief base $\mathcal{B}$). In fact, the framework builds at runtime a JSHOP planning problem representation automatically from the JACK domain knowledge.

Some differences in the implementation arise from the nature of the systems chosen for the implementation. Since JSHOP is a total-order HTN planner, it cannot accommodate the concurrent construct $\|$. However, since parallelism has benefits, the integrated framework converts JSHOP's total-order solutions into partial-order solutions so that JACK can exploit possible parallelism at execution time. Some other differences exist between the implementation and the semantics for the sake of simplicity. For example, we excluded the $\mathsf{Goal}(\phi_s, P, \phi_f)$ construct in our system, as this construct does not have a direct matching concept in JACK or JSHOP. Including this goal construct and using SHOP2 [16] to accommodate parallel execution of sub-goals natively are left for future work.

The main difference, however, is that the implementation *does not* re-plan at every step, as indicated by the Plan derivation rule defined in the semantics. This would clearly be unnecessarily inefficient. Instead, JSHOP was modified to return the relevant methods and bindings (rather than simply the actions); the BDI execution engine then follows step-by-step the decomposition suggested by the planner. Relevant environmental changes are detected by virtue of a step in the returned plan no longer being applicable within the BDI cycle. At that point, the planner is then called once again to provide an updated plan, and if none is available failure will occur in the BDI system. A disadvantage of this is that environmental changes leading to failure may be detected later in the implemented version than in the semantic rules. However, this drawback is offset by the much greater efficiency in what can be expected to be the majority of cases. This approach also has the benefit that an intention produced by a call to Plan will, in fact, *terminate*—successfully if there is no environmental interference. This is stronger than what Theorem 1 states, in which we needed to account for the strange, but potentially possible, situation where the Plan module continually returns a new and different plan prior to termination.

6. RELATED WORK

Except for INDIGOLOG [6], which is not *per se* a typical BDI agent programming language (see below), we are not aware of any other *formal* BDI-style agent programming language (e.g., 3APL [12], AGENTSPEAK [19], PRS [13]) providing a clean account of planning as we do here with CANPLAN. There are however a number of (implemented) *systems* or *frameworks* which do, in some way or another, mix planning and BDI-style execution. Some of these are *planners*, such as IPEM [1] and SAGE [14], that allow for the interleaving of action execution during the planning process. Others are *agent architectures*, such as RETSINA [17], CYPRESS and CPEF [23], and PROPICE-PLAN [10]; they are able to do lookahead planning. CYPRESS is based on the ACT [22] formalism that provides a uniform representation framework for BDI execution systems and hierarchical planners, hence supporting the type of mapping we have proposed in section 2.2. PROPICE-PLAN is perhaps the most similar system to ours, in that it is a typical BDI agent system that is able to call a planning module to find a solution for a particular problem. Like CANPLAN, a unified representation is used by both the planner and the BDI system. The work done in this paper differs, at least, in two ways form the above systems. First of all, we are particularly concerned with the *formal specification* of a BDI agent with built-in planning capabilities as well as with the *formal relation* between BDI systems and HTN planners. To our knowledge, none of the above systems come with a precise formal semantics. In some sense, however, our work was much motivated by the existence of these systems in an analogous way to how AGENTSPEAK [19] was motivated by systems like PRS [13]. Secondly, CANPLAN provides a mechanism for local deliberation *on-demand* that the programmer can use, as opposed to a fixed integration of planning within the execution engine (e.g., planning always [17] or just on (goal) failure [10, 23]).

Our work is possibly most related to that of De Giacomo and Levesque [6] in which INDIGOLOG, an incremental version of CONGOLOG [5] with a local deliberation module Σ, is proposed in the context of the situation calculus. Several ideas are taken from that work and applied to the BDI context. Our work is however different in that (i) INDIGOLOG is a cognitive agent language, with no explicit notions of events, plan library, plan selection, failure handling, intention base, etc., whereas our approach is linked to a

[5] A framework where $\mathsf{Plan}(\mathsf{Plan}(P_1)\|P_2)$ is not equivalent to $\mathsf{Plan}(P_1\|P_2)$ would require an account of HTN planning within an HTN planner. This framework can be obtained by dropping rule Plan_P and making rule Plan also available within the plan context.

[6] www.agent-software.com.au

[7] www.cs.umd.edu/projects/shop/description.html

whole family of typical BDI languages and systems; (ii) our planning mechanism is provably linked to a well understood approach in the planning community, namely HTN planning, whereas, as far as we know, the INDIGOLOG deliberator module is very general and is not directly related to any planning system; and (iii) the integration of the planning module with the notion of declarative goals in CANPLAN has no counterpart in INDIGOLOG. In some ways, our approach has a more practical orientation than that of INDIGOLOG. It would be interesting to investigate the relations between INDIGOLOG and CANPLAN (e.g., identify the BDI subclass of agents that could be written and executed in INDIGOLOG).

7. CONCLUSION AND FUTURE WORK

We have proposed a mechanism for planning within BDI systems based on the intrinsic requirements of the BDI architecture. To do so, we provided an operational semantics that substantially simplifies and extends that presented in [24] to incorporate a new *planning* construct Plan. The new construct offers power and flexibility to the BDI programmer for specifying lookahead planning points in programs. We described results showing that the integration between the planning module and the whole BDI execution is the one intuitively expected, and proved that, under suitable assumptions, the planning task reduces to HTN planning. Lastly, we showed that the combined system allows a larger set of "good" executions than the planning module alone and discussed an implementation that incorporates many of the concepts from the semantics. We believe the work presented here is a significant step towards incorporating lookahead deliberation into BDI-type agent systems in a principled manner. More importantly, it provides a firm foundation for a range of interesting further work.

The fact that we have chosen to provide planning via a new construct is very much in the spirit of BDI systems, namely, allowing for direct encoding of programmer or domain expertise. In this case, the knowledge about when planning would be beneficial. It may be argued, though, that an intelligent agent should (also) make its own decisions as to when to plan. It is therefore worth investigating a more general account in which the agent could itself take the initiative to plan; for example, when all plans for a goal fail or when there is substantial spare time. *Re-planning* following failure of a plan produced by the planner module is also a topic we have not explored here and which deserves further work (see [23]).

We have already started exploring how to accommodate extensions to classical HTN planning within our formal framework. For instance, decoupling the hierarchical structure of BDI plans and using a planning account more akin to first principles would allow for potential discovery of new plan structures. This, in turn, could provide the basis for the agent to "learn" new plans that could be added to the plan library. Also, it can be useful to plan only to a certain level of abstraction or detail, leaving further remaining decompositions to execution time or until absolutely necessary as done in [4]. Both above generalisations are likely to require (or benefit from) extra representation of effects for high-level plans. Such extra representation would also provide support for reasoning about interactions of the plan being explored with other goals and intentions of the agent [4, 20]. In particular, we would like to extend our current *local* lookahead mechanism so that the agent considers, at least, all of its own active intentions when engaging in planning.

The framework presented here provides a basis for exploring the role of declarative goals [24, 21] in planning—preliminary results were given in section 4.1 but further investigation is needed.

Lastly, it would be interesting to develop *resource-bounded* version of our planning module. To that end, we are considering developing an *anytime* or *incremental* version of the Plan construct.

8. REFERENCES

[1] J. Ambros-Ingerson. *IPEM: Integrated Planning, Execution, and Monitoring*. PhD thesis, Dept. of Computer Science, University of Essex, U.K., 1987.

[2] M. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.

[3] P. Busetta, R. Rönnquist, A. Hodgson, and A. Lucas. JACK Intelligent Agents - Components for Intelligent Agents in Java, AgentLink News Letter, Agent Oriented Software Pty. Ltd., Melbourne, January 1999.

[4] B. J. Clement and E. H. Durfee. Theory for Coordinating Concurrent Hierarchical Planning Agents Using Summary Information. In *Proc. of AAAI-99*, pages 495–502, 1999.

[5] G. De Giacomo, Y. Lespérance, and H. Levesque. ConGolog, A Concurrent Programming Language Based on the Situation Calculus. *Artificial Intelligence*, 121(1–2):109–169, 2000.

[6] G. De Giacomo and H. Levesque. An Incremental Interpreter for High-Level Programs with Sensing. In H. Levesque and F. Pirri, editors, *Logical Foundation for Cognitive Agents: contr. in honor of Ray Reiter*, pages 86–102. Springer, 1999.

[7] L. P. de Silva and L. Padgham. A Comparison of BDI Based Real-Time Reasoning and HTN Based Planning. In *Proc. of Australian Joint Conference on AI*, pages 1167–1173, 2004.

[8] L. P. de Silva and L. Padgham. Planning on Demand in BDI Systems. In *Proc. of ICAPS-05 (Poster)*, 2005.

[9] D. Dennett. *The Intentional Stance*. MIT Press, 1987.

[10] O. Despouys and F. F. Ingrand. Propice-Plan: Toward a Unified Framework for Planning and Execution. In *Proc. of European Conference on Planning*, pages 278–293, 1999.

[11] K. Erol, J. Hendler, and D. S. Nau. HTN Planning: Complexity and Expressivity. In *Proc. of AAAI-94*, pages 1123–1228, 1994.

[12] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. Agent Programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

[13] F. F. Ingrand, M. P. Georgeff, and A. S. Rao. An Architecture for Real-Time Reasoning and System Control. *IEEE Expert: Intelligent Systems and Their Applications*, 7(6):34–44,1992.

[14] C. A. Knoblock. Planning, Executing, Sensing, and Replanning for Information Gathering. In *Proc. of IJCAI-95*, pages 1686–1693, 1995.

[15] D. S. Nau, Y. Cao, A. Lotem, and H. Muñoz-Avila. SHOP: Simple Hierarchical Ordered Planner. In *Proc. of IJCAI-99*, pages 968–973, 1999.

[16] D. S. Nau, H. Muñoz-Avila, Y. Cao, A. Lotem, and S. Mitchell. Total-Order Planning with Partially Ordered Subtasks. In *Proc. of IJCAI-01*, pages 425–430, 2001.

[17] P. Paolucci, O. Shehory, K. P. Sycara, K. P. Kalp, and A. Pannu. A Planning Component for RETSINA Agents. In *Proc. of ATAL-99*, pages 147–161, 1999.

[18] G. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI-FN-19, Dept. of Computer Science Department, Aarhus University, Denmark, 1981.

[19] A. S. Rao. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In W. V. Velde and J. W. Perram, editors, *Agents Breaking Away (LNAI)*, volume 1038 of *LNAI*, pages 42–55. Springer-Verlag, 1996.

[20] J. Thangarajah, L. Padgham, and M. Winikoff. Detecting & Exploiting Positive Goal Interaction in Intelligent Agents. In *Proc. of AAMAS-03*, pages 401–408, 2003.

[21] M. B. van Riemsdijk, M. Dastani, and J.-J. Ch. Meyer. Semantics of Declarative Goals in Agent Programming. In *Proc. of AAMAS-05*, pages 133–140, 2005.

[22] D. E. Wilkins and K. L. Myers. A Common Knowledge Representation for Plan Generation and Reactive Execution. *Journal of Logic and Computation*, 5(6):731–761, 1995.

[23] D. E. Wilkins and K. L. Myers. A Multiagent Planning Architecture. In *Proc. of AIPS-98*, pages 154–162, 1998.

[24] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative & Procedural Goals in Intelligent Agent Systems. In *Proc. of KR-02*, pages 470–481, 2002.

Modular BDI Architecture

Peter Novák *
peter.novak@in.tu-clausthal.de

Jürgen Dix
dix@tu-clausthal.de

Department of Computer Science
Clausthal University of Technology
Julius-Albert-Str. 4, D-38678 Clausthal-Zellerfeld, Germany

ABSTRACT

One of the main challenges in agent-oriented programming is the design of *specialized programming languages* for single agent development. They should provide transparent interfaces to existing mainstream programming languages for easy integration with external code and legacy software. The underlying architecture of such programming languages has to be robust enough to support various approaches to knowledge representation and agent reasoning models.

In this paper we propose a modular BDI agent programming architecture, which is independent of the internal structure of its components and agent reasoning model. The connections between the components of such a BDI system are provided by interaction rules. Using this separation, we are able to draw a clear distinction between knowledge representation issues of a BDI agent system components and its dynamics.

Categories and Subject Descriptors

D.3.1 [**Programming Languages**]: Formal Definitions and Theory; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages; I.2.5 [**Artificial Intelligence**]: Programming Languages and Software; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents*

General Terms

Languages, Theory

Keywords

agent and multiagent architectures; agent programming languages; frameworks, infrastructures and environments for agent systems; BDI architecture

*Primary author of this paper is a PhD. student.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

In [15] Rao and Georgeff introduced an abstract architecture for rational agents based on BDI logic together with a proposal for a practical implementation of this architecture in a software system. Their proposal introduced three data structures for *beliefs*, *goals* and *intentions*, relations and interactions of which are governed by a set of basic axioms of rationality [14].

Apart from a mass of theoretical work inspired by this abstract architecture a number of programming languages for development of rational agents have been implemented as well. State-of-the-art BDI agent programming frameworks implementing this abstract schema fall into two main groups. On the one hand, there are agent programming platforms like Jadex [12] and JACK [9, 17], in which agent programming support is integrated into an existing programming language by addition of a layer of specialized agent oriented programming constructs. The main advantage of this approach is to enable clear and transparent integration of agent system with external and legacy software systems. Additionally, these platforms provide access to a wide range of supporting programming libraries of the underlying programming language. As the main motivation for these systems is to exploit the strengths of mainstream programming frameworks, these approaches lack a strong formal semantics defined in terms of BDI logic. Therefore a formal investigation of agent system properties relies more or less only on the semantics of the underlying programming language.

On the other hand, standalone agent programming languages like AgentSpeak(L) [3, 13] and 3APL [5, 8] provide a clear semantics of the agent system. Most of these programming languages build on the use of logic based knowledge representation and support a declarative way of encoding the dynamics of the agent system. This allows deeper theoretical insights into the behaviour and the properties of systems built with these programming languages and allows the use of formal methods like verification, or model checking. Unfortunately, they usually provide only a basic interface to external software systems. Because of this lack of integration with mainstream programming languages and various knowledge representation techniques, they still remain more theoretical tools than widely applied software development environments.

Clearly, if agent oriented programming languages are going to gain ground in software engineering in general, besides providing a solid theoretical basis for studying properties of applications built with them, they also have to support traditional software development techniques to the largest

possible extent. Therefore programming languages providing transparent integration with legacy systems and robust interfaces to external software systems are of high interest for agent oriented programming community [11].

In this paper, we address this issue and propose a modular BDI programming architecture to bridge this gap (Section 2). We describe syntax and semantics of an abstract programming language, together with an interpreter providing the dynamics of the system. The architecture consists of independent components for beliefs, desires, intentions and agent's capabilities, which are glued together by rules governing their interactions in terms of component queries and updates. This distinction provides a *separation of concerns* of knowledge representation issues for individual BDI components and the agent system dynamics encoding. It also allows the exploitation of a wide range of knowledge representation techniques, programming languages and libraries.

We do not focus specifically on development of rational agents in the traditional BDI sense. Rather we consider more general schemes, in which the rational agent reasoning model is only one of their possible specializations. We also show how various agent models can be implemented in it (Section 3) and discuss its relation to existing agent programming systems (Section 4). We conclude by pointing out directions for future work (Section 5).

2. MODULAR BDI ARCHITECTURE

The inspiration for our work was a particular aspect of the agent programming language 3APL [8], in which the interactions between BDI components are independent of the internal structure of agent's belief base. Essentially, only a query and update interfaces are important for interactions of a belief base with other components of a 3APL agent system.

Our main idea is to *generalize this independence of components* also for goals and intentions of BDI agent schema, thus allowing replacement of individual modules of an agent system as far as they provide a proper interface which integrates with the rest of the BDI system.

Our BDI agent consists of four basic components for *beliefs*, *desires*, *intentions* and *capabilities*, where the last one serves as a technical shortcut for encapsulating implementation of agent's capabilities w.r.t. its environment (i.e. agent's *sensors* and *effectors* influencing the environment). Each of these components provides an interface for *query* and *update*. Interactions between components of an agent are defined by *interaction rules* similar to usual production rules of the form: *"if query Q holds, then update U is performed"*. The dynamics of the whole system is provided by *selection* and *execution* of interaction rules by an interpreter. This is provided by an interpreter. By specialising the rule selection and imposing constraints on rule types allowed, we can implement various agent reasoning models (see Section 3).

Figure 1 depicts a schema of the proposed BDI agent system architecture. Replaceable modules of a BDI system (depicted as rectangles with round corners) provide query and update interfaces. Apart from using these basic operations, no other means of interaction between components of a BDI system is considered. When an interpreter executes an interaction rule, the query part of it (left side of a rule) is consulted, and when evaluated to true, the update part (right side) is executed. The result of such an execution is an update of a certain BDI component. In the subscript of

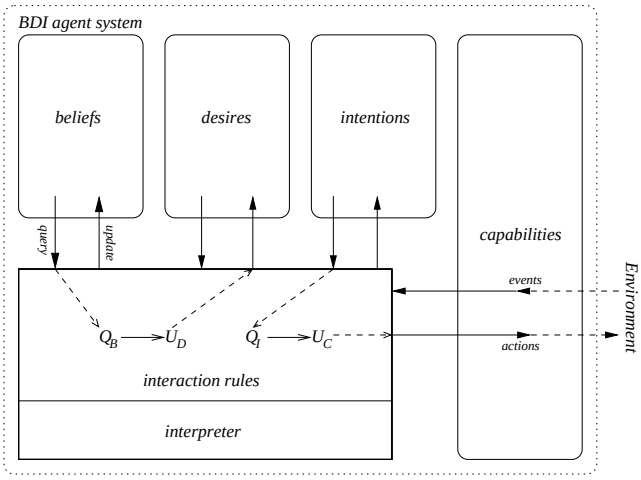


Figure 1: BDI Agent System

query and update expressions, the consulted component is indicated (depicted by dashed arrows). Agent's interaction with the outer environment is facilitated by its capabilities. Events are perceived by sensory capabilities, so that when the base of capabilities is queried, it provides information about state of the outer environment and events occurred in it. Based upon its mental state, an agent can act in its environment by updating its base of capabilities, what in turn results in performing an action by agent's effectors.

Note that we consider only the run-time representation of an agent. Design-time form, or syntax of an agent programming language associated with our framework, is not in the scope of this paper.

2.1 Syntax

In order to investigate the strengths of various approaches to knowledge representation (for implementation of a component of BDI system), we need an abstract programming language general enough to accommodate interfaces of wide range of programming languages from Prolog to C++, or LISP. Therefore we reduce the requirements on this language as much as possible and assume only the existence of generic domains and variables over them.

DEFINITION 1. ***(BDI component languages)*** *Let $D_1, \ldots, D_n$ be a set of domains and $Var_{D_1}, \ldots, Var_{D_n}$ be mutually disjoint sets of corresponding domain variables. Let $\mathcal{L}_B$, $\mathcal{L}_D$, $\mathcal{L}_I$, $\mathcal{L}_C$ be languages possibly involving elements of $D_1, \ldots, D_n$ and $Var_{D_1}, \ldots, Var_{D_n}$.*

We call $\mathcal{L}_B$, $\mathcal{L}_D$, $\mathcal{L}_I$, $\mathcal{L}_C$ BDI component languages of beliefs, desires, intentions and capabilities with typical element belief, desire, intention and capability respectively. We also assume w.l.o.g., that BDI component languages are mutually disjoint.

Domains and domain variables over them are elements of BDI component languages which can be shared between theories in these languages. Typical example would be domain of natural numbers and set of variables standing for them. From this point on, we will denote formulas of BDI component languages by φ. Formulas which do not include variables are called *ground* and those which do are *non-ground*.

From formulas of BDI component languages, query and update expressions can be constructed.

DEFINITION 2. ***(query and update)*** *Let $\mathcal{L}$ be a BDI component language. Then query and update languages $\mathcal{L}_Q$ and $\mathcal{L}_U$ are defined as follows:*

- *if $\varphi \in \mathcal{L}$, then $Q(\varphi) \in \mathcal{L}_Q$ and $U(\varphi) \in \mathcal{L}_U$,*
- *if $\phi_1, \phi_2 \in \mathcal{L}_Q$ then $\phi_1 \wedge \phi_2 \in \mathcal{L}_Q$,*
- *if $\psi_1, \psi_2 \in \mathcal{L}_U$ then $\psi_1 \circ \psi_2 \in \mathcal{L}_U$.*

EXAMPLE 1. ***(running example)*** *Consider a simple crane agent which is able to take/put a container from/to a cart. The belief base of the agent is implemented using C++ and contains the following code chunk:*

```
class CCraneBelief {
public:
  bool isHolding(int & nContID) const;
  void tookCont(int nContID);
  bool isCartPrepared() const;
  void cartArrived(bool bValue);

  int m_nContId;
  bool m_bCartOK;
  ...
};
static CCraneBelief crane;
```

Querying and updating the belief base corresponds to execution of valid C++ expression invoking the interface of object crane*. Sample query and update formulas look as follows:*

$$Q(\text{crane.isHolding}(2)) \wedge Q(\text{crane.isCartPrepared}()) \quad (1)$$

$$U(\text{crane.tookCont}(3)) \quad (2)$$

As BDI component languages can be arbitrarily expressive, we consider only conjunctions of query formulas and sequences of updates. More complex binary connectives of queries and updates are left to abilities of particular BDI component language.

Intersection of any two BDI component languages is empty, therefore each query and update formula is uniquely associated with only one BDI component language. However, for clarity, we will indicate the corresponding BDI component language in the subscript of Q and U whenever necessary. From this point on, we will usually denote query formulas by ϕ and update formulas by ψ.

Interactions between components of a BDI agent system are defined by interaction rules as follows.

DEFINITION 3. ***(interaction rules)*** *Let $\mathcal{L}_Q$ and $\mathcal{L}_U$ be query and update languages, $\phi \in \mathcal{L}_Q$ and $\psi \in \mathcal{L}_U$. We say that a rule of the form*

$$\phi \longrightarrow \psi$$

is an interaction rule.

The informal semantics of interaction rules is that of production rules. If the query ϕ is true w.r.t. given theory from a BDI component language, then the update operation defined by update formula ψ can be performed on another theory.

2.2 Semantics

The formal semantics of our BDI system is, in tradition with other BDI agent programming languages such as AgentSpeak(L) and 3APL, provided in terms of a transition system. A transition system is a set of derivation rules for deriving transitions. A transition is a transformation of one system configuration into another and it corresponds to a single computation step. The configuration of a BDI agent is a record of its current mental state at a certain point of time. Components of a BDI agent are defined as theories in given BDI component languages. Transitions between them are performed via execution of interaction rules.

DEFINITION 4. ***(BDI agent configuration)*** *A BDI agent configuration is $(\beta, \delta, \iota, \kappa)$, a tuple of theories, where theory $\beta \subseteq \mathcal{L}_B$ is a belief base of an agent, $\delta \subseteq \mathcal{L}_D$ is the base of desires, $\iota \subseteq \mathcal{L}_I$ stands for the base of intentions of it and finally $\kappa \subseteq \mathcal{L}_C$ represents the base of capabilities.*

A BDI agent configuration is a run-time representation of an agent.

Bases of beliefs, desires, intentions and capabilities are structures storing information about corresponding aspect of agent's mental state: they provide interfaces for its manipulation. Formally, they are represented as theories in corresponding BDI component languages. From now on, we will formally speak about theories in languages of individual BDI components β, δ, ι and κ and on the technical level we will use term *base* for them.

EXAMPLE 2. ***(running example cont.)*** *A belief base of the crane agent from Example 1 at a certain point in time is a state of member variables of the* crane *object. If, let's say,* m_nContId=2 *and* m_bCartOK= true*, the agent believes that it is holding a container No.* 2 *and the cart is prepared to be loaded.*

In order to evaluate queries and perform updates on BDI components we define semantical operators *Query* and *Update*.

DEFINITION 5. ***(query and update operators)*** *Let $\mathcal{L}$ be a BDI component language and $\tau \subseteq \mathcal{L}$ be a theory in that language. $\mathcal{L}_G \subseteq \mathcal{L}$ is the set of all ground formulas from $\mathcal{L}$ and $\varphi, \psi \in \mathcal{L}_G$ formulas in it .*

A query operator $Query_{\mathcal{L}}$ associated with $\mathcal{L}$ is a mapping

$$Query_{\mathcal{L}} : \mathcal{L}_G \times 2^{\mathcal{L}} \rightarrow \{true, false\}; \langle \varphi, \tau \rangle \mapsto Query_{\mathcal{L}}(\varphi, \tau)$$

Similarly, corresponding update operator $Update_{\mathcal{L}}$ is a mapping

$$Update_{\mathcal{L}} : \mathcal{L}_G \times 2^{\mathcal{L}} \rightarrow 2^{\mathcal{L}}; \langle \psi, \tau \rangle \mapsto Update_{\mathcal{L}}(\psi, \tau)$$

In the case of a query, the result is truth value of φ w.r.t. τ. The result of an update operator application is a new theory $\tau' \subseteq \mathcal{L}$ which reflects an update ψ on τ.

Query and update operators $Query_{\mathcal{L}}$ and $Update_{\mathcal{L}}$ should be computable procedures evaluating formula $\varphi \in \mathcal{L}$ against theory $\tau \subseteq \mathcal{L}$. Especially in the field of logic programming and non-monotonic reasoning, there's a vast number of theories dealing with knowledge base updates (for overview see e.g. [10]). For many of them, there are even implementations of update operators. We can exploit capabilities and strengths of these approaches for implementation of various update operators in specializations of our architecture.

EXAMPLE 3. ***(running example cont.)*** *Query and update operators for the belief base, as they were introduced in Example 1, simply evaluate the given C++ expression against current state of the belief base*[1]*. The query is true, when the return value of the executed expression is also* true *(i.e. not equal* 0*).*

By the query formula 1 from Example 1, we consult the belief base to find out whether our crane agent beliefs that it is holding container No. 2 *and at the same time also believes that the cart, into which the container is to be put, is prepared for loading. After execution of the update formula 2, the crane agent should believe that it took the container No.* 3*, so now it holds it.*

In order to define the semantics of interaction rules in terms of queries and updates of BDI components, we first need to define a semantics of these query and update formulas. We do this by relating formulas of the form $Q_{\mathcal{L}}(\varphi)$ and $U_{\mathcal{L}}(\psi)$ to applications of corresponding operators $Query_{\mathcal{L}}$ and $Update_{\mathcal{L}}$.

DEFINITION 6. ***(semantics of ground query and update formulas)*** *Let $\mathcal{L}_1, \ldots, \mathcal{L}_4$ be a quadruple of BDI component languages and $\mathcal{L}_Q$ and $\mathcal{L}_U$ be query and update languages over union of these disjoint BDI component languages. Let also $(\tau_1, \tau_2, \tau_3, \tau_4)$ be a BDI agent configuration with $\tau_i \subseteq \mathcal{L}_i$ for $1 \leq i \leq 4$. Query operator is denoted by $\models$ and $\oplus$ denotes an update operator.*

The semantics of a ground query formula $\phi \in \mathcal{L}_Q$ is defined as follows

- *if $\phi = Q(\varphi)$ and $\varphi \in \mathcal{L}_i$ then $(\tau_1, \tau_2, \tau_3, \tau_4) \models \phi$ iff $Query_{\mathcal{L}_i}(\varphi, \tau_i) = true$,*
- *if $\phi = Q(\varphi_1) \wedge \cdots \wedge Q(\varphi_n)$ for $n > 1$, then $(\tau_1, \tau_2, \tau_3, \tau_4) \models \phi$ iff for each φ_i holds $(\tau_1, \tau_2, \tau_3, \tau_4) \models Q(\varphi_i)$ for $1 \leq i \leq n$.*

and for a ground update formula $\psi \in \mathcal{L}_U$

- *if $\psi = U(\varphi)$ and $\varphi \in \mathcal{L}_i$, then $(\tau_1, \tau_2, \tau_3, \tau_4) \oplus \psi = (\tau_1', \tau_2', \tau_3', \tau_4')$ iff $Update_{\mathcal{L}_i}(\tau_i, \varphi) = \tau_i'$ and $\forall j \neq i : \tau_j' = \tau_j$,*
- *if $\psi = U(\varphi_1) \circ \cdots \circ U(\varphi_n)$ for $n > 1$, then $(\tau_1, \tau_2, \tau_3, \tau_4) \oplus \psi = (\tau_1', \tau_2', \tau_3', \tau_4')$ iff there exists a chain of configurations $(\sigma_{1,1}, \sigma_{2,1}, \sigma_{3,1}, \sigma_{4,1}), \ldots, (\sigma_{1,n+1}, \sigma_{2,n+1}, \sigma_{3,n+1}, \sigma_{4,n+1})$, such that $(\tau_1, \tau_2, \tau_3, \tau_4) = (\sigma_{1,1}, \sigma_{2,1}, \sigma_{3,1}, \sigma_{4,1})$, $(\tau_1', \tau_2', \tau_3', \tau_4') = (\sigma_{1,n+1}, \sigma_{2,n+1}, \sigma_{3,n+1}, \sigma_{4,n+1})$ and for each i, $1 \leq i \leq n$, holds $(\sigma_{1,i+1}, \sigma_{2,i+1}, \sigma_{3,i+1}, \sigma_{4,i+1}) = (\sigma_{1,i}, \sigma_{2,i}, \sigma_{3,i}, \sigma_{4,i}) \oplus U(\varphi_i)$.*

Informally, an update formula $U(\varphi_1) \circ \cdots \circ U(\varphi_n)$ is evaluated from left to right, where each successive update is applied to the result of the previous one. Since different orders of updating a configuration can lead to different results, note that the operation of updates chaining $\circ$ is not commutative.

EXAMPLE 4. ***(running example cont.)*** *Assume a BDI agent configuration with the current state of the belief base as it was described in Example 2 and semantics of Query and Update operators from Example 3. The query formula 1 from Example 1 is true and after execution of the update formula 2 the agent should believe that it is holding container* 3*. I.e. the result of this update operation is a configuration in which the belief base of an agent is modified so that* m_nContId=3.

For evaluation of a query formula against a BDI component theory by *Query* operator, the formula should be ground. Similarly for update formulas. However, non-ground formulas provide a more concise form of programming and therefore we have to deal with them. Transformation of non-ground formulas to ground ones is provided by means of *variable substitution*. Variable substitution is a mapping $\theta \subseteq \{[V/T] | V \in Var_{D_i} \wedge T \in D_i \wedge 0 \leq i \leq n\}$, where $D_1, \ldots D_n$ are domains and $Var_{D_1}, \ldots, Var_{D_n}$ sets of corresponding domain variables. By $\varphi\theta$ we denote a ground formula with all occurrences of variable $V \in Var_{D_k}$ replaced by an element $T \in D_k$, such that $[V/T] \in \theta$ for some $0 \leq k \leq n$.

Finally, for a query formula $\phi \in \mathcal{L}_Q$ we denote by $\phi\theta$ a formula $Q(\varphi\theta)$ for $\phi = Q(\varphi)$. In the case that $\phi = Q(\varphi_1) \wedge \cdots \wedge Q(\varphi_n)$, then $\phi\theta = Q(\varphi_1\theta) \wedge \cdots \wedge Q(\varphi_n\theta)$. Analogically for update formulas.

We say that variable substitution θ is ground w.r.t. formula ϕ, when $\phi\theta$ does not contain any variables.

At each time point of a system life cycle, queries of only a subset of all interaction rules hold. We say that an interaction rule $\phi \longrightarrow \psi$ is *applicable* to a BDI agent configuration $(\beta, \delta, \iota, \kappa)$ iff there exists a ground variable substitution θ, such that $(\beta, \delta, \iota, \kappa) \models \phi\theta$.

An agent system moves from one configuration to another exclusively by the application of some currently applicable interaction rule. All possible agent configurations are connected to each other via transitions from an associated transition system induced by interaction rules. More formally

DEFINITION 7. ***(agent system transition)*** *Let $\phi \longrightarrow \psi$ be an interaction rule, $(\beta, \delta, \iota, \kappa)$ a BDI agent configuration and θ a ground variable substitution w.r.t. ϕ and ψ. Then*

$$\frac{(\beta, \delta, \iota, \kappa) \models \phi\theta, (\beta, \delta, \iota, \kappa) \oplus \psi\theta = (\beta', \delta', \iota', \kappa')}{(\beta, \delta, \iota, \kappa) \longrightarrow (\beta', \delta', \iota', \kappa')}$$

We say that the rule $\phi \longrightarrow \psi$ induces this agent system transition.

Although the semantics of interaction rules is defined for ground query and update formulas, a practical programming system has to handle also non-ground interaction rules. We therefore consider the concept of *grounding on-the-fly*. Non-ground rules are grounded at the time of their execution. For this, a mechanism of variable instantiation similar to the one provided by Prolog can be used. For other languages like C, C++, or Java, mechanism of returning a value to a variable passed as an argument by reference can be exploited.

As the execution of a BDI agent starts with some initial configuration, all other possible transitions are determined by the first one. Thus a BDI agent is formally defined as follows.

DEFINITION 8. ***(BDI agent)*** *A BDI agent is a tuple $(\beta_0, \delta_0, \iota_0, \kappa_0, \mathcal{IR})$, where $(\beta_0, \delta_0, \iota_0, \kappa_0)$ is the initial configuration of an agent and $\mathcal{IR}$ is a set of interaction rules.*

Given an initial configuration of an agent system, we can see its evolution as a path within the transition system. We call such a path *computation run*.

[1]Note, that we consider here an application of a C++ *interpreter*. Therefore a given expression can be executed at any time.

DEFINITION 9. ***(computation run)*** *A computation run $Comp(s_0)$ for a BDI agent $(\beta_0, \delta_0, \iota_0, \kappa_0, \mathcal{IR})$, where $s_0 = (\beta_0, \delta_0, \iota_0, \kappa_0)$, is a finite or infinite sequence $s_0, \ldots, s_n, \ldots$ of BDI agent configurations and $\forall i \geq 0 : s_i \longrightarrow s_{i+1}$ is an agent system transition.*

The semantics of a BDI agent system is the set of all possible computation runs. This corresponds to all possible evolutions of an agent system within its transition system.

Because of non-determinism there are more than one possible evolutions of an agent system. Basically, there are two sources of non-determinism in the BDI agent system:

internal: caused by multiple applicable interaction rules w.r.t. a single configuration, and

external: coming from non-determinism of agent's environment and uncertainty of results of agent's actions in it.

While external non-determinism of an agent system cannot be influenced by an agent itself, the degree of internal is handled by agent system's interpreter.

2.3 Interpreter

What exactly is the relation between the interpreter of an agent system and the set of interaction rules? While interaction rules secure the encoding of the dynamics of the system, the interpreter is responsible for their *selection* and *execution*. In order to achieve the highest possible flexibility, we have to provide a mechanism for the fine-tuning of the interpreter.

Tuning the mechanism of rule selection can be done by means of partial ordering over interaction rules. This gives a very high degree of flexibility and provides enough space for further extensions. It is even possible to remove internal non-determinism of an agent's transition system by applying a total ordering on interaction rules. Such a system would then resemble a standard Prolog interpreter which applies rules simply in order in which they were encoded in the source program. On the other extreme side, when no ordering on interaction rules is applied, the only mechanism governing the evolution of an agent system is the content of query formulas in query parts of interaction rules. Selection of a rule to execute is then arbitrary.

DEFINITION 10. ***(interpreted run)*** *Let $(\beta_0, \delta_0, \iota_0, \kappa_0, \mathcal{IR})$ be a BDI agent and $\prec$ be a partial ordering on $\mathcal{IR}$. We say that the computation run $Comp((\beta_0, \delta_0, \iota_0, \kappa_0))$ is an interpreted run iff for each agent system transition $(\beta_i, \delta_i, \iota_i, \kappa_i) \longrightarrow (\beta_{i+1}, \delta_{i+1}, \iota_{i+1}, \kappa_{i+1})$ induced by a rule $r = \phi \longrightarrow \psi$, r is some minimal rule applicable to configuration $(\beta_i, \delta_i, \iota_i, \kappa_i)$ w.r.t. partial ordering $\prec$.*

Considering only interpreted runs of BDI agent system, BDI agent's definition could be extended by including the ordering $\prec$ to it.

EXAMPLE 5. ***(running example cont.)*** *Consider an extension of the agent introduced in Examples 1, 2 and 3.*

Let the base of desires of this agent be implemented as a simple set of string identifiers {Load(ContID),Unload(ContID)}*, standing for loading and unloading the cart. The language of desires $\mathcal{L}_D$ is Prolog. Query and update operators correspond to Prolog query execution of predicates* isGoal/1*,* addGoal/1 *and* removeGoal/1 *with the obvious semantics.*

The base of intentions is implemented as a queue of string identifiers {PerformTake(ContID),PerformPut(ContID)} *standing for raising and putting down the container. Language of intentions $\mathcal{L}_I$ is also Prolog and query/update operations correspond to execution of predicates* currentInt/1*,* queueInt/1 *and* dequeueInt/1*.*

Finally the language of capabilities $\mathcal{L}_C$ is C and query/update are implemented as for beliefs by invoking the following routines:

```
int isContainerAtLoc(int nContID);
int isCartAtLoc();
void take(int nContID);
void put(int nContID);
```

Sample interaction rules for loading a container are

$$Q_D(\mathsf{isGoal}(\mathsf{Load}(\mathsf{ContID}))) \longrightarrow U_I(\mathsf{queueInt}(\mathsf{PerformPut}(\mathsf{ContID})))$$

$$Q_I(\mathsf{currentInt}(\mathsf{PerformPut}(\mathsf{ContID}))) \wedge Q_B(\mathsf{crane.isHolding}(\mathsf{ContID}) \;\&\&\; \mathsf{crane.isCartPrepared}()) \longrightarrow U_C(\mathsf{put}(\mathsf{ContID})) \circ U_I(\mathsf{dequeueInt}(\mathsf{PerformPut}(\mathsf{ContID})))$$

Considering a precedence ordering on the rules above, in one step of the interpreter's cycle, the upper rule will be evaluated as the first. If an agent has the goal to load a container (e.g. Load(2)*$\in$belief base), then this rule will be also executed, which will cause the term* PerformPut(2) *to be appended to the end of the queue of intentions. Then the interpreter will start the cycle all over again. If the query part of the first rule is false, the interpreter will proceed to considering the second interaction rule.*

Note, that in the second rule, we exploited the internal ability of C++ implementation of belief base component to evaluate conjunctions of expressions, instead of using query of the form $Q_B \wedge Q_B$.

By further extending this basic interpreter, by e.g. allowing changes of the ordering of interaction rules on-the-fly, we can implement a very dynamic behaviour of an agent system. However, this is not in the scope of this paper and we will investigate properties of such extensions in future work.

3. AGENT REASONING MODELS

The definition of a BDI agent system as given in the last section is quite abstract: it does not make any differences between individual BDI components. This uniformity is a direct implication of the requirement of independence of BDI agent's components w.r.t. applied knowledge representation technique.

However, we believe that crucial differences of BDI system components can be reflected in *constraints* on their interactions. By allowing, or enforcing certain interactions between components of a BDI agent, we can achieve various properties of the whole system. We will show that a wide range of agent reasoning models can be implemented in our framework by regulating types of interaction rules.

3.1 I-System

Implementations of rational agent reasoning model usually descend from the original I-system [14], which axiomatizes interactions within rational agents. Informally, an agent should adopt only goals it believes to be an option w.r.t. to its beliefs (AI1). An agent should adopt intentions only in order to achieve its goals (AI2). If an agent has an intention to perform a certain action, it will eventually also perform it (AI3). It should be aware of the fact that it committed itself to certain goals and intentions (AI4, AI5). If an agent intends to achieve something, it also has to have a goal to intend it (AI6). It should be aware of its actions, i.e. if it performs certain action, it should also believe that it already performed it (AI7). And finally an agent should never hold its intentions forever, i.e. each intention must be eventually dropped (AI8).

Out of these eight basic axioms, four (AI1-AI3 and AI7) enforce certain interactions between components. These correspond to interaction rules which, following the notation introduced in Example 1, look like $Q_B \longrightarrow U_D$ (AI1), $Q_D \longrightarrow U_I$ (AI2), $Q_I \longrightarrow U_C$ (AI3) and $Q_C \longrightarrow U_B$ (AI7). Axiom AI7 is implemented here indirectly, since agents can learn about their successful actions by perceiving changes caused by them in their environments. An alternative way to implement this axiom is to keep history of actions agent tried to perform in its belief base. We can achieve this by interaction rules of type $Q_I \longrightarrow U_C \circ U_B$ (AI3).

Axioms AI4 and AI5 are secured via accessibility of all BDI components for the agent reasoning mechanism. Interaction rules can examine desire and intention bases by means of queries, so agent reasoning mechanism is "aware" of the content of its components. Axiom AI6 is satisfied by simple execution of an interaction rule of the type $Q_D \longrightarrow U_I$. Finally axiom AI8 is very vaguely specified, since "eventually" can be arbitrarily long. Realisation of this axiom is more a matter of a particular implementation of agent's base of intentions, than a feature of software agent architecture. Finally, rational agents interact with their environment by rules of the form $Q_I \longrightarrow U_C$ for performing actions and $Q_C \longrightarrow U_B$ for perceiving events.

3.2 Extensions of I-System

A closer look at the flow of information in the rational reasoning model, described in the previous subsection, reveals that on the ground of events occurring in the environment, an agent adopts beliefs, which are again the basis for adopting desires. On the ground of desires, intentions are adopted which are finally the base for agent's actions in the environment. This information flow can be informally described as a cycle of BDI components through which the information flows $C \rightarrow B \rightarrow D \rightarrow I \rightarrow C$. By inserting additional links and shortcuts, we can achieve various different agent reasoning models.

In their original paper [14], authors provide examples of *blind*, *single minded*, and *open minded* agent. Informally, a blindly committed agent maintains its intentions until it actually believes that it had achieved them. Single minded agent maintains them as long as it believes they are still possible to achieve and finally open minded agent maintains its intentions only as long as these intentions are still its goals.

In our modular BDI architecture with appropriate implementation of BDI components, a blind agent would need interaction rules for removing intentions from the intention base once it recognizes that actions based on these intentions were successful in the environment. A single minded agent cancels certain intentions with preconditions φ exactly when φ does not hold anymore. Informally, for every rule $Q_B(\varphi) \wedge \ldots \longrightarrow U_I(+\varphi')$, which introduces certain intention into the base of intentions, also the interaction rule $Q_B(\neg\varphi) \longrightarrow U_I(-\varphi')$ must occur in agent's rule base. Similarly for an open minded agent, but instead of Q_B, it would query its base of desires Q_D.

Apart from rational agents, we can also model irrational[2] agent reasoning in our framework. Consider for example a *servant* agent, to which goals can be "implanted" from outside as commands which it should follow, although it doesn't believe they are an option w.r.t. its beliefs. This can be implemented by allowing rules of the type $Q_C \longrightarrow U_D$. Informally, such a rule can be interpreted as *"upon perceiving a command C, agent should carry out C"*. A slightly stricter version of servant agent, is a *slave* agent, to which intentions could be changed and added in a similar way by interaction rule of the type $Q_C \longrightarrow U_I$.

Slave agents resemble more software objects in their traditional meaning in object oriented programming. They simply execute a requested procedure without having a "free will" not to do so. However they are still able to freely decide on later canceling this intention. As an extreme case of irrational reasoning a *religiously fanatic* agent can be considered. Such an agent doesn't want to believe facts which are conflicting its goals, or intentions. According to its current intentions and goals, it is able to change its beliefs by executing rules of the type $Q_D/Q_I \longrightarrow U_B$.

Although purely irrational agents like slave agent lack autonomy and thus do not comply with classic definitions of an agent, mixing rationality with certain degree of non-rational behaviour (w.r.t. certain aspects of agent's functionality) can lead to more efficient implementations of software agent systems. However, this claim has to be yet experimentally demonstrated.

3.3 Modular BDI Programming Platform

As we already mentioned, the main disadvantage of our modular BDI architecture is its high degree of abstraction. On the other hand, we showed that it provides also a high degree of flexibility and a lot of space for customization to particular needs.

Implementation of a software development platform based on our proposal should be based on a plug-in architecture supporting development of BDI components in various programming languages. The interpreter of interaction rules serves only as a glue between them. Each plug-in should be an interpreter for a specific programming language able to evaluate queries and perform updates on the current operational state of the component. Since nowadays many interpreters for various programming languages ranging from C++ to Prolog, LISP or Java are available, such implementation is feasible.

Because of its high flexibility, a programming platform based on our modular BDI architecture could serve as a testbed for investigating applications based on combinations of various knowledge representation techniques. We will pay special attention to applications of logic programming

[2]In the sense of *not rational w.r.t. I-System*.

techniques and especially declarative non-monotonic knowledge representation approaches like Answer Set Programming (ASP) [7] (see also [6] for the relevance of ASP in agent programming). For ASP, there's a vast amount of research on its practical applications (see e.g. [1]) as well as knowledge base updates (e.g. [10]) a lot of which can be used and practically evaluated using our system.

4. RELATED WORK

Because of its legacy, 3APL agent programming language [8] is the closest relative to our modular BDI architecture. Modular BDI schema is rather a generalization of 3APL architecture than a parallel approach. 3APL uses fixed logic based languages for implementation of BDI components. In the 3APL platform [5] Prolog is used to implement the belief base. Goal base is realized as a set of Prolog terms, and plans in the plan base are stack resembling structures of Prolog terms. In the language of our framework, 3APL allows allows only certain types of interaction rules, namely goal rules of the form $Q_D \wedge Q_B \longrightarrow U_D$, plan rules $Q_I \wedge Q_B \longrightarrow U_I$ and interaction rules $Q_D \wedge Q_B \longrightarrow U_D$ [4]. Since belief base of a 3APL agent is implemented in Prolog, all other BDI components are therefore also limited to use structures of first order logic terms.

Another important representative of BDI agent programming systems is AgentSpeak(L) [13] and its incarnation in the agent development platform Jason [3]. AgentSpeak(L) is the original implementation of already discussed I-system in a logic based language. Similarly to 3APL, it uses language of first order logic terms for belief base and stack structures for implementation of intentions. The goal base is realized via direct event processing. Contrary to 3APL, AgentSpeak(L) is even more specific in terms of queries and updates of its components. Interaction rules, or plans, are of the type $Q_D \wedge Q_B \longrightarrow U_I$, or $Q_D \wedge Q_B \longrightarrow U_D$, where updating the base of desires means firing an event. Querying it is similar to processing events from event pool which is updated according to agent's perceptions.

We see the independence of implementation of an individual BDI component of knowledge representation technique as the main advantage of our modular architecture over 3APL and AgentSpeak(L). Because of their focus on rational agent reasoning model, both 3APL and AgentSpeak(L) limit design freedom of a system programmer and enforce particular software development methodology together with a specific knowledge representation technique. Use of proprietary programming language limits 3APL and AgentSpeak(L) w.r.t. exploitation of a wide range of standard libraries for software development and external code packages. This has a strong impact on possibilities of their integration with legacy systems. On the other hand, because of the clear specification of the architecture details, like implementation languages of BDI components, systems built using 3APL or AgentSpeak(L) have probably much clearer theoretical properties w.r.t. verification of an agent system.

The other extreme of the spectrum is represented by the second family of BDI inspired programming frameworks which includes systems like JACK [17] and Jadex [12]. These are focused on integration with a mainstream programming languages (Java in both cases) and provide a high degree of freedom w.r.t. software development techniques used. However, since they both provide a particular methodology for development of internal structure of components of BDI system, integration with other knowledge representation approaches is not straightforward. Finally, they do not provide a clear semantics of systems developed with them, therefore study of theoretical properties of their applications is even more difficult than in the case of our BDI architecture.

Our modular BDI architecture proposal is positioned somewhere between the two approaches to BDI agent programming. On one hand it provides a clear operational semantics and a simple technique for implementation of agent reasoning and on the other it allows high degree of integration with existing programming languages, standard libraries and external legacy code.

5. CONCLUSION AND FUTURE WORK

In this paper we presented a modular BDI architecture which is independent of the underlying knowledge representation approach used and provides a high degree of flexibility for implementation of rational and non-rational agent reasoning models. Its core, interaction rules, are specified as the connecting element between heterogeneous components of BDI agent system. We are convinced that by using such a flexible approach, it will be easier to step from agent system analysis to its implementation and exploit strengths of mainstream programming languages and wide range of libraries provided for them.

Since the proposed modular BDI architecture was conceived to provide support for various, particularly logic based, knowledge representation techniques, our future work will be focused in this direction. We are planning to implement the interpreter for the proposed modular BDI system and a set of generic plug-ins for various knowledge representation approaches. We are interested especially in exploitation of non-monotonic reasoning and incomplete information handling techniques like Answer Set Programming [6, 7]. We will develop a set of plug-ins for programming languages like SMODELS [16] and C++ in order to study properties of agent systems built with them.

The degree of abstraction, our architecture provides, allows embedding various agent reasoning models into it (including those of 3APL or AgentSpeak(L)) and thus provides a common base for their comparison and possibly also classification. Therefore we will also study properties of various agent reasoning models in our BDI architecture and extensions of it with concepts like role of an agent and agency.

6. REFERENCES

[1] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

[2] R. H. Bordini, M. Dastani, J. Dix, and A. E. F. Seghrouchni. *Multi-Agent Programming Languages, Platforms and Applications*, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*. Kluwer Academic Publishers, 2005.

[3] R. H. Bordini, J. F. Hübner, and R. Vieira. *Jason and the Golden Fleece of Agent-Oriented Programming*, chapter 1, pages 3–37. Volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations* [2], 2005.

[4] M. Dastani, B. van Riemsdijk, F. Dignum, and J.-J. C. Meyer. A Programming Language for Cognitive Agents Goal Directed 3APL. In M. Dastani,

J. Dix, and A. E. Fallah-Seghrouchni, editors, *PROMAS*, volume 3067 of *Lecture Notes in Computer Science*, pages 111–130. Springer, 2003.

[5] M. Dastani, M. B. van Riemsdijk, and J.-J. Meyer. *Programming Multi-Agent Systems in 3APL*, chapter 2, pages 39–68. Volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations* [2], 2005.

[6] J. Dix and T. Eiter. Answer Set Programming and Agents. *AgentLink News*, Vol. 19, 2005.

[7] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *ICLP/SLP*, pages 1070–1080, 1988.

[8] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. C. Meyer. Agent Programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

[9] N. Howden, R. Rönnquist, A. Hodgson, and A. Lucas. JACK Intelligent Agents - Summary of an Agent Infrastructure. In T. Wagner and O. Rana, editors, *Infrastructure for Agents, MAS, and Scalable MAS*, 2001.

[10] J. A. Leite. *Evolving Knowledge Bases*, volume 81 of *Frontiers of Artificial Intelligence and Applications*. IOS Press, 2003.

[11] M. Luck, P. McBurney, O. Shehory, and S. Wilmott, editors. *Agent Technology Roadmap: A Roadmap for Agent Based Computing*. University of Southampton on behalf of AgentLink III, September 2005.

[12] A. Pokahr, L. Braubach, and W. Lamersdorf. *Jadex: A BDI Reasoning Engine*, chapter 6, pages 149–174. Volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations* [2], 2005.

[13] A. S. Rao. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In W. V. de Velde and J. W. Perram, editors, *MAAMAW*, volume 1038 of *Lecture Notes in Computer Science*, pages 42–55. Springer, 1996.

[14] A. S. Rao and M. P. Georgeff. Modeling Rational Agents within a BDI-Architecture. In *KR*, pages 473–484, 1991.

[15] A. S. Rao and M. P. Georgeff. An Abstract Architecture for Rational Agents. In *KR*, pages 439–449, 1992.

[16] T. Syrjänen and I. Niemelä. The SMODELS System. In T. Eiter, W. Faber, and M. Truszczynski, editors, *LPNMR*, volume 2173 of *Lecture Notes in Computer Science*, pages 434–438. Springer, 2001.

[17] M. Winikoff. *JACK(TM) Intelligent Agents: An Industrial Strength Platform*, chapter 7, pages 175–193. Volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations* [2], 2005.

Awareness as a vital ingredient of teamwork

Barbara Dunin-Kęplicz
Institute of Informatics
Warsaw University
Banacha 2, 02-097 Warsaw, Poland
Institute of Computer Science
Polish Academy of Sciences
Ordona 21, 01-237 Warsaw, Poland
keplicz@mimuw.edu.pl

Rineke Verbrugge
Department of Artificial Intelligence
University of Groningen
Grote Kruisstraat 2/1
9712 TS Groningen, The Netherlands
rineke@ai.rug.nl

ABSTRACT

For successful coordination and cooperation in a multiagent system, participants need to establish a sufficiently accurate awareness of the current situation. Awareness is understood here as a limited form of consciousness: in the minimal form, it refers to the state of an agent's beliefs about itself, about others and about the environment. When considered in the context of agents' mental states, this leads to distinguishing three levels of awareness: intra-personal (about the agent itself), inter-personal (about other agents as individuals), and group awareness.

Problems in modeling agents' awareness on all three levels are analyzed. It turns out that both the communication medium and agents' cognitive and computational limitations make the achievement of awareness difficult. Cognitive science is used to analyze and explain problems in human awareness, based on the concept of bounded rationality. The BDI framework, originally designed to formally define agents' informational and motivational attitudes, turns out to be also fruitful both for precisely formulating the problems concerning agents' awareness, and, even more importantly, for formulating and comparing possible solutions. Thus, the two fields of cognitive science and MAS mutually benefit from each other's viewpoints, especially in the light of the currently growing need for teamwork in which both computational agents and humans are involved.

In this paper, some possible avenues to solutions for defining and achieving appropriate levels of awareness are suggested. In some cases, these are concrete formal solutions, which have been adopted in our theory of collective motivational attitudes, presented in a number of conference and journal papers [5, 6, 7]. They give rise to more generic solutions that can be of use in any advanced BDI system, especially in those aiming to realize teamwork.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed AI—*multiagent systems*

General Terms

Theory, Human factors, Design

Keywords

teamwork, intra-personal awareness, inter-personal awareness, group awareness, BDI logic

1. INTRODUCTION

Multiagent systems (MAS) as a research area have been created to enable solving of complex problems, that usually are beyond capabilities of individuals (agents or humans), and which usually need expertise and/or capabilities of many different kinds. Very often these problems need *teamwork* to be solved. In the first phase of designing multiagent systems, the emphasis was put on cooperating teams of agents, while nowadays there is a growing need for teams consisting of computational agents as well as humans [22].

For teamwork to be successful, the first need for the participants is to establish a common view of the environment, which can be built by communication, reasoning and various forms of observation, for example of other agents' behavior and changes in the environment. Then, the next steps of Distributed Cooperative Problem Solving (DCPS) need to be implemented. These phases include building the entire variety of individual, social and collective motivational attitudes, as described in [4, 5, 6, 7]. In this paper we will show that in these complex processes which together constitute teamwork, agents' *awareness* about others and the environment is a vital ingredient.

Awareness is understood here as a limited form of consciousness: in the minimal form, it refers to the state of an agent's beliefs about *itself*, about *others* and about *the environment*. These aspects, when considered in the context of agents' mental states, lead to distinguishing different levels of agents' awareness: *intra-personal* (about the agent itself), *inter-personal* (about other agents as individuals), and *group awareness*. For example, an agent's awareness of itself refers to its beliefs about sentences of the form "I (do not) believe φ". In general, an agent's awareness of φ certainly implies that it does not believe that φ is false, but in

many contexts, much stronger kinds of awareness are needed (see subsection 4.1).

In practice of MAS, agents' awareness is expressed in terms of *beliefs*. In the majority of applications, agents make do with belief instead of knowledge for at least the following reasons. First, agents' perception provides the main background for beliefs. In a dynamic unpredictable environment the natural (e.g. computational) limits of perception may give rise to false beliefs or to beliefs that, while true, still cannot be fully justified by the agent. Second, communication channels may be of uncertain quality, so that even if a trustworthy sender knows a certain fact, the receiver may only believe it. In the sequel, we will use the term of awareness in a generic way, sometimes pointing out by what means it is expressed. Usually, there will be different levels of beliefs (see subsection 4.1). However, we will not go into the phase of creating those beliefs, assuming generally that perception, communication and reasoning are basic steps in this process.

This paper is meant as a voice in the discussion on the general subject of formal specification of both static and dynamic aspects of teamwork. In our considerations we will address two issues. Firstly, we will argue that agents' awareness becomes a first class citizen in contemporary multiagent applications. Secondly, we will point out problems in modeling agents' awareness, suggesting some possible avenues to solutions. In some cases, we will present a concrete formal solution. These solutions have been adapted in our theory of collective motivational attitudes, that was presented in a series of journal papers [5, 6, 7] and [1]. However we will try to show that they give rise to more generic solutions, which can be of use in any advanced BDI system, especially in those aiming to realize teamwork.

Our postulates, even though they are formulated in logic, may be particularly interesting for system developers when tailoring a multi-agent system for a specific application, especially if both computational agents and humans are involved in teamwork. When looking for possible solutions, our aim has been to forge synergy between the viewpoints on awareness within teams from the fields of cognitive science and multi-agent systems. Cognitive science has been important in analyzing and explaining problems in human awareness. The BDI framework used in MAS turns out to be fruitful both for formulating the problems around agents' awareness in a very precise way, and, even more importantly, for formulating and comparing possible solutions. Thus, the two fields mutually benefit from each other's viewpoints.

The paper is structured as follows. Section 2 focuses on agents' awareness about their own mental states, and the effects of their bounded rationality. In section 3, attention is given to problematic aspects of agents' models of other individuals' mental states. These strands come together in section 4 where we show that awareness in groups is of vital importance. We show some pitfalls in achieving it, and some possibilities for system developers to flexibly adapt the type of group awareness in a multi-agent system to the environment and the envisioned kind of organization.

2. ON INTRA-PERSONAL AWARENESS

Intra-personal awareness, or consciousness of one's own mental states, also called meta-consciousness, plays an important role in an agent's thinking and reasoning. Such introspection has for long been held as totally unproblematic:

> Consciousness was often viewed as though it was *the* defining feature of human thought. The philosophical traditions that have had the strongest influence on psychology are those of Locke and Descartes, and while these two didn't agree on much, the one proposition they shared was that cognitive states are transparent to introspection [16].

In the second half of the twentieth century, however, cognitive scientists started to study phenomena like implicit cognition, where experimental subjects could correctly recognize well-formed strings of abstract languages by learning from examples, without being able to formulate the complex underlying rule [16]. Thus, humans are often not aware of their own knowledge and beliefs. In this section, we will see how the epistemic logics that are usually used in BDI systems to model agents' knowledge do not always form an accurate model of human cognitive abilities.

2.1 Problems of logical omniscience

The standard logic of beliefs is a modal logic with possible worlds semantics. The formula $\mathrm{BEL}(i, \varphi)$ stands for "agent i believes that φ", and it is defined to be true at a possible world w if φ is true at all worlds which are belief-accessible from w. In this paper, we do not go deeper into the semantics, but see [10, 17].

In order to represent beliefs, in our logical framework for teamwork in MAS [7], we adopt the standard $KD45_n$-system for n agents as explained in [10], containing axioms and rules for a n agents named $i = 1, \ldots, n$. Similar axioms hold in the standard epistemic logic for knowledge (KNOW), except that there, A6 is replaced by the stronger A3:

A3 $\mathrm{KNOW}(i, \varphi) \rightarrow \varphi$ (Veracity of knowledge)

Here follows the axiom system $KD45_n$, in which K refers to the basic modal logic (axioms A1, A2 and rules R1, R2), D refers to axiom A6, 45 refers to axioms A4 and A5, and the subscript n refers to the number of agents:

A1 All instantiations of propositional tautologies

A2 $\mathrm{BEL}(i, \varphi) \wedge \mathrm{BEL}(i, \varphi \rightarrow \psi) \rightarrow \mathrm{BEL}(i, \psi)$ (Belief Distribution)

A4 $\mathrm{BEL}(i, \varphi) \rightarrow \mathrm{BEL}(i, \mathrm{BEL}(i, \varphi))$ (Positive Introspection)

A5 $\neg\mathrm{BEL}(i, \varphi) \rightarrow \mathrm{BEL}(i, \neg\mathrm{BEL}(i, \varphi))$ (Negative Introspection)

A6 $\neg\mathrm{BEL}(i, \bot)$ (Belief Consistency)

R1 From φ and $\varphi \rightarrow \psi$ infer ψ (Modus Ponens)

R2 From φ infer $\mathrm{BEL}(i, \varphi)$ (Belief Generalization)

A first problem for modeling humans is that, as mentioned above, they often lack positive or negative introspection into their beliefs. As a counter-example to negative introspection, one may be completely unaware of a sentence φ that one doesn't believe, and thus not believe that one does not believe it. A counter-example to positive introspection is formed by the implicit cognition experiments mentioned above. (On introspection for humans, see also [25].) In multi-agent systems, in order for agents to model themselves

properly, the developer needs to take care that a modicum of (especially positive) introspection is present.

Another problem of systems based on epistemic logic is that we have the following theorems (similar ones hold for knowledge instead of belief):

$$\models \varphi \Rightarrow \models \mathrm{BEL}(i, \varphi) \qquad \text{(Belief of valid formulas)}$$

$$\models \varphi \rightarrow \psi \Rightarrow \models \mathrm{BEL}(i, \varphi) \rightarrow \mathrm{BEL}(i, \psi) \qquad \text{(Closure under valid implication)}$$

These are examples of *logical omniscience*: agents believe all theorems, as well as all logical consequences of their beliefs. Any modal logic with standard Kripke semantics in which belief is formalized as a necessity operator has this property. Logical omniscience definitely does not apply to humans nor to computational agents, who have only limited time available. It is unrealistic to assume that they believe every logical theorem, however complicated.

Two belief-related problems of logical omniscience are:

$$\mathrm{BEL}(i, \varphi) \rightarrow \neg \mathrm{BEL}(i, \neg\varphi) \qquad \text{(Consistency of beliefs)}$$

$$\mathrm{BEL}(i, (\mathrm{BEL}(i, \varphi) \rightarrow \varphi)) \qquad \text{(Belief of having no false beliefs)}$$

The first one is problematic because the agent may believe two sentences which are in fact (equivalent to) each other's negation without the agent being aware of it. The second one (which follows from A6 and A5) makes an agent too arrogant about its beliefs: it is not aware of its limitations.

There are several possible solutions to the problems of logical omniscience, involving non-standard semantics or syntactic operators for awareness and explicit belief. Good logical references to the logical omniscience problem and its possible solutions are [17, Chapter 2] and [10, Chapter 9].

2.2 Bounded rationality

According to Herbert Simon, who coined the term *bounded rationality*, "boundedly rational agents experience limits in formulating and solving complex problems and in processing (receiving, storing, retrieving, transmitting) information" [24]. We agree with Simon that models which present humans as logically omniscient or as perfectly rational in the sense of optimizing their own utility are problematic. Moreover, we would propose to extend this discussion to software agents, because they need to reason under bounded rationality as well as humans.

Another example where one needs to be aware of limitations on agents' information occurs in a quantified setting. The distinction de re / de dicto stems from the philosophy of language [21]. A sentence of the form

$$\exists x \mathrm{BEL}(j, A(x))$$

is a *de re* belief attribution which relates j to a *res*, an individual that the belief is about. On the other hand,

$$\mathrm{BEL}(j, \exists x A(x))$$

is a *de dictum* belief attribution, relating j to a *dictum*, namely the proposition $\exists x A(x)$. Luckily, here the distinction is easily modeled in quantified modal logic, where $\exists x \mathrm{BEL}(j, A(x))$ logically implies $\mathrm{BEL}(j, \exists x A(x))$, but not vice versa. For example, it usually takes the detective a whole novel to move from "I believe (or know) that somebody murdered the victim" to "I believe (or know) of a specific suspect that he/she murdered the victim", thus from general awareness that something is wrong, to a much more specific awareness. In subsection 5.3.2, we show that developers also need to take this distinction into account when deciding what kinds of awareness are appropriate for teamwork in the given circumstances.

3. ON INTER-PERSONAL AWARENESS

Bounded rationality plays a role not only in limiting intra-personal awareness, it also constraints agents' inter-personal awareness. Formal models of human reasoning, such as those in epistemic logic and game theory, assume that humans can faultlessly reason about other people's individual knowledge and beliefs, for example in card games such as happy families [23]. However, recent research in cognitive psychology reveals that adults do not always correctly use their theory of what others know in concrete situations [15, 13].

In Keysar's experiments, some adult subjects could not correctly reason in a practical situation about another person's lack of knowledge (first-order theory of mind reasoning of the form "a does not know p") [15]. Hedden and Zhang, when describing their experiments involving a sequence of dyadic games, suggested that players generally began with first-order reasoning. When playing against first-order co-players, some began to use second-order reasoning (for example, of the form "a does not know that I know that p"), but most of them remained on the first level [13].

In recent experiments by Lisette Mol, it turns out that humans *can* learn to play a version of symmetric Mastermind involving natural language utterances such as "some colors are right". After mastering the first task, namely to play the game according to its rules, many of them learn to perform a second task, namely to develop a winning strategy for the game by using a higher-order theory of mind: "Which sentences reveal the least information while still being true?" "What does the opponent think I am trying to make him think?" [18]. Thus, some awareness of others' mental states can be learned.

3.1 Inter-personal awareness in BDI systems

Inter-personal awareness, so crucial in BDI systems, can be modeled using multi-modal logics, but some drawbacks are inevitable. Here follows a theorem of the logic of knowledge and belief related to inter-personal reasoning:

$$\mathrm{KNOW}(i, \mathrm{KNOW}(j, \varphi)) \rightarrow \mathrm{KNOW}(i, \varphi) \quad \text{(Transparency)}$$

This is not realistic, for example a child may know that her father has proved Fermat's last theorem, without knowing the theorem herself (where knowing includes being able to justify it). The transparency problem has been treated by using an alternative semantics of "local worlds" [11]. Also the following theorem may be unrealistic:

$$\mathrm{BEL}(j, \mathrm{BEL}(i, \varphi)) \rightarrow \mathrm{BEL}(j, \neg \mathrm{BEL}(i, \neg\varphi))$$

(This theorem follows from A6 and R2.) The above and similar theorems with even higher stacks of belief operators unrealistically presuppose that agents are constantly aware

that other agents monitor the consistency of their beliefs (and follow other epistemic rules) as well as they themselves do, and that others are in their turn aware of still other agents following the logical rules.

4. ON GROUP AWARENESS

When analyzing different aspects of teamwork from the viewpoint of BDI systems, one of the first purposes is to define the scope and strength of motivational and informational attitudes needed for successful team action. These determine the strength and scope of the necessary awareness within a cooperating team.

4.1 Epistemic logic for group awareness

Knowledge, which always corresponds to the facts and can be justified by a formal proof or less rigorous argumentation, is the strongest and therefore preferred informational attitude. The strongest notion of knowledge in a group is *common knowledge*, which is the basis of all conventions and the preferred basis of coordination. Halpern and Moses proved that common knowledge of certain facts is on the one hand necessary for coordination in well-known examples, while on the other hand, it cannot be established by communication if there is uncertainty about the communication channel [10].

4.1.1 General belief and common belief

Common belief is the notion of group belief which is constructed in a similar way as common knowledge, except that a collective belief among a group that φ need not imply that φ is true. Here follows a short reminder of the axioms governing group beliefs. Let $G \subseteq \{1, \ldots, n\}$ be a group. The formula E-BEL$_G(\varphi)$ (group G has φ as a *general belief*) is meant to stand for "every agent in group G believes φ":

C1 E-BEL$_G(\varphi) \leftrightarrow \bigwedge_{i \in G}$ BEL(i, φ)

C-BEL$_G(\varphi)$ (group G has φ as a *common belief*) is meant to be true if everyone in G believes φ, everyone in G believes that everyone in G believes φ, etc. Let E-BEL$^1_G(\varphi)$ be an abbreviation for E-BEL$_G(\varphi)$, and let E-BEL$^{k+1}_G(\varphi)$ for $k \geq 1$ be an abbreviation of E-BEL$_G$(E-BEL$^k_G(\varphi)$). Thus, we have intuitively C-BEL$_G(\varphi)$ iff E-BEL$^k_G(\varphi)$ for all $k \geq 1$; it turns out that this seemingly infinitary property can be captured by the following (finitary) axiom and rule.

C2 C-BEL$_G(\varphi) \leftrightarrow$ E-BEL$_G(\varphi \wedge$ C-BEL$_G(\varphi))$

RC1 From $\varphi \rightarrow$ E-BEL$_G(\psi \wedge \varphi)$ infer $\varphi \rightarrow$ C-BEL$_G(\psi)$ (Induction Rule)

The system containing individual axioms of subsection 2.1 together with the above axioms and rule is called $KD45^C_n$ [10]. Although using common belief as the intended type of awareness puts less constraints on the communication media than common knowledge, it is still logically highly complex.

4.1.2 Degrees of beliefs important in teamwork

It is well-known that for teamwork, as well as for coordination, it often does not suffice that a group of agents has a general believe about something (E-BEL$_G(\psi)$), but they should collectively believe it (C-BEL$_G(\psi)$). An example is formed by collective actions where the success of each individual agent is vital to the result, for example, lifting a heavy object together or coordinated attack. It has been proved that for such an attack to be guaranteed to succeed, the starting time of the attack must be a collective belief (even common knowledge) for the generals involved [10].

One positive feature of collective belief is that if C-BEL$_G$ holds for ψ, then C-BEL$_G$ also holds for all logical consequences of ψ. The same is true for common knowledge. Thus, agents reason in a similar way from ψ and collectively believe in this similar reasoning and the final conclusions.

In cases in which only general belief E-BEL$_G(\psi)$ has been established, it is much more difficult for agents to maintain a model of the other team members with respect to ψ and its consequences. However, establishing E-BEL$_G(\psi)$ places much less constraints on the communication medium than C-BEL$_G(\psi)$ does. In short, one could say that common knowledge and collective belief are hard to achieve, but easy to understand. Thus, the system developer's decision about the level k of group belief (E-BEL$^k_G(\psi)$) to be established, hinges on determining a good balance between communication and reasoning for a particular application.

Parikh has introduced a hierarchy of levels of knowledge between individual knowledge and common knowledge and, together with Krasucki, proved a number of interesting mathematical properties. It turns out that, due to the lack of the truth axiom, the similarly defined hierarchy between individual belief and collective belief is structurally different from the knowledge hierarchy [20].

If even limited orders of theory of mind, as in inter-personal awareness, present such difficulties for humans, it seems that creating group awareness is impossible: reasoning about common belief and common knowledge apparently involves an infinitude of levels. From the time when these notions were first studied, there has been a puzzle about their establishment and assessment, the so-called *Mutual Knowledge Paradox*, most poignantly described in [3]. How can it be that to check whether one makes a felicitous reference when saying "Have you seen the movie showing at the Roxy tonight", one has to check an infinitude of facts about reciprocal knowledge, but people seem to do this in a finite, indeed short, time? Clark's solution for human communication was that such *common ground* (common knowledge) about a sentence can be created if a number of conditions is met, namely co-presence, mutual visibility, mutual audibility, co-temporality, simultaneity, sequentiality, reviewability and revisability. Most of these conditions do not hold in multiagent systems, where agents communicate over non-instantaneous and possibly faulty communication media.

Even though common knowledge cannot in general be established by communication, we have shown that common belief can. In [8] we presented a somewhat tricky procedure that, under some assumptions about the communication channels, trust among group members and temporary persistence of some relevant beliefs (e.g. the group should be aware of the procedure), establishes a common belief C-BEL$_G(\varphi)$. The idea is essentially that one initiator first broadcasts the message φ to all agents in the group, based on a standard low-level communication protocol such as TCP, ensuring that it knows at a certain point that E-BEL$_G(\varphi)$; then the initiator broadcasts the message that C-BEL$_G(\varphi)$ to all of them. We have discussed this procedure and the needed assumptions in [8]. Intuitively, the reason that this procedure can establish common belief, whereas common knowledge can never be established, is that common beliefs need not be true.

5. TUNING AS A METHOD TO EXPRESS DEGREES OF TEAM AWARENESS

The next part of this paper refers to a larger research program, aiming at investigating and, finally, understanding the role of collective motivational attitudes in Distributed Cooperative Problem Solving (DCPS). The analysis resulted, firstly, in a *static* characterization of DCPS, with collective intention and collective commitment as central notions. (A complete theory of collective motivational attitudes in teamwork was presented in [5, 7]). Secondly, an application of this theory in a dynamic and unpredictable environment led to the reconfiguration problem, formally treated in [6].

When building the static and dynamic parts of this theory we've experienced that our theory evolved in the course of the evolution of our thinking about awareness. Over a couple of years this resulted in a generalization and relaxation of some, previously rather strong, conditions. As the role of awareness in agents' setting became more and more clear, this led us to understanding the vital need of *explicitly* expressing awareness in these systems. This conclusion resulted in a tuning schema for collective commitments ([7]), further developed for social (bilateral) commitment and collective intention in the sequel.

While investigating the role of awareness within cooperating teams of agents and humans, let us turn to the commonsense meaning of teamwork. It is clear that there are different *gradations* of being a team. For example, in a group of researchers who jointly plan their research and divide roles, and who reciprocally keep a check on how the others are doing, all aspects of teamwork are openly discussed, and team members keep each other informed about relevant changes in the plan. This non-hierarchical teamwork may be contrasted with a group of spies who all work for the same (top secret) goal. In their case a plan is designed by one mastermind, who divides the roles and divulges to each participant *solely* the information that is absolutely necessary for him to do his own part. Thus, members may not know the main goal, nor even which other agents are included in the group. In the latter example, even though the connection between members is much looser than in the former one, we would still like to speak about DCPS, albeit a non-typical case.

These two examples, showing a different gradation of being a team, differ significantly as individual and collective awareness about the ingredients of DCPS (like the main goal and the plan to achieve it) ranges from very high in the first example to very low in the second. One way to express this subtle gradation of awareness is to *characterize* its different degrees, taking into account specific circumstances (e.g. the state of the environment), as well as the application in question. In the majority of applications, different degrees of awareness could be formally expressed in terms of different kinds of beliefs, defined in the previous section.

For efficiency reasons it is often important to minimize the degree of awareness between agents, which usually allows to minimize the level of communication among agents. This degree should be tuned to the circumstances under consideration. Thus in some situations individual belief may suffice, while in other situations, general belief (E-BEL$_G$ of a relevant proposition within a group G) ensures a proper level of awareness, and again in others the strongest notion of common belief is needed. This tuning mechanism is adopted below in our definitions of collective intention, social commitment, and collective commitment. As discussed in [5, 7, 1], collective intentions and collective commitments are the two essential team attitudes that allow teamwork to happen and be successful. While collective intention may be viewed as a sort of glue that holds a group together, collective commitment reflects the concrete manner to achieve the goal in question. Social commitments, finally, represent the concrete agent-to-agent "promises" to fulfill agents' individual actions that make up the team's social plan on which the collective commitment is based.

5.1 Tuning scheme for collective intentions

In strictly cooperative teams the notion of collective intention seems to be rather strong. A necessary condition for a collective intention is that all members i of the team G have the associated individual intention $\mathrm{INT}(i, \varphi)$ towards φ. (Note that individual intentions can be modeled using possible worlds semantics based on intention-accessibility relations similar to those for beliefs.) However, the combination of individual intentions is not sufficient. Imagine that two agents want to achieve the same goal but are in a competition about this, willing to achieve it exclusively. Therefore, to exclude the case of competition, all agents should *intend* all members to have the associated individual intention, as well as the intention that all members have the individual intention, and so on; we call such a mutual intention M-INT$_G(\varphi)$. In order to formalize this condition, E-INT$_G(\varphi)$ (standing for "everyone intends") is defined first:

M1 $\text{E-INT}_G(\varphi) \leftrightarrow \bigwedge_{i \in G} \mathrm{INT}(i, \varphi)$.

The mutual intention M-INT$_G(\varphi)$ is meant to be true if everyone in G intends φ, everyone in G intends that everyone in G intends φ, etc. As we do not have infinite formulas to express this, let $\text{E-INT}^1_G(\varphi)$ be an abbreviation for E-INT$_G(\varphi)$, and let $\text{E-INT}^{k+1}_G(\varphi)$ for $k \geq 1$ be an abbreviation of $\text{E-INT}_G(\text{E-INT}^k_G(\varphi))$. Thus again we have intuitively that M-INT$_G(\varphi)$ iff $\text{E-INT}^k_G(\varphi)$ for all $k \geq 1$.

M2 $\text{M-INT}_G(\varphi) \leftrightarrow \text{E-INT}_G(\varphi \wedge \text{M-INT}_G(\varphi))$

RM1 From $\varphi \rightarrow \text{E-INT}_G(\psi \wedge \varphi)$ infer $\varphi \rightarrow \text{M-INT}_G(\psi)$ (Induction Rule)

The resulting system is called $KD_n^{\text{M-INT}_G}$, and it is sound and complete with respect to Kripke models where all n accessibility relations are serial [5].

Furthermore, all members of the team need to be aware of this mutual intention. Adding this condition completes the definition of collective intention.

M3 $\text{C-INT}_G(\varphi) \leftrightarrow \text{M-INT}_G(\varphi) \wedge \mathit{awareness}_G(\text{M-INT}_G(\varphi))$

Instantiating the above schema corresponds to tuning the $\mathit{awareness}_G$-dials from $\emptyset$, through individual beliefs and different degrees of E-BEL^k_G, to collective belief, and/or analogously for degrees of knowledge.

Even though C-INT$_G(\varphi)$ seems to be an infinite concept, collective intentions may be established in practice in a finite number of steps. As defined by **M3**, collective intentions are appropriate to model those situations in which communication, in particular announcements, work, especially if one initiator establishes the team. We have showed in detail in [4] how team formation in such an ideal case may actually

work in terms of the first two stages of collective problem solving, namely potential recognition and team formation, and how at these stages the proper attitudes are established through dialogues consisting of the appropriate speech acts.

The degree of awareness in the definition of collective intention clearly depends on the circumstances, and varies from just recognizing the situation by perception when communication is difficult or impossible, through simply confirming what situation we deal with, to more complex cases. An example of a collective intention where $awareness_G$ is instantiated as E-BEL$_G$ occurs in a situation where two persons are needed to save a third one from disaster and have severely limited means to communicate (e.g. because of a storm), but can see each other act. In this case the fact that they see each other running toward the victim may be sufficient for them to conclude that the mutual intention is present, thus M-INT$_G(\varphi) \wedge$ E-BEL$_G$(M-INT$_G(\varphi)$) is achieved. Other examples of variants of collective intentions are discussed in [5].

5.2 Tuning scheme for social commitments

In teamwork, bilateral "promises" to execute one's allocated actions in order to achieve the main goal are very important. Such a *social commitment* is not as strong as a collective one, but stronger than an individual intention. If an agent commits to a second agent to do something, then the first agent should have the *intention* to do that. Moreover, the second one should be *interested* in the first one fulfilling its intention. These two conditions (inspired by [2]), need to be enhanced by the condition expressing the agents' awareness about the situation, i.e. about their individual attitudes. In our earlier papers, such awareness was expressed in terms of collective belief [7]. In order to relax the framework to be better adaptable to the environment, here follow the schemes for defining social commitments with respect to actions α and propositions φ:

$$\begin{aligned}&\text{COMM}(i,j,\alpha) \leftrightarrow \text{INT}(i,\alpha) \wedge \text{GOAL}(j, done(i,\alpha)) \wedge \\ &\quad awareness_{\{i,j\}}(\text{INT}(i,\alpha) \wedge \text{GOAL}(j, done(i,\alpha)))\end{aligned}$$

$$\begin{aligned}&\text{COMM}(i,j,\varphi) \leftrightarrow \text{INT}(i,\varphi) \wedge \text{GOAL}(j, \texttt{stit}(i,\varphi)) \wedge \\ &\quad awareness_{\{i,j\}}(\text{INT}(i,\varphi) \wedge \text{GOAL}(j, \texttt{stit}(i,\varphi)))\end{aligned}$$

Here $done(i,\alpha)$ means that agent i has just executed action α and $\texttt{stit}(i,\varphi)$ means that agent i sees to it (takes care) that φ is achieved. For a formal treatment of *done* and `stit`, see [6]. Instantiating the above schema again corresponds to tuning the $awareness_G$-dials from $\emptyset$, through individual beliefs and different degrees of E-BEL$_G^k$, to collective belief, and/or analogously for degrees of knowledge.

5.3 Tuning scheme for collective commitments

After a group is constituted on the basis of collective intention, as a next step a *collective commitment* between the team members may be established. While a collective intention may be viewed as an inspiration for team activity, the plan-based collective commitment reflects the concrete manner of achieving the intended goal by the team. This concrete manner is provided by planning, and hinges on the allocation of actions according to an adopted plan. The allocation is concluded when agents accept social commitments to realize their individual actions. We claim that it is important for system developers to make appropriate decisions about the type or gradation of teamwork, and consequently the type of agents' awareness, needed to achieve the goal in given circumstances. Thus, it is useful for them to have a mechanism that helps to choose the corresponding type of group commitment. See [1] for a thorough discussion on this matter.

5.3.1 General schema of collective commitment

Clearly, the two examples in the introduction to section 5 cannot be covered by *one* generic type of collective commitment. In [7], we give a generic method for the system developer to tune the type of collective commitment to the application in question, the organizational structure of the group or institution, and to the environment, especially to its communicative possibilities. This generic solution allows to provide a full range of types of collective commitments and weaker group attitudes covering the range from proper teams to more loosely connected groups involved in DCPS. In our generic description we define solely basic ingredients constituting collective commitments, leaving room for case-specific extensions. The obligatory ingredients are related to different aspects of teamwork:

1. Mutual intention M-INT$_G(\varphi)$ between a group of agents.

   The team is *based* on this attitude, and exists as long as the mutual intention exists. Thus, no teamwork is considered without a mutual intention among team members.

2. Social plan P on which a collective commitment will be based.

   The social plan provides a concrete manner for the team to collectively achieve the goal φ. The predicate $cons(\varphi, P)$ informally stands for "P is a correct social plan to achieve φ".

   For definitions of social plans and $cons(\varphi, P)$, see [6]. Here it is enough to say that social plan P consists of actions to be executed sequentially or in parallel.

3. Pairwise social commitments COMM(i,j,α) for actions from the plan.

   The group splits the tasks according to social plan P, and each agent takes on responsibility to do its part by accepting relevant social commitments (see 5.2).

Next to the above ingredients, different degrees of awareness about them may be present in a team. Thus, a general schema covering different types of collective commitment is the following, where the conjuncts between curly brackets may be present or not, according to the position of the awareness 'dial':

$$\begin{aligned}&\text{C-COMM}_{G,P}(\varphi) \leftrightarrow \\ &\quad \text{M-INT}_G(\varphi) \ \wedge \ \{awareness_G(\text{M-INT}_G(\varphi))\} \wedge \\ &\quad cons(\varphi, P) \ \wedge \ \{awareness_G(cons(\varphi, P))\} \wedge \\ &\quad \bigwedge_{\alpha \in P} \bigvee_{i,j \in G} \text{COMM}(i,j,\alpha) \wedge \\ &\quad \{awareness_G(\bigwedge_{\alpha \in P} \bigvee_{i,j \in G} \text{COMM}(i,j,\alpha))\}\end{aligned}$$

In words, group G has a collective commitment to achieve overall goal φ based on social plan P (C-COMM$_{G,P}(\varphi)$) iff all of the following hold. The group mutually intends φ (with or without being aware); moreover, successful execution of social plan P leads to φ ($cons(\varphi, P)$) (with or without the group being aware of this); and finally, for every one of the actions α from a plan P, there should be one agent in the group who is socially committed to at least one (mostly other) agent in the group to fulfil the action (COMM(i, j, α)) (with or without the group being aware of this).

Instantiating the above schema again corresponds to tuning the *awareness*$_G$-dials from $\emptyset$, through individual beliefs and different degrees of general belief, to collective belief, and/or analogously for degrees of knowledge.

5.3.2 Different aspects of agents' awareness

The notion of collective commitment, whatever strength of it is considered, combines essentially different aspects of teamwork: strictly technical ones related to social plans, as well as those related to agents' intentional stance. The latter concerns different aspects of awareness that appear in a group of agents in the course of DCPS. The degree of this awareness varies, as explained before. In the sequel, the strongest version is considered, namely collective belief about considered aspects of DCPS. For this reason it is justified to speak about *collective awareness* in this context, while in other circumstances, the degree of awareness can be weakened by using general belief E-BEL$_G$ (or another E-BEL^k_G) instead of collective belief C-BEL$_G$. Let us discuss the relevant aspect of building pair-wise commitments, where the notion of awareness plays a crucial role, more in detail. For extensive discussion of other aspects of commitments see [7].

When a plan P as a recipe is in place, then the particular actions α from it need to be allocated to particular team members (i, j) in order to create pairwise social commitments. This way a social team structure is built, and the plan acquires the property of being social. The type of awareness connected with this phase may be twofold.

- A *detailed* collective awareness of each social commitment:
  $\bigwedge_{\alpha \in P} \bigvee_{i,j \in G}$ C-BEL$_G$(COMM(i, j, α))
  This corresponds to the interpretation *de re*.
- A *global* collective awareness of the bare existence of social commitments:
  C-BEL$_G(\bigwedge_{\alpha \in P} \bigvee_{i,j \in G}$ COMM$(i, j, \alpha))$
  This corresponds to the interpretation *de dicto*.

Clearly, the distinction de dicto versus de re (of section 2.2) is also fruitful for complex epistemic operators such as collective belief. Note that C-BEL$_G$ in (a) and (b) distributes over conjunction ($\bigwedge_{\alpha \in P}$), so that only the position of C-BEL$_G$ with respect to $\bigvee_{i,j \in G}$ matters. The cognitive multi-agent distinction between both interpretations is the following. In the detailed collective awareness everybody knows every single detail, while in the global collective awareness everybody knows that all details are somehow arranged, and that things are under control.

The above aspects of awareness serve to distinguish different strengths of collective commitments. Exemplar definitions are produced by keeping the *awareness*$_G$-dial fixed to a choice between $\emptyset$ and collective belief. In [7], we started from the strongest form of collective commitment, fully reflecting the collective aspects of DCPS. Subsequently, some underlying assumptions were relaxed, leading ultimately to weaker notions of team and distributed commitment. In ongoing research, the interrelations between different forms of commitments and management theory are investigated.

6. DISCUSSION AND CONCLUSIONS

In this paper we have shown that agents working as a team depend, for good coordination and cooperation, on awareness of their own and others' mental states, as well as of the group attitudes. We have shown that both communicative possibilities and cognitive and computational limitations make the achievement of awareness difficult. Some examples showed that in BDI logics, used to make high-level specifications of MAS behavior, it is possible to show rather precisely where the limitations lie. It is important for system developers to keep these limitations in mind and to adapt to them.

Let us sum up some of the possible avenues of adaptation the system developer might choose. With respect to intrapersonal awareness, to enable agents' positive and negative introspection, it helps to restrict the language to a limited number of propositional variables, for which agents can easily check whether they believe in them (respectively know them) or not. In order to create a specification framework in which the problems of logical omniscience does not occur, the system developer may adapt a framework of non-normal modal logic as presented in [10] to the application at hand.

With respect to inter-personal awareness, a system developer designing a combined agent-human multiagent system should adapt to the fact that humans can only correctly reason about at most level two theory of mind ("you do not know that I know p"). Thus, it may be useful to model their reasoning about other individuals by a framework of local reasoning adapted from the proposal in [11].

Finally, in order to adapt to the problems in creating group awareness, a system developer may use the tuning mechanisms of section 5 to adapt the kind of awareness needed in collective intentions, social commitments, and collective commitments to one that is appropriate for the environment and kind of organization. This seems to be a rather sensitive mechanism allowing to calibrate the expressive power of the notion in question. Moreover, this is a very flexible method of defining various common-sense notions, as any case-specific extensions can be easily incorporated into the collective intention or collective commitment schema. Again, the resulting definition and its properties can be viewed as a high level specification of a system.

Emma Norling [19] also combines human modeling with BDI systems. She focuses on extending the BDI framework with characteristics of human reasoning such as timing, learning and memory, all understood in a *folk psychological* sense, in order to allow developers of training simulators to model human-like synthetic agents. We, on the other hand, remain within the scope of the BDI framework in that we focus on agents' informational and motivational attitudes, but pose the question where BDI models should be adapted to better model human and computational information processing constraints and cognitive limitations. Thus, we build on *cognitive science* studies of human awareness about themselves, others and the groups they are part of.

Software agents' bounded rationality is also intimately connected to the computational complexity of agent logics. The system defining collective intentions and collective commitments in [5, 7] is decidable, however it is highly complex. In order to decide whether a given formula is valid, one needs an amount of time that grows exponentially in terms of the length of the formula in question [9]. Thus, in general it is not feasible to give automated proofs of desired properties; at least there is no single algorithm that performs well on all inputs. As with other modal logics, the better option would be to develop a variety of different algorithms and heuristics, each performing well on a limited class of inputs. For example, it is known that restricting the number of propositional atoms to be used or the depth of modal nesting may reduce the complexity (cf. [12, 14]).

The outcome of our considerations on possible solutions can still be combined with some other, for example strictly technical, elements. Also, when creating teams in which humans and agents cooperate, system developers may divide roles in such a way that they make use of strong points of humans, such as common sense knowledge, as well as those of software agents, such as their memory, which enhances monitoring the team's activity.

7. REFERENCES

[1] J. Brzezinski, P. Dunin-Kęplicz, and B. Dunin-Kęplicz. Collectively cognitive agents in cooperative teams. In M. P. Gleizes, A. Omicini, and F. Zambonelli, editors, *Engineering Societies in the Agents World V, (ESAW 2004): Revised Selected and Invited Papers*, volume 3451 of *LNCS*, pages 191–208, Berlin, 2005. Springer.

[2] C. Castelfranchi. Commitments: From individual intentions to groups and organizations. In V. Lesser, editor, *Proceedings First International Conference on Multi-Agent Systems*, pages 41–48, San Francisco, 1995. AAAI-Press and MIT Press.

[3] H. H. Clark and C. Marshall. Definite reference and mutual knowledge. In A. Joshi, B. Webber, and I. Sag, editors, *Elements of Discourse Understanding*, pages 10–63. Cambridge University Press, 1981.

[4] F. Dignum, B. Dunin-Kęplicz, and R. Verbrugge. Creating collective intention through dialogue. *Logic Journal of the IGPL*, 9:145–158, 2001.

[5] B. Dunin-Kęplicz and R. Verbrugge. Collective intentions. *Fundamenta Informaticae*, 51(3):271–295, 2002.

[6] B. Dunin-Kęplicz and R. Verbrugge. Evolution of collective commitments during teamwork. *Fundamenta Informaticae*, 56:329–371, 2003.

[7] B. Dunin-Kęplicz and R. Verbrugge. A tuning machine for cooperative problem solving. *Fundamenta Informaticae*, 63:283–307, 2004.

[8] B. Dunin-Kęplicz and R. Verbrugge. Creating common beliefs in rescue situations. In B. Dunin-Keplicz, A. Jankowski, A. Skowron, and M. Szczuka, editors, *Proceedings of Monitoring, Security and Rescue Techniques in Multiagent Systems (MSRAS)*, Advances in Soft Computing, pages 69–84, Berlin, 2005. Springer.

[9] M. Dziubinski, R. Verbrugge, and B. Dunin-Kęplicz. Complexity of a theory of collective attitudes in teamwork. In A. S. et al., editor, *International Conference on Intelligent Agent Technology (IAT 2005)*, pages 579–586, Los Alamitos (CA), 2005. IEEE Computer Society.

[10] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, MA, 1995.

[11] P. Gochet and E. Gillet. On Professor Weingartner's contribution to epistemic logic. In *Advances in Scientific Philosophy*, pages 97–115. Rodopi, Amsterdam, 1991.

[12] J. Halpern. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75:361–372, 1995.

[13] T. Hedden and J. Zhang. What do you think I think you think? Strategic reasoning in matrix games. *Cognition*, 85:1–36, 2002.

[14] U. Hustadt and R. Schmidt. On evaluating decision procedures for modal logics. In M. Pollack, editor, *Proceedings IJCAI'97*, Los Angeles (CA), 1997. Morgan Kauffman.

[15] B. Keysar, S. Lin, and D. Barr. Limits on theory of mind use in adults. *Cognition*, 89:25–41, 2003.

[16] L. Litman and A. Reber. Implicit cognition and thought. In K. Holyoak and R. Morrison, editors, *The Cambridge Handbook of Thinking and Reasoning*, pages 431–453. Cambridge University Press, Cambridge, 2005.

[17] J.-J. C. Meyer and W. van der Hoek. *Epistemic Logic for AI and Theoretical Computer Science*. Cambridge University Press, Cambridge, 1995.

[18] L. Mol, N. Taatgen, L. Verbrugge, and P. Hendriks. Reflective cognition as secondary task. In B. Bara, L. Barsalou, and M. Bucciarelli., editors, *Proceedings of Twenty-seventh Annual Meeting of the Cognitive Science Society*, pages 1925–1930, Mahwah (NJ), 2005. Erlbaum.

[19] E. Norling. Folk psychology for human modelling: Extending the BDI paradigm. In *International Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 202–209, New York, 2004. ACM Press.

[20] R. Parikh and P. Krasucki. Levels of knowledge in distributed computing. *Sadhana: Proceedings of the Indian Academy of Sciences*, 17:167–191, 1992.

[21] W. Quine. Quantifiers and propositional attitudes. *Journal of Philosophy*, 53:177–187, 1956.

[22] K. Sycara and M. Lewis. Integrating intelligent agents into human teams. In E. Salas and S. Fiore, editors, *Team Cognition: Understanding the Factors that Drive Process and Performance*, pages 203–232, Washington (DC), 2004. American Psychological Association.

[23] W. van der Hoek and R. Verbrugge. Epistemic logic: A survey. In L. Petrosjan and V. Mazalov, editors, *Game Theory and Applications*, pages 53–94. Nova Science Publishers, vol. 8, New York, 2002.

[24] O. Williamson. The economies of organization: The transaction cost approach. *American Journal of Sociology*, 87:548–577, 1981.

[25] T. Williamson. *Knowledge and its Limits*. Oxford University Press, Oxford, 2000.

Exact Solutions of Interactive POMDPs Using Behavioral Equivalence

Bharanee Rathnasabapathy
Dept. of Computer Science
University of Illinois at Chicago
brathnas@cs.uic.edu

Prashant Doshi
Dept. of Computer Science
University of Georgia
pdoshi@cs.uga.edu

Piotr Gmytrasiewicz
Dept. of Computer Science
University of Illinois at Chicago
piotr@cs.uic.edu

ABSTRACT

We present a method for transforming the infinite interactive state space of interactive POMDPs (I-POMDPs) into a finite one, thereby enabling the computation of exact solutions. I-POMDPs allow sequential decision making in multi-agent environments by modeling other agents' beliefs, capabilities, and preferences as part of the interactive state space. Since beliefs are allowed to be arbitrarily nested and are continuous, it is not possible to compute optimal solutions using value iteration as in POMDPs. We present a method that transforms the original state space into a finite one by grouping the other agents' behaviorally equivalent models into equivalence classes. This enables us to compute the *complete* optimal solution for the I-POMDP, which may be represented as a policy graph. We illustrate our method using the multi-agent Tiger problem and discuss features of the solution.

1. INTRODUCTION

Interactive partially observable Markov decision processes (I-POMDPs) [8] provide a framework for sequential decision making in partially observable multi-agent environments. Along with the physical states of the environment they include other agents' computable models in the state space. The models encompass all information influencing the corresponding agent's behavior, which includes its preferences, capabilities, and beliefs. They are analogous to *types* in Bayesian games[10]. This augmented state space is called an *interactive state space*. Because beliefs in I-POMDPs are arbitrarily nested, the interactive state space is not the traditional one – it is a complex infinite space that contains beliefs over others' beliefs.

Traditional methods for solving POMDPs [4, 11, 16] focus on finite state spaces and exploit the piecewise linearity and convexity (PWLC) property of the value function. For POMDPs with infinite state spaces exact closed form solutions exist only in linear Gaussian systems [13, 17]. I-POMDPs, which generalize POMDPs to multi-agent domains, have a infinite state space as well, making it impossible to compute the exact solution by applying value iteration based methods in a straightforward manner.

In this paper, we present a method for computing the exact solutions to finitely nested I-POMDPs introduced in [8]. Our method transforms the infinite interactive state space into a finite space by grouping models of other agents into a finite set of behavioral equivalence classes. They are defined so that the optimal action(s) of the modeled agents are the same for all models inside a class. Given such behavioral equivalence classes, and the agents' optimal actions in these classes, we show that it is sufficient to reason about the other agents' equivalence classes, rather than the infinity of individual models, to compute a solution. When the set of actions and observations of each agent is finite, the set of behavioral equivalence classes for a finite horizon is also finite, and at most countably infinite for an infinite horizon. In practice though the number of behavioral equivalence classes will be finite for the infinite horizon because we usually settle for ϵ-optimal policies. Once we have transformed the interactive state space using the equivalence classes, we modify the I-POMDP belief update and value iteration to operate over the transformed interactive state space. Consequently, we can extend the exact POMDP solution methods in a straightforward manner to solve I-POMDPs exactly.

Related work in the area of multi-agent and multi-body planning like DEC-POMDPs [1, 2] and MTDP [12] only allow us to compute equilibrium based solutions under the assumption of common knowledge of the agents' prior beliefs. Further, dynamic programming approaches to find optimal solutions to DEC-POMDPs [9] are only possible in special co-operative cases where all agents share the same payoff function. In the context of state equivalence and aggregation, Given et al. [7] provide a nice discussion of using stochastic bisimilarity for model minimization in Markov decision processes and Poupart and Boutilier [14] propose an approach to get linear lossy compression of value function in POMDPs. Our method, in contrast, exploits the piecewise linearity and convexity of the value function of agent models to find the behavioral equivalence classes. Additionally, unlike [7, 14] our approach yields an optimal solution. Methods to solve I-POMDPs and DEC-POMDPs approximately have also appeared. In [5, 6], a sampling based approximation technique is presented, while in [3], bounded finite state controllers are used to derive the joint policy.

The rest of the paper is structured in the following manner. In Section 2 we give a brief overview of the I-POMDP framework. Section 3 introduces the idea of behavioral equivalence and its use in solving I-POMDPs exactly. In Section 4, we illustrate our method, using an example problem and discuss several solutions. We conclude in Section 5 and outline some avenues of future work.

2. OVERVIEW OF I-POMDP

Interactive POMDPs extend POMDPs to multi-agent settings by including other agents' models as part of the state space [8]. Since other agents might also be reasoning about others, the interactive

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

state space can be arbitrarily nested (belief about other agents' beliefs about other agents' beliefs and so on). For simplicity of presentation we will consider an agent, i, that is interacting with only one other agent, j.

DEFINITION 1 (I-POMDP$_{i,l}$). *A finitely nested interactive POMDP of agent i with a belief nesting level l referred to as I-POMDP$_{i,l}$ is defined as the tuple*

$$\textit{I-POMDP}_{i,l} = \langle IS_{i,l}, A, T_i, \Omega_i, O_i, R_i \rangle$$

where,

• $IS_{i,l}$ denotes a set of interactive states defined as, $IS_{i,l} = S \times \{\Theta_{j,l-1} \cup SM_j\}$, for $l \geq 1$, and $IS_{i,0} = S$, where S is the set of states of the physical environment, $\Theta_{j,l-1}$ is the set of computable *intentional models* of agent j at belief nesting level $l-1$: $\theta_{j,l-1} = \langle b_{j,l-1}, \hat{\theta}_{j,l-1} \rangle$ where the *frame*, $\hat{\theta}_{j,l-1} = \langle A, \Omega_j, T_j, O_j, R_j, OC_j \rangle$. Here, j is Bayes rational and OC_j is j's optimality criterion that influences its decision making process. SM_j is the set of sub-intentional models of j. We give a recursive bottom-up construction of the interactive state space below,

$$\begin{aligned}
IS_{i,0} &= S, \\
&\quad \Theta_{j,0} = \{\langle b_{j,0}, \hat{\theta}_{j,0} \rangle \mid b_{j,0} \in \Delta(IS_{j,0})\} \\
IS_{i,1} &= S \times \{\Theta_{j,0} \cup SM_j\}, \\
&\quad \Theta_{j,1} = \{\langle b_{j,1}, \hat{\theta}_{j,1} \rangle \mid b_{j,1} \in \Delta(IS_{j,1})\} \\
&\vdots \quad \vdots \\
IS_{i,l} &= S \times \{\Theta_{j,l-1} \cup SM_j\}, \\
&\quad \Theta_{j,l} = \{\langle b_{j,l}, \hat{\theta}_{j,l} \rangle \mid b_{j,l} \in \Delta(IS_{j,l})\}
\end{aligned}$$

• A is the set of joint actions of all agents in the environment, $A = A_i \times A_j$

• T_i describes the effect of the joint actions on the physical states of the environment, $T_i : S \times A \times S \rightarrow [0,1]$

• Ω_i is the set of observations of agent i

• $O_i : S \times A \times \Omega_i \rightarrow [0,1]$ gives the likelihood of the observations given the physical state and joint action

• $R_i : IS_i \times A \rightarrow \mathbb{R}$ describes agent i's preferences over it's interactive states. Usually only the physical states will matter

Agent i's policy is the mapping $\Omega_i^* \rightarrow \Delta(A_i)$ where Ω_i^* is the set of all observation histories of agent i. Since belief over the interactive states forms a sufficient statistic [8], the policy can also be represented as a mapping from the set of all belief's of agent i to a distribution over it's actions, $\Delta(IS_i) \rightarrow \Delta(A_i)$.

2.1 Belief Update

Analogous to POMDPs, an agent within the I-POMDP framework also updates its belief as it acts and observes. However, there are two differences that complicate a belief update in multi-agent settings when compared to single-agent ones. First, since the state of the physical environment depends on the actions performed by both agents, – in our example agents i and j – i's prediction of how the physical state changes has to be made based on it's prediction of j's actions. Second, changes in j's model have to be included in i's belief update. Specifically, if j is intentional then an update of j's beliefs due to its action and a new observation has to be included. In other words, i has to update its belief based on it's prediction of what j would observe and how j would update its belief. If j's model is sub-intentional, then j's probable observations are appended to the observation history contained in the model. Formally, following [8], we have:

$$\begin{aligned}
&Pr(is^t | a_i^{t-1}, b_{i,l}^{t-1}) = \beta \int_{IS^{t-1}: \widehat{m}_j^{t-1} = \widehat{\theta}_j^t} b_{i,l}^{t-1}(is^{t-1}) \\
&\times \textstyle\sum_{a_j^{t-1}} Pr(a_j^{t-1} | \theta_{j,l-1}^{t-1}) O_i(s^t, a_i^{t-1}, a_j^{t-1}, o_i^t) \\
&\times T_i(s^{t-1}, a_i^{t-1}, a_j^{t-1}, s^t) \textstyle\sum_{o_j^t} O_j(s^t, a_i^{t-1}, a_j^{t-1}, o_j^t) \\
&\times \delta_D(SE_{\widehat{\theta}_j^t}(b_{j,l-1}^{t-1}, a_j^{t-1}, o_j^t) - b_{j,l-1}^t) d(is^{t-1})
\end{aligned} \tag{1}$$

where β is the normalizing constant, δ_D is the Dirac-delta function, $SE(\cdot)$ is an abbreviation for the belief update, and $Pr(a_j^{t-1} | \theta_{j,l-1}^{t-1})$ is the probability that a_j^{t-1} is Bayes rational for the agent described by model $\theta_{j,l-1}^{t-1}$. When j's model is sub-intentional, the integration is over $IS^{t-1} : \widehat{m}_j^{t-1} = \widehat{m}_j^t$, $Pr(a_j^{t-1} | \theta_{j,l-1}^{t-1})$ is replaced with $Pr(a_j^{t-1} | m_{j,l-1}^{t-1})$, and $\delta_D(SE_{\widehat{\theta}_j^t}(b_{j,l-1}^{t-1}, a_j^{t-1}, o_j^t) - b_{j,l-1}^t)$ is replaced with $\delta_K(APPEND(h_j^{t-1}, o_j^t) - h_j^t)$. δ_K is the Kronecker delta, and $APPEND$ returns a string with the second argument appended to the first. If j is also modeled as an I-POMDP, then i's belief update invokes j's belief update (via the term $SE_{\widehat{\theta}_j^t}(b_{j,l-1}^{t-1}, a_j^{t-1}, o_j^t)$), which in turn could invoke i's belief update and so on. This recursion in belief nesting bottoms out at the 0th level. At this level, belief update of the agent reduces to a POMDP belief update. [1] For an illustration of the belief update, additional details on I-POMDPs, and how they compare with other multi-agent planning frameworks, see [8].

2.2 Value Iteration

Each belief state in a finitely nested I-POMDP has an associated value reflecting the maximum payoff the agent can expect in this belief state:

$$\begin{aligned}
&U^n(\langle b_{i,l}, \widehat{\theta}_i \rangle) = \max_{a_i \in A_i} \Big\{ \int_{is \in IS_{i,l}} ER_i(is, a_i) b_{i,l}(is) d(is) + \\
&\gamma \sum_{o_i \in \Omega_i} Pr(o_i | a_i, b_{i,l}) U^{n-1}(\langle SE_{\widehat{\theta}_i}(b_{i,l}, a_i, o_i), \widehat{\theta}_i \rangle) \Big\}
\end{aligned} \tag{2}$$

where, $ER_i(is, a_i) = \sum_{a_j} R_i(is, a_i, a_j) Pr(a_j | m_{j,l-1})$ (since $is = (s, m_{j,l-1})$). Eq. 2 is a basis for value iteration in I-POMDPs.

Agent i's optimal action, a_i^*, for the case of finite horizon with discounting, is an element of the set of optimal actions for the belief state, $OPT(\theta_i)$, defined as:

$$\begin{aligned}
&OPT(\langle b_{i,l}, \widehat{\theta}_i \rangle) = \operatorname*{argmax}_{a_i \in A_i} \Big\{ \int_{is \in IS_{i,l}} ER_i(is, a_i) b_{i,l}(is) d(is) \\
&+ \gamma \sum_{o_i \in \Omega_i} Pr(o_i | a_i, b_{i,l}) U^n(\langle SE_{\widehat{\theta}_i}(b_{i,l}, a_i, o_i), \widehat{\theta}_i \rangle) \Big\}
\end{aligned} \tag{3}$$

3. BEHAVIORAL EQUIVALENCE

The main idea in this paper is to aggregate interactive states into a finite number of equivalence classes using behavioral equivalence. Instead of reasoning over the infinite set of interactive states, we operate over the finite set of equivalence classes of interactive states (ECIS). In the next subsection, we introduce the concept of behavioral equivalence and show that reasoning over the equivalence classes preserves the optimal policy.

3.1 Defining the Equivalence Classes

In order to illustrate the construction of the behavioral equivalence classes we look at a simple example – the classical *tiger*

[1] The 0th level model is a POMDP: Other agent's actions are treated as exogenous events and folded into the T, O, and R functions.

problem introduced in [11]. The problem is that of an agent having to open either of two doors. Behind one of the doors is a tiger waiting to eat the agent and behind the other is a pot of gold. There is a reward of +10 to get the gold and -100 when the agent is eaten by the tiger. There are two states signifying the tiger's location behind the left (*TL*) door and the right (*TR*) door. The agent has three actions: open left door (*OL*), open right door (*OR*) and listen (*L*). Listening always incurs a cost of 1. When the agent listens, it can hear a growl on the left (*GL*) or right (*GR*) with 85% certainty. The

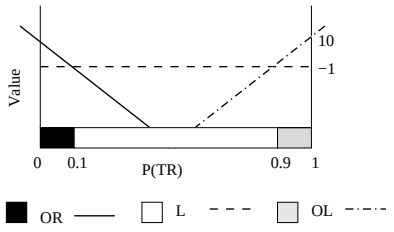


Figure 1: Horizon-1 value function in the tiger game and the belief ranges corresponding to different optimal actions.

value function gives the value of performing the optimal plan at any given belief. In Fig. 1, the value function is the envelope that provides the maximum value of any plan over all beliefs. We note that the agent opens the right door if it believes the probability that the tiger is behind the right door, *P(TR)*, is less than 0.1. It will listen if 0.1 < *P(TR)* < 0.9 and open left door if *P(TR)* > 0.9.

In Fig. 1 we observe that each of the optimal plans spans over multiple belief points. For example, opening the right door is the optimal action for all beliefs in the set [0–0.1), i.e. OPT(P(TR)= [0–0.1)) = OR. Thus, beliefs in the set [0–0.1) are equivalent to each other in that they induce the same optimal behavior. We call such beliefs *behaviorally equivalent*. [2] The collection of the equivalence classes (five of them in the example in Fig. 1) forms a partition of the belief space. For finite horizons, and a finite number of actions and observations, the number of distinct plans and therefore the equivalence classes is also finite. Note that each equivalence class is a node in the policy graph in a POMDP.

We use the above insight to discretize the infinite interactive state space of I-POMDPs into a finite one. The optimal value function in I-POMDPs, like POMDPs, is also piecewise linear and convex (PWLC) [8]. Recall that the interactive state space, $IS_{i,l} = S \times \{\Theta_{j,l-1} \cup SM_j\}$, where $\Theta_{j,l-1}$ is the set of intentional models of agent j that includes its beliefs, and all other parameters sufficient to determine its behavior. SM_j is the set of j's sub-intentional models. We transform the infinite space into a finite one by grouping together the behaviorally equivalent models into the equivalence classes. As we mentioned before, for a finite horizon and finite number of actions and observations, there exist a finite number of equivalence classes, thereby facilitating the application of standard POMDP solution methods for solving I-POMDPs.

We point out that the method scales well to higher levels of nesting. Since at each level of nesting we operate over the equivalence classes only, we need not concern ourselves with the complex space of beliefs over beliefs of other agents and so on. Of course, in deriving the equivalence classes we must solve all lower level models in a recursive manner. We formally define the equivalence classes below:

[2] It is possible that some plans may be optimal for only a single belief. In this case, the equivalence class would be a singleton set containing the single belief.

DEFINITION 2 (ECIS). *Equivalence classes of interactive states ($ECIS_{i,l}$) of an agent i is a* partition *of $IS_{i,l}$ such that the behavior of agent j is the same for all interactive states in a given subset.*

$$ECIS_{i,l} = \{ (s, M_{j,k}^{l-1}) \mid M_{j,k}^{l-1} \subseteq \{\Theta_{j,l-1} \cup SM_j\} and \forall_{m_{j,l-1}, m'_{j,l-1} \in M_{j,k}^{l-1}} OPT(m_{j,l-1}) = OPT(m'_{j,l-1}) \}$$

Here, $M_{j,k}^{l-1}$, is the kth equivalence class in the partition. Note that OPT(.) takes a model of the other agent as a parameter and gives the optimal action of the agent (see Eq. 3).

3.2 Belief Update and Value Iteration over ECIS

The main difference between the belief update as defined in Eq. 1 over the original interactive state space and the belief update over the transformed one is in the manner in which agent i updates j's models. Since j's model space is partitioned into equivalence classes, i must update j's equivalence class based on j's action and observation. To understand how this is done, let us consider the simple case where i is uncertain only about j's beliefs. In this case, each equivalence class is a disjoint subset of j's beliefs that induces identical optimal behavior.

As we mentioned before, the optimal solution to an I-POMDP (and a POMDP) provides a conditional plan or a policy graph that dictates the action(s) that agent j must perform given the equivalence class of its beliefs. Agent i updates j's equivalence class by locating the node for the current equivalence class in j's policy graph, and using j's predicted action and possible observation to trace to the next one. Before presenting the belief update equations, we note that i's belief over the transformed interactive state space forms a sufficient statistic.

PROPOSITION 1. *(Sufficiency) In a finitely nested I-POMDP, a probability distribution over $ECIS_{i,l}$, $\tilde{b}_{i,l} \in \Delta(ECIS_{i,l})$, provides a sufficient statistic for the past history of i's observations.*

We outline the proof of this proposition in the Appendix. We give the belief update over ECIS for an arbitrary belief nesting level l below:

$$\begin{aligned} & Pr(ecis^t | o_i^t, a_i^{t-1}, \tilde{b}_{i,l}^{t-1}) = \beta \sum_{ECIS^{t-1}} \tilde{b}_{i,l}^{t-1}(ecis^{t-1}) \\ & \times \sum_{a_j^{t-1}} Pr(a_j^{t-1} \mid M_{j,k}^{t-1})\, O_i(s^t, a^{t-1}, o_i^t) \\ & \times\ T_i(s^{t-1}, a^{t-1}, s^t) \sum_{o_j^t} \tilde{\tau}(M_{j,k}^{t-1}, a_j^{t-1}, o_j^t, M_{j,k}^t) \\ & \times\ O_j(s^t, a^{t-1}, o_j^t) \end{aligned} \quad (4)$$

where β is the normalizing constant. Agent i's update of j's equivalence classes is captured using the $\tilde{\tau}$ operator which returns 1 if $Pr(M_{j,k}^t | M_{j,k}^{t-1}, a_j, o_j) = 1$, and 0 otherwise. The $\tilde{\tau}$ operator uses j's policy graph to determine the updated equivalence class. Note that $M_{j,k}^{t-1}$ contains j's equivalent models before j performs it's belief update and $M_{j,k}^t$ contains models that are equivalent after j's belief update.

Since i's beliefs are now distributions over the transformed interactive state space, the value iteration, as shown in Eq. 2 is revised to include the value function defined over the transformed belief space:

$$\tilde{U}^n(\langle \tilde{b}_{i,l}, \widehat{\theta}_i \rangle) = \max_{a_i \in A_i} \Big\{ \sum_{ecis \in ECIS_{i,l}} ER_i(ecis, a_i) \tilde{b}_{i,l}(ecis) \times \gamma \sum_{o_i \in \Omega_i} Pr(o_i | a_i, \tilde{b}_{i,l}) \tilde{U}^{n-1}(\langle SE_{\widehat{\theta}_i}(\tilde{b}_{i,l}, a_i, o_i), \widehat{\theta}_i \rangle) \Big\} \quad (5)$$

In the next proposition, we show that the discretization of the interactive state space is a *lossless* one. In other words, for a given belief over the original interactive state space, the value of the optimal plan (and the optimal plan itself) remains unchanged for the corresponding belief over the discretized interactive state space. We define the concept of a value preserving transformation and formalize our result in Proposition 2.

DEFINITION 3 (VALUE PRESERVING TRANSFORMATION). *Define a transformation $F : B_1 \rightarrow B_2$, where B_1 is the space of original beliefs, and B_2 is the space of corresponding transformed beliefs. Let $Q : N \times B_1 \times A \rightarrow \mathbb{R}$ and $\tilde{Q} : N \times B_2 \times A \rightarrow \mathbb{R}$ be the N horizon action-value functions. F is said to be value preserving iff $\forall_{n \in [1,N], b \in B_1, a \in A} Q^n(b,a) = \tilde{Q}^n(F(b),a)$. In other words, the value of the transformed belief and action pair is same as the value of the original belief and action, for all horizons.*

PROPOSITION 2. *Define a mapping $CP : \Delta(IS_{i,l}) \rightarrow \Delta(ECIS_{i,l})$ in a finitely nested I-POMDP such that,*

$$\tilde{b}_i(s, M_{j,k}^{l-1}) = \int_{m_j \in M_{j,k}^{l-1}} b_i(s, m_j) dm_j \qquad (6)$$

where $b_i \in \Delta(IS_{i,l})$, $\tilde{b}_i \in \Delta(ECIS_{i,l})$, and $M_{j,k}^{l-1}$ is some equivalence class. The mapping CP is value preserving.

The proof of proposition 2 is given in the Appendix.

COROLLARY 1. *The optimal policy of the transformed finitely nested I-POMDP remains unchanged.*

PROOF. *Proof of this corollary is inductive, and involves a straightforward application of Proposition 2. From Proposition 2, CP is a value-preserving transformation. For the basis step – horizon is 1 – $\forall_{b_i,a_i} Q^1(b_i,a_i) = \tilde{Q}^1(CP(b_i),a_i)$. Therefore, the optimal single step action remains unchanged. The inductive proof then follows for any arbitrary horizon n. Since $\forall_{b_i,a_i} Q^n(b_i,a_i) = \tilde{Q}^n(CP(b_i),a_i)$, therefore $\forall_{b_i \in \Delta(IS_i)} OPT(b_i) = \tilde{OPT}(CP(b_i))$, the optimal policy remains the same.* □

4. MULTI-AGENT TIGER PROBLEM

To illustrate our method we use a slightly modified version of the multi-agent tiger problem discussed in [8] (based on [15].) The problem has two agents, each of which can open the right door (OR), the left door (OL) or listen (L). In addition to observing growls when they listen, the agents can also hear creaks (creak on the left (CL), creak on the right (CR) or silence (S), i.e no creaks) indicating the other agent's opening one of the doors. When any door is opened, the tiger's location is switched with a probability of 5%. The agent's preferences are as in the single agent game discussed in Section 3. The transition, reward and observation functions are shown in Table 1. From the observation functions we see that, in this case, agent i hears growls with a reliability of 65% and creaks with a reliability of 95%. Agent j, on the other hand, hears growls with a reliability of 95%. Thus, the setting is such that agent i can hear agent j opening doors more reliably than the tiger's growls. This suggests that i could use j's actions as an indication of the location of the tiger, as we discuss below.

For the sake of simplicity we assume agent i's I-POMDP to be singly nested and that agent i ascribes only intentional models to j. Additionally, we assume that i is uncertain only about j's beliefs and not j's frame. Note that in a general I-POMDP, i can be uncertain about all parameters of j's model. In Fig. 2(a) we give the optimal solution of the level 0 I-POMDP of agent j. Since

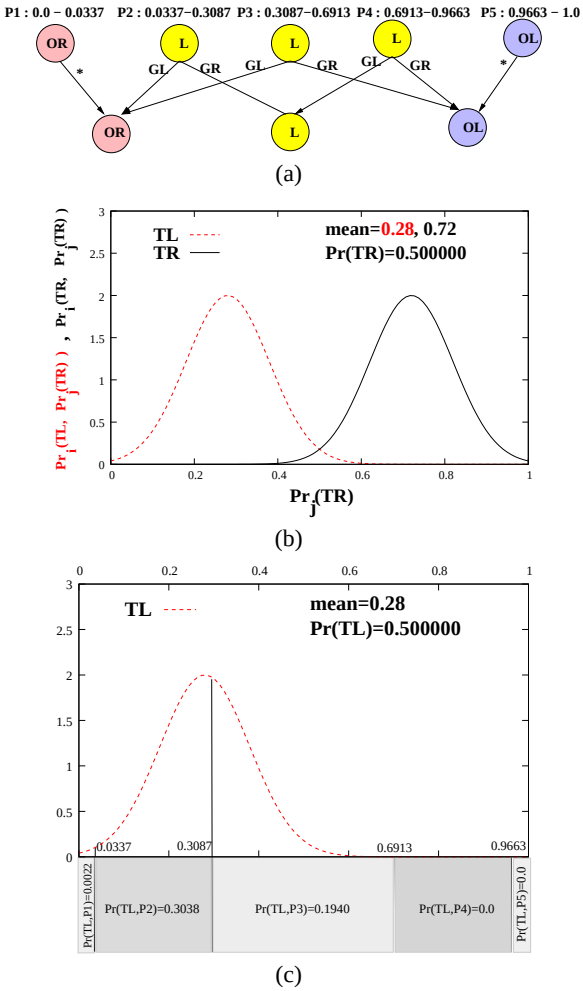


Figure 2: (a) Agent j's horizon 2 policy showing optimal plans given different prior beliefs. Since j's belief is not nested, its prior belief is given as the probability that the tiger is behind the right door. (b) An example of agent i's level-1 belief. Note that we have 2 distributions, one when the tiger is on the left and other when tiger is on the right. In the belief shown, i thinks that j is likely to be somewhat informed about the tiger's true location. (c) Shows how agent i's belief $b_{i,1} \in \Delta(IS_{i,1})$ (at 2 steps to go) is transformed into a belief $\tilde{b}_{i,1} \in \Delta(ECIS_{i,1})$. For example, the probability that the tiger is behind the left door and j's belief is in region $P2$ is given as $\tilde{b}_{i,1}(TL, P2) = \int_{0.0337}^{0.3087} b_{i,1}(TL, Pr_j(TR)) dPr_j(TR) = 0.3038$, where $b_{i,1}(TL, Pr_j(TR))$ is the Gaussian pdf describing i's belief about j's belief when the tiger is on the left. $\int_0^1 b_{i,1}(TL, Pr_j(TR)) dPr_j(TR)$ gives i's belief that the tiger is behind the left door. Similarly, $\tilde{b}_{i,1}(TR, P1), \tilde{b}_{i,1}(TR, P2) \ldots \tilde{b}_{i,1}(TR, P5)$ are computed from the Gaussian describing i's belief about j's belief when the tiger is behind the right door.

agent j's belief is not nested, j reasons about the tiger's location only and not about agent i's beliefs. Therefore, agent j's actions depend on hearing growls alone. The transformed horizon-2 interactive state space is: $ECIS_{i,1} = S \times \{P1, P2, P3, P4, P5\}$, where the classes $P1 - P5$ are shown in Fig. 2(a). Since $ECIS_{i,1}$ is finite, we can obtain the *complete* optimal value function using the modified value iteration shown in Eq. 5.

In the remainder of this section we visualize i's value function

(a) Transition function for i and j

Action <i,j>	State	TL	TR
<OL,*> , <*,OL>	TL	0.95	0.05
<OL,*> , <*,OL>	TR	0.05	0.95
<OR,*> , <*,OR>	TL	0.95	0.05
<OR,*> , <*,OR>	TR	0.05	0.95
<L,*> , <*,L>	TL	1.0	0.0
<L,*> , <*,L>	TR	0.0	1.0

(b) Reward function for i and j

Action <i,j>	TL	TR
<OL,OL>	-100,-100	10,10
<OL,OR>	-100,10	10,-100
<OL,L>	-100,-1	10,-1
<OR,OL>	10,-100	-100,10
<OR,OR>	10,10	-100,-100
<OR,L>	10,-1	-100,-1
<L,OL>	-1,-100	-1,10
<L,OR>	-1,10	-1,-100
<L,L>	-1,-1	-1,-1

(c) Observation function for i with a growl reliability of 65% and creak reliability of 95%

Action <i,j>	State	<GL,CL>	<GL,CR>	<GL,S>	<GR,CL>	<GR,CR>	<GR,S>
<OL,*>	*	1/6	1/6	1/6	1/6	1/6	1/6
<OR,*>	*	1/6	1/6	1/6	1/6	1/6	1/6
<L,OL>	TL	0.65*0.95	0.65*0.025	0.65*0.025	0.35*0.95	0.35*0.025	0.35*0.025
<L,OL>	TR	0.35*0.95	0.35*0.025	0.35*0.025	0.65*0.95	0.65*0.025	0.65*0.025
<L,OR>	TL	0.65*0.025	0.65*0.95	0.65*0.025	0.35*0.025	0.35*0.95	0.35*0.025
<L,OR>	TR	0.35*0.025	0.35*0.95	0.35*0.025	0.65*0.025	0.65*0.95	0.65*0.025
<L,L>	TL	0.65*0.025	0.65*0.025	0.65*0.95	0.35*0.025	0.35*0.025	0.35*0.95
<L,L>	TR	0.35*0.025	0.35*0.025	0.35*0.95	0.65*0.025	0.65*0.025	0.65*0.95

(d) Observation function for j

Action <i,j>	State	GL	GR
<*, OL>	*	0.5	0.5
<*, OR>	*	0.5	0.5
<OL, *>	*	0.5	0.5
<OR, *>	*	0.5	0.5
<L, L>	TL	0.95	0.05
<L, L>	TR	0.05	0.95

Table 1: The transition, reward and observation functions of agents i and j

and discuss i's behavior in the presence of j. The value function is defined over a ten dimensional space since agent j has five behavioral equivalence classes and there are two physical states. While we can compute the complete value function and policy, it is not possible to visualize it. Hence, here we plot agent i's value function for a subset of beliefs it has about agent j.

An example of agent i's belief about the location of the tiger and about j's belief is shown in Fig. 2(b). We represent agent i's belief about its interactive state as a pair of Gaussian densities with a variance of 0.01. One Gaussian captures i's belief about j's belief when the tiger is on the left, and the other captures i's belief about j's belief when the tiger is on the right (see [8] for a more detailed exposition on representing nested beliefs in I-POMDPs). To enable visualization, we capture i's belief about j's using a single parameter. To do this, we only look at those beliefs of i in which the mean of the two probability densities sum to 1, as do the densities shown in Fig. 2(b). The value function is plotted against agent i's belief about the tiger's location ($Pr_i(TR)$) and the mean of the probability density function (describing $Pr_i(Pr_j(TR))$ given the tiger is on the left. Note that in considering the particular form of i's beliefs, we have reduced the dimensions over which the value function spans in a non-linear fashion, and hence the value function plot is not linear.

Figures 3(a), 3(b), 3(c) and 3(d) show the projections of i's horizon 2 value functions on the belief space of agent i. We varied the reliability of the creaks, in order to explore its effect on i's policy. Figure 3(e) describes the optimal policies for the corresponding belief regions.

When creaks are not very reliable i's behavior is analogous to that of a single agent POMDP as seen in Fig. 3(a), as should be expected. Here, the agent uses only growls to reason about the location of the tiger.

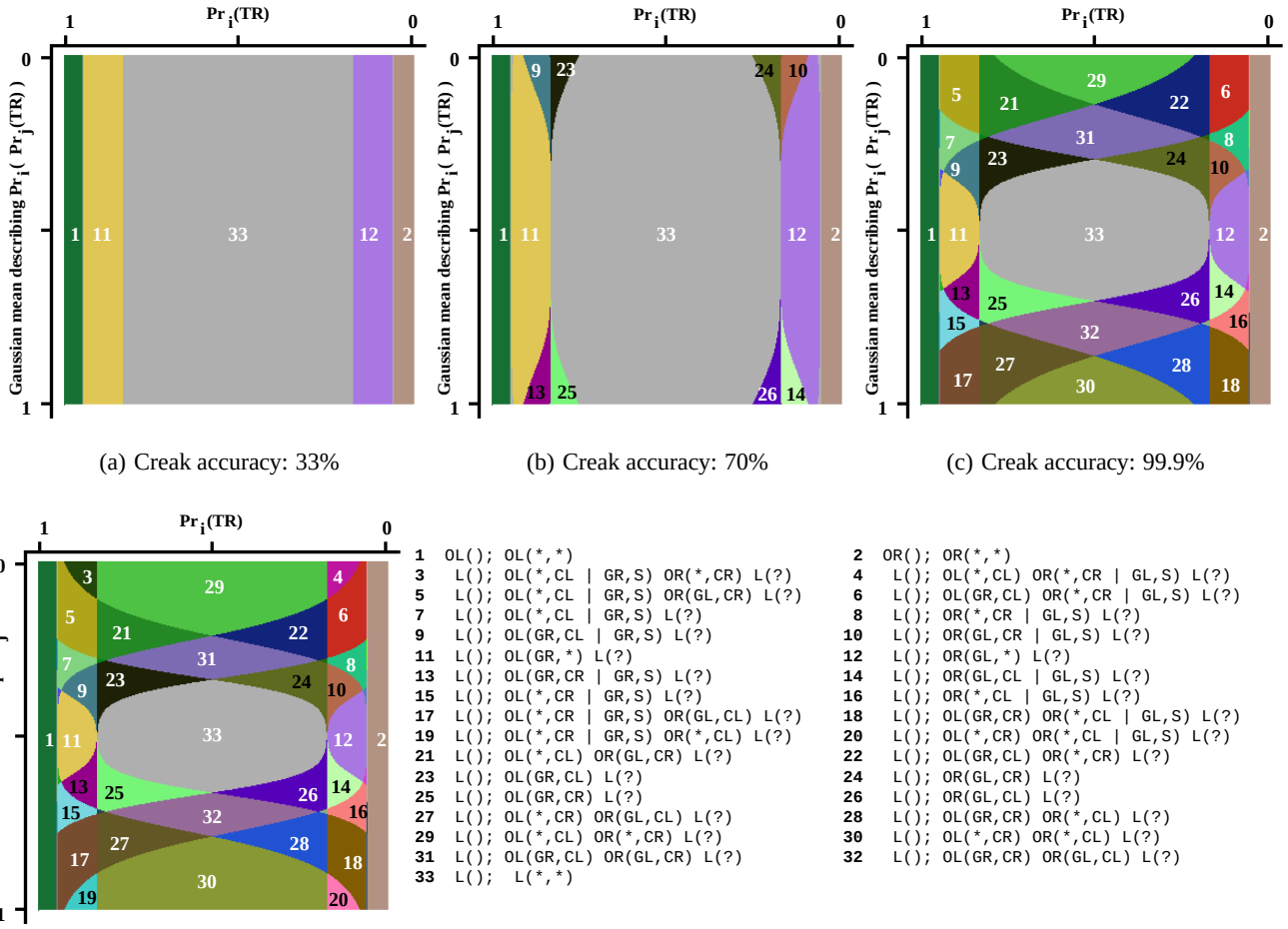
When creaks become more reliable, it is possible for i to use them to reinforce the estimates made using the growls. For example, in the belief region 9 [*Policy: L(); OL(GR,CL|GR,S) L(?)*] shown in Fig. 3(b), agent i *supplements* the information from growl (GR) with either a creak from the left (CL) or silence (S). This is because a creak from the left or silence indicates that j likely did not open the right door. The fact that CL or S are not likely to be the outcome of noise is due to i's belief that j believes with a fairly large probability that the tiger is behind the right door. When agent i is less certain about the location of the tiger (in region 23 [*Policy: L(); OL(GR,CL) L(?)*]), it has to observe a creak from the left, in addition to the growl from the right, to open the left door.

In regions of the policy (Fig. 3(b)) where the Gaussian mean is closer to 1, it is possible to get misleading information and still make a decision of opening doors. For example, in the region 25 [*Policy: L(); OL(GR,CR) L(?)*], the agent opens the left door even though it hears a growl from the right and a creak from the right. Because in this region the mean is closer to 1, i thinks that j is misinformed. In other words, for the physical state that the tiger is on the right, i likely believes that j believes the tiger to be on the left. Hence, the creak from the right is interpreted to be a result of this misinformation, thereby reinforcing i's belief that the tiger is on the right.

As the reliability of the creaks increases to 99.90%, we see that agent i starts exhibiting a more complex behavior. Sometimes it is guided only by creaks (regions 29 and 30), when it knows that j's opening the doors is a good predictor of the tiger's location. Sometimes, however, when i is quite sure about the tiger's location and it thinks that j is misinformed, i can ignore the creaks and base its decisions only on growls (regions 1, 2, 11, and 12).

Let us now turn our attention to regions 5 and 17 and their symmetric counterparts – regions 6 and 18 in Fig. 3(c). We focus on only the region 5, but the arguments for the other regions are analogous. For region 5, i believes j is informed and it believes that the tiger is likely on the right. For this region, the agent opens the left door if it hears a creak from the left (since it was fairly certain that the tiger was behind right door and the tiger persists after opening doors) or on hearing a growl from the right and no creaks. It opens the right door only if it hears a growl from the left along with creak from the right. Since i believes j is informed, a creak from the right indicates that j likely opened the right door which means that j believed that the tiger was on the left. An observation of GL further reinforces i's belief that the tiger is indeed on the left, causing i to open the right door.

When the reliability of the creaks increases to 99.97%, the size of the regions that utilize only the creaks to guide the actions, increase



(a) Creak accuracy: 33%

(b) Creak accuracy: 70%

(c) Creak accuracy: 99.9%

(d) Creak accuracy: 99.97%

1	OL(); OL(*,*)	2	OR(); OR(*,*)
3	L(); OL(*,CL | GR,S) OR(*,CR) L(?)	4	L(); OL(*,CL) OR(*,CR | GL,S) L(?)
5	L(); OL(*,CL | GR,S) OR(GL,CR) L(?)	6	L(); OL(GR,CL) OR(*,CR | GL,S) L(?)
7	L(); OL(*,CL | GR,S) L(?)	8	L(); OR(*,CR | GL,S) L(?)
9	L(); OL(GR,CL | GR,S) L(?)	10	L(); OR(GL,CR | GL,S) L(?)
11	L(); OL(GR,*) L(?)	12	L(); OR(GL,*) L(?)
13	L(); OL(GR,CR | GR,S) L(?)	14	L(); OR(GL,CL | GL,S) L(?)
15	L(); OL(*,CR | GR,S) L(?)	16	L(); OR(*,CL | GL,S) L(?)
17	L(); OL(*,CR | GR,S) OR(GL,CL) L(?)	18	L(); OL(GR,CR) OR(*,CL | GL,S) L(?)
19	L(); OL(*,CR | GR,S) OR(*,CL) L(?)	20	L(); OL(*,CR) OR(*,CL | GL,S) L(?)
21	L(); OL(*,CL) OR(GL,CR) L(?)	22	L(); OL(GR,CL) OR(*,CR) L(?)
23	L(); OL(GR,CL) L(?)	24	L(); OR(GL,CR) L(?)
25	L(); OL(GR,CR) L(?)	26	L(); OR(GL,CL) L(?)
27	L(); OL(*,CR) OR(GL,CL) L(?)	28	L(); OL(GR,CR) OR(*,CL) L(?)
29	L(); OL(*,CL) OR(*,CR) L(?)	30	L(); OL(*,CR) OR(*,CL) L(?)
31	L(); OL(GR,CL) OR(GL,CR) L(?)	32	L(); OL(GR,CR) OR(GL,CL) L(?)
33	L(); L(*,*)		

(e) Policy of agent i. ? indicates all other observations and * indicates all possible values

Figure 3: Figures (a-d) show the projections of the value functions of agent i, with varying creak reliabilities. The belief regions are marked with numbers that index into the list of policy trees in table (e)

(see regions 29 and 30), as should be expected. Additionally, new regions (3, 4, 19, and 20) appear as part of the policy.

5. CONCLUSION

We presented a method that exploits behavioral equivalence among other agents' possible models to exactly solve finitely nested interactive POMDPs. By aggregating the behaviorally equivalent models into equivalence classes, we transformed the infinite interactive state space of I-POMDPs into a finite one, thereby allowing the application of standard POMDP solution techniques for solving I-POMDPs. Our method works because the discretization of the state space preserves the value of the plans. We applied our method to the multi-agent tiger problem, and presented a series of policies for various values of the parameters of the problem. Our aim is to understand the interplay between an agent's belief about other agents' beliefs and its observations, in arriving at its optimal plan of action. As we demonstrated, an agent's optimal policy changes drastically as its capability about observing others' behaviors improves. We believe it may be possible to combine different behavioral equivalence regions together to approximately represent an agent's behavior and reason about them. This is one possible avenue of future work.

Acknowledgements

This research was supported by the NSF CAREER award IRI-9702132 and NSF award IRI-0119270. Prashant Doshi would also like to acknowledge the support of a UIC University Fellowship.

APPENDIX

PROOF OF PROPOSITION 1.

$$\tilde{b}^t_{i,l}(ecis^t) = Pr(ecis^t \mid o^t_i, a^{t-1}_i, \tilde{b}^{t-1}_{i,l}) = \frac{Pr(ecis^t, o^t_i \mid a^{t-1}_i, \tilde{b}^{t-1}_{i,l})}{Pr(o^t_i \mid a^{t-1}_i, \tilde{b}^{t-1}_{i,l})}$$

$$= \beta \sum_{ecis^{t-1}} \tilde{b}^{t-1}_{i,l} Pr(ecis^t, o^t_i \mid a^{t-1}_i, ecis^{t-1})$$

$$= \beta \sum_{ecis^{t-1}, a^{t-1}_j} \tilde{b}^{t-1}_{i,l} Pr(a^{t-1}_j \mid ecis^{t-1}) \times Pr(ecis^t, o^t_i \mid a^{t-1}_i, a^{t-1}_j, ecis^{t-1})$$

$$= \beta \sum_{ecis^{t-1}, a_j^{t-1}} \tilde{b}_{i,l}^{t-1} Pr(a_j^{t-1} \mid \langle s^{t-1}, M_{j,k}^{t-1} \rangle) \times Pr(ecis^t, o_i^t \mid a^{t-1}, ecis^{t-1})$$

$$= \beta \sum_{ecis^{t-1}, a_j^{t-1}} \tilde{b}_{i,l}^{t-1} Pr(a_j^{t-1} \mid M_{j,k}^{t-1}) Pr(o_i^t \mid ecis^t, a^{t-1}, ecis^{t-1}) \times Pr(ecis^t \mid a^{t-1}, ecis^{t-1})$$

$$= \beta \sum_{ecis^{t-1}, a_j^{t-1}} \tilde{b}_{i,l}^{t-1} Pr(a_j^{t-1} \mid M_{j,k}^{t-1}) O_i(s^t, a^{t-1}, o_i^t) \times Pr(\langle s^t, M_{j,k}^t \rangle \mid a^{t-1}, \langle s^{t-1}, M_{j,k}^{t-1} \rangle)$$

$$= \beta \sum_{ecis^{t-1}, a_j^{t-1}} \tilde{b}_{i,l}^{t-1} Pr(a_j^{t-1} \mid M_{j,k}^{t-1}) O_i(s^t, a^{t-1}, o_i^t) \times T_i(s^t, a^{t-1}, s^{t-1}) \sum_{o_j^t} \tilde{\tau}(M_{j,k}^t, a_j^{t-1}, o_j^t, M_{j,k}^{t-1}) \times O_j(o_j^t \mid a^{t-1}, s^t)$$

Thus belief at time t can always be expressed in terms of belief and actions at time $t-1$ and the observations at time t. $\square$

PROOF OF PROPOSITION 2 BY INDUCTION. To keep the proof short, we will assume that only the belief of the other agent is unknown, i.e. all other parameters of the model are known. Thus Eq. 6 can be written as $\tilde{b}_i(s, \mathbb{B}_{j,k}) = \int_{b_j \in \mathbb{B}_{j,k}} b_i(s, b_j) db_j$

Let b_i be an arbitrary belief of agent i. Let $\{\mathbb{B}_{j,1}, \mathbb{B}_{j,2}, ..., \mathbb{B}_{j,n}\}$ be the collection of equivalence classes of agent j's belief. Each class $\mathbb{B}_{j,k}$ is a set of beliefs of j such that the action a_j^k is optimal for each belief. Thus $\forall_{b_j \in \mathbb{B}_{j,k}} ER(s, b_j, a_i) = R(s, a_i, a_j^k)$, because a_j^k is optimal for all $b_j \in \mathbb{B}_{j,k}$.

Basis Step: We show that the horizon 1 value remains unchanged when i's original belief is replaced by its belief over the equivalence classes.

$$Q^1(b_i, a_i) = \int_{is_i} b_i(is_i) ER(is_i, a_i) = \sum_s \int_{b_j \in B_j} b_i(s, b_j) ER(s, b_j, a_i)$$

From the definition of equivalence classes,

$$Q^1(b_i, a_i) = \sum_s \left\{ \int_{b_j \in \mathbb{B}_{j,1}} b_i(s, b_j) ER(s, b_j, a_i) + ... + \int_{b_j \in \mathbb{B}_{j,n}} b_i(s, b_j) ER(s, b_j, a_i) \right\}$$

Note that $ER(s, b_j, a_i) = \sum_{a_j} R(s, a_i, a_j) Pr(a_j | b_j)$, and for all $b_j \in \mathbb{B}_{j,n}$ the $Pr(a_j|b_j)$ will be 1 for some a_j^n (assuming a deterministic policy) and 0 for others. Therefore,

$$Q^1(b_i, a_i) = \sum_s \left\{ \int_{b_j \in \mathbb{B}_{j,1}} b_i(s, b_j) R(s, a_i, a_j^1) + ... + \int_{b_j \in \mathbb{B}_{j,n}} b_i(s, b_j) R(s, a_i, a_j^n) \right\}$$

$$= \sum_s \left\{ R(s, a_i, a_j^1) \int_{b_j \in \mathbb{B}_{j,1}} b_i(s, b_j) + ... + R(s, a_i, a_j^n) \int_{b_j \in \mathbb{B}_{j,n}} b_i(s, b_j) \right\}$$

Using Eq. 6,

$$Q^1(b_i, a_i) = \sum_s \Big\{ R(s, a_i, a_{j,1}) \tilde{b}_i(s, \mathbb{B}_{j,1}) + ... + R(s, a_i, a_{j,n}) \tilde{b}_i(s, \mathbb{B}_{j,n}) \Big\} = \sum_{s,k} \tilde{b}_i(s, \mathbb{B}_{j,k}) R(s, a_i, a_{j,k}) = \tilde{Q}^1(\tilde{b}_i, a_i)$$

Because the Q values remain unchanged, maximizing over them will also yield identical values.

Inductive Hypothesis: Let us assume that $\forall a_i, b_i \; Q^n(b_i, a_i) = \tilde{Q}^n(\tilde{b}_i, a_i)$ where $\tilde{b}_i$ is related to b_i using Eq. 6. Because the Q values are identical, the N horizon value function also remains unchanged.

Inductive Proof:

$$Q^{n+1}(b_i, a_i) = \int_{is_i} b_i(is_i) ER(is_i, a_i) + \gamma \sum_{o_i} Pr(o_i | b_i, a_i) V^N(SE(b_i, a_i, o_i)) = Q^1(b_i, a_i) + \gamma \sum_{o_i, is_i} Pr(o_i | is_i, a_i) \times b_i(is_i) U^n(SE(b_i, a_i, o_i))$$

From the basis step,

$$Q^{n+1}(b_i, a_i) = \tilde{Q}^1(\tilde{b}_i, a_i) + \gamma \sum_{o_i, is_i, a_j} Pr(o_i | is_i, a_i, a_j) \times Pr(a_j | b_j) b_i(is_i) U^n(SE(b_i, a_i, o_i)) = \tilde{Q}^1(\tilde{b}_i, a_i) + \gamma \sum_{o_i, s} \int_{b_j} \sum_{a_j} Pr(o_i | is_i, a_i, a_j) \times Pr(a_j | b_j) b_i(s, b_j) U^n(SE(b_i, a_i, o_i))$$

From the definition of equivalence classes,

$$Q^{N+1}(b_i, a_i) = \tilde{Q}^1(\tilde{b}_i, a_i) + \gamma \sum_{o_i, s, \mathbb{B}_{j,k}} \int_{b_j \in \mathbb{B}_{j,k}} \sum_{a_j} Pr(o_i | is_i, a_i, a_j) \times Pr(a_j | b_j) b_i(s, b_j) U^n(SE(b_i, a_i, o_i))$$

Using the BNM and BNO assumptions of the I-POMDP framework [8],

$$Q^{n+1}(b_i, a_i) = \tilde{Q}^1(\tilde{b}_i, a_i) + \gamma \sum_{o_i, s, \mathbb{B}_{j,k}} \int_{b_j \in \mathbb{B}_{j,k}} Pr(o_i | s, a_i, a_j^k) \times b_i(s, b_j) U^n(SE(b_i, a_i, o_i))$$

From Eq. 6,

$$Q^{n+1}(b_i, a_i) = \tilde{Q}^1(\tilde{b}_i, a_i) + \gamma \sum_{o_i, s, \mathbb{B}_{j,k}} Pr(o_i | s, a_i, a_j^k) \times \int_{b_j \in \mathbb{B}_{j,k}} b_i(s, b_j) U^n(SE(b_i, a_i, o_i)) = \tilde{Q}^1(\tilde{b}_i, a_i) + \gamma \sum_{o_i, s, \mathbb{B}_{j,k}} Pr(o_i | s, a_i, a_j^k) \times \tilde{b}_i(s, \mathbb{B}_{j,k}) U^n(SE(b_i, a_i, o_i))$$

Using $Pr(o_i|s,a_i,\mathbb{B}_{j,k}) = \sum_{a_j} Pr(o_i|s,a_i,a_j,\mathbb{B}_{j,k})Pr(a_j|\mathbb{B}_{j,k}) = Pr(o_i|s,a_i,a_{j,k})$ and the inductive hypothesis,

$$\begin{aligned} Q^{n+1}(b_i,a_i) &= \tilde{Q}^1(\tilde{b}_i,a_i) + \gamma \sum_{o_i,s,\mathbb{B}_{j,k}} Pr(o_i|s,a_i,\mathbb{B}_{j,k}) \\ &\quad \times \tilde{b}_i(s,\mathbb{B}_{j,k})\tilde{U}^n(SE(\tilde{b}_i,a_i,o_i)) \\ &= \tilde{Q}^1(\tilde{b}_i,a_i) + \gamma \sum_{o_i} Pr(o_i|a_i,\tilde{b}_i) \\ &\quad \times \tilde{U}^n(SE(\tilde{b}_i,a_i,o_i)) \\ &= \tilde{Q}^{n+1}(\tilde{b}_i,a_i) \end{aligned}$$

Because the Q values are identical under the mapping, the values of the beliefs over the equivalence classes of interactive states remain unchanged. We have assumed in the proof that agent j's policies are deterministic, i.e., for every class $\mathbb{B}_{j,k}$ there is one optimal action $a_{j,k}$. The proof extends in a straightforward manner when there is more than one optimal action for a class. □

A. REFERENCES

[1] R. Becker, S. Zilberstein, and V. Lesser. Decentralized markov decision processes with event-driven interactions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-04)*, 2004.

[2] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes. In *Mathematics of Operations Research*, volume 27, No. 4, pages 819–840, 2002.

[3] D. S. Bernstein, E. A. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized pomdps. In *Proceedings of the The Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, 2005.

[4] A. R. Cassandra, M. L. Littman, and N. L. Zhang. Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. In *Uncertainty in Artificial Intelligence*, Rhode Island, Providence, 1997.

[5] P. Doshi and P. Gmytrasiewicz. Approximating state estimation in multiagent settings using particle filters. In *Proceedings of the Fourth International Conference on Autonomous Agents and Multi Agent Systems (AAMAS-05)*, 2005.

[6] P. Doshi and P. Gmytrasiewicz. A particle filtering based approach to approximating interactive pomdps. In *Proceedings of the The Twentieth National Conference on Artificial Intelligence (AAAI-05)*, 2005.

[7] R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in markov decision processes. In *Artificial Intelligence Journal*, 2003.

[8] P. J. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multi-agent settings. In *Journal of AI Research*, 2005.

[9] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-04)*, 2004.

[10] J. C. Harsanyi. Games with incomplete information played by 'bayesian' players. In *Management Science*, pages 14(3):159–182, Nov 2005.

[11] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 2, 1998.

[12] R. Nair, M. Tambe, M. Yokoo, D. Pynadath, and S. Marsella. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *Proceedings of International Joint Conference in Artificial Intelligence (IJCAI)*, 2003.

[13] J. M. Porta, M. T. Spaan, and N. Vlassis. Value iteration for continuous-state pomdps. In *IAS Technical report IAS-UVA-04-04*, 2004.

[14] P. Poupart and C. Boutilier. Value directed compression of pomdps. In *Seventeenth Annual Conference on Neural Information Processing Systems*, 2003.

[15] D. V. Pynadath and M. Tambe. Multiagent teamwork: Analyzing the optimality and complexity of key theories and models. In *Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2002.

[16] R. Smallwood and E. Sondik. The optimal control of partially observable markov decision processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[17] S. Thrun. Monte carlo POMDPs. In S. Solla, T. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 1064–1070. MIT Press, 2000.

On Proactivity and Maintenance Goals

Simon Duff
RMIT University
Melbourne, Australia
sduff@cs.rmit.edu.au

James Harland
RMIT University
Melbourne, Australia
jah@cs.rmit.edu.au

John Thangarajah
RMIT University
Melbourne, Australia
johthan@cs.rmit.edu.au

ABSTRACT

Goals are an important concept in intelligent agent systems, and can take a variety of forms. One such form is *maintenance goals*, which, unlike *achievement goals*, define states that must remain true, rather than a state that is to be achieved. Maintenance goals are generally restricted to acting as trigger conditions for goals or plans, and often take no part in any deliberation process. These goals are *reactive* and are only acted upon when the maintenance conditions are no longer true. In this paper, we study maintenance goals that are *proactive*, in that the agent system needs to not only react when the maintenance conditions fail, but also anticipate the failures of these conditions, and act in order to avoid them failing. This can be done by performing actions that prevent the condition from failing, or suspending goals that will cause the maintenance conditions to fail. We provide a representation for maintenance goals that captures both their reactive and proactive aspects, algorithms that identify in advance where maintenance conditions may not hold, and mechanisms for enabling preventative actions in such situations. We also provide some experimental results on an implementation of these ideas.

Categories and Subject Descriptors

D.3.1 [**Programming Languages**]: Formal Definitions and Theory; I.2.5 [**Artificial Intelligence**]: Programming Languages and Software; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Representations (procedural and rule-based)*

Keywords

Agent programming languages, Formal models of agency, Task and resource allocation in agent systems

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

Goals are a crucial concept in agent programming languages. Agent properties such as proactiveness, reactivity, autonomy and social ability can be described in terms of the goals of the agent. A *proactive* agent is required to actively work towards achieving its goals, while a *reactive* agent is required to adapt to environmental changes in order to maximise its ability to achieve its goals. An *autonomous* agent is free to choose how to achieve its goals, how to resolve conflicts between its goals and whether to drop goals which appear to be unachievable. A *social* agent may choose to adopt a goal on a request from another agent, or to request another agent to adopt one of its goals. Accordingly, there are a number of agent systems in which goals are a central concept, such as 3APL[5], KAOS[4], and Tropos[7].

Goals can hence be seen as a link between the agent's reasoning and its actions; actions are only undertaken in order to achieve one or more of the agent's goals, and changes in the status of the agent's goals are reflected in changes in its actions.

For the most part, goals in agent systems are *achievement goals*, i.e. goals which are not currently true, and which the agent, by means of pursuing the appropriate actions, acts to make true. However, the agent has no obligation once the goal becomes true, in that there is no requirement to act to ensure that goal remains true.

There are some systems (such as Jadex[9] and JAM[8]) which include *maintenance goals* as well as achievement goals. In such systems, a maintenance goal is one which is initially true, and if it becomes false, the agent will take action to make it become true. Hence such goals act as a trigger, in that it is only when they become false that any action is taken. We refer to such goals as *reactive maintenance goals*, as no action is taken for such goals until the goal becomes false.

In this paper we extend the notion of maintenance goals to include *proactive* behaviour as well. In other words, an agent may take action not only to bring about an achievement goal or in response to a maintenance goal becoming false, but also to ensure that a maintenance goal will remain true. Hence the agent reasons about possible future events, and perform actions to avoid maintenance goals becoming false (as well as possibly taking actions if any such goals do become false).

For example, consider an agent controlling a Mars Rover, which has a number of target locations it is required to visit (the agent's achievement goals), while ensuring that certain conditions related to the safe operation of the rover are maintained (the agent's maintenance goals). A typical

example of a maintenance goal could be to ensure that the rover has sufficient battery charge to perform its activities, recharging when it has less than 20% of its battery remaining [1]. The rover knows several locations where it can recharge its battery. The process of moving consumes charge, and the electrical usage of the rover may be predictable in some cases (for example, where the rover has visited some location before). In such a case, the rover can determine how much charge it will use, and the expected battery level it will have upon reaching its destination. It would appear rational for a rover to anticipate that its maintenance condition may be unsatisfied if it attempts to move towards the location immediately – instead, it should take the opportunity to recharge its battery whilst it is near the charger, prior to moving to a distant location.

A second example can be found in the domain of *autonomic computing*. An intelligent operating system may be given the goal to maintain the availability of a cluster of machines by preventing all machines being shut down at once. There is no obvious mechanism that would allow a reactive maintenance goal to ensure that all machines are active, without first allowing all machines to be shut down, recognising that the maintenance condition is violated, and restarting a machine to re-establish the maintenance condition. On the other hand, a proactive maintenance goal would prevent the last machine from being shut down. We believe that the domain of autonomic computing is a rich one for proactive maintenance goals.

A final example consists of a boiler that produces some output power. An agent is able to control the output power by increasing or decreasing the temperature of the boiler. This example can be used to illustrate how maintenance goals can be made to interact. As a safety measure, the agent may have a maintenance goal to ensure that the boiler's temperature never exceeds some amount. There may also be a second maintenance goal that the output power always be greater than some minimum value. The agent may have several achievement goals, such as providing power for several devices, which require the agent to increase the temperature to produce enough power. A rational agent may limit the number of devices that can concurrently be powered, to ensure that the temperature required is not too elevated. The rational agent may also choose to ensure that there is always some device being powered by the boiler, to avoid wasted energy.

Hence, achievement goals drive the agent to perform actions to produce a desired state; maintenance goals either trigger corrective actions to be taken, or constrain the choice of actions so that the maintenance goals will not be violated.

This paper is organised as follows. In Section 2, we provide background material to a number of elements concerning the agent framework support for proactive maintenance goals. Next, in Section 3 we provide a detailed explanation of the requirements and mechanisms an agent system should incorporate to support our ideals of proactive maintenance goals. A discussion of mechanisms for implementing proactive behaviour follows in Section 4. We have done some preliminary experiments to evaluate our work which we present in Section 5. We conclude with possible future directions in Section 6.

2. MAINTENANCE GOALS

The earliest examples of maintenance goals (such as [6]) are purely reactive, i.e. they act as triggers to adopt an extra achievement goal, and only when the stated condition becomes false. In the Mars rover example, a reactive maintenance goal of maintaining a battery level of at least 20% will only cause the agent to act once the level drops below this amount. Hence the maintenance goal can be completely characterised as a trigger which adds an achievement goal such as "recharge the battery". In a system such as JACK[2], this is not very different from adding the negation of the maintenance goal as a context condition for the plan which fills the tank.

The life cycle of such reactive maintenance goals is discussed in some detail in [1]. As discussed in that paper, "a maintenance goal is introduced to observe and maintain some world state as long as the goal exists". Whilst the discussion of goal types and the way in which goals are managed during the execution cycle is quite comprehensive, their use of maintenance goals remains reactive.

A more active approach to maintenance goals includes not only triggering some recovery action once the goal is violated, but *taking actions in anticipation of the maintenance goal becoming false*. As mentioned above, this will presumably lead to more rational behaviour, such as the battery-charging activity of the robots discussed in [1] taking place shortly before the low power threshold is reached, rather than just after it. However, there is no point in recharging the robot too often, such as when the battery is 85% charged, just because the robot happens to be nearing a charging point. In other words, the agent has to have a reasonable expectation that the maintenance goal is likely to be violated in the near future; action is then taken to ensure that it is not violated.

In some sense, this approach assumes that it is important to prevent the maintenance goal from becoming false. For example, consider a maintenance goal to maintain the pressure of a gas cylinder at no more than a certain amount. Whilst emergency action is clearly needed if the pressure exceeds this value, rational management of this property would also include actively preventing the pressure from getting too close to the limit.

Hence the proactive approach takes action not only when the maintenance goal is violated, but also when the agent anticipates that it will be violated in the near future. In order to perform such reasoning, the agent needs to be aware of the effects of other goals on the maintenance condition. This can be of the form of altering the belief of an agent, or consuming or producing some resource involved in the maintenance condition. Therefore, we require the effects and required resources of goals to be explicitly represented in the structure of the agent goals, as well as having these structures utilised in the agent execution cycle.

It should be noted that maintenance goals are a means of improving the rationality of the agent by using a richer class of goals. Hence they can be seen as a way of more closely approximating perfectly rational behaviour whilst remaining intuitively simple. The main thrust of this paper is to show that the extra subtlety gained comes at a cost of comparatively little efficiency.

3. REPRESENTATION

Our agents are based on standard implementations of the Belief-Desire-Intention (BDI)[11]. In particular, our central notions are *beliefs*, *goals* and *plans*.

The agents we develop have a pre-defined set of plans that are utilised at run-time to satisfy goals. The agents also have belief-bases which contains facts about the world known to the agents. In this section we describe a representation for an agent's beliefs, plans, achievement goals and maintenance goals. We build on some of our earlier work to provide a representation for achievement goals and plans [12, 13], and provide a new structure for maintenance goals.

Beliefs can be represented in many different forms. For example, in AgentSpeak [10] they are represented in first order logic. In our framework, we present a restricted form of beliefs in order to better explain the reasoning mechanisms in later sections. We represent a belief as a tuple, ⟨ *Name, Type, Value* ⟩, where *Name* is an alphanumeric string which allows the agent's beliefs to be uniquely referenced, *Type* defines the type of object the Value attribute represents, and *Value* is the actual data representing the belief. Some examples of beliefs are:

⟨FuelLevel, Number, 60⟩
⟨GripperEmpty, Boolean, True⟩
⟨RoverID, Text, Rover1⟩

Plans are lists of instructions an agent can follow to potentially establish a desired state with respect to some goal. A plan is a tuple of the form ⟨*Type, Pre-Condition, Effects, Plan Body*⟩. *Type* is an alphanumeric string which identifies the type of the plan. The *Pre-Condition* is an expression over the agent's beliefs which must be satisfied before this plan can be executed. If it is not satisfied, this plan cannot be instantiated. In systems like JACK, pre-conditions are represented as *context conditions*. The *Effects* denote the changes to the agent's belief set that will occur if the plan is executed successfully.

The *Plan Body* consists of sub-goals and actions, which can be combined by either sequencing them (denoted by ;), or by executing them in parallel (denoted by ‖). Sub-goals are treated like top-level goals. Actions can be any arbitrary code specified by the user.
An example of a plan is as follows:

Type *RefuelRoverPlan*
Pre-Condition $FuelLevel \geq 50$
Plan Body {*FindRefuelPoint ; MoveToRefuelPoint ; Refuel*}
Effects {*Update(FuelLevel,100)*}

Achievement goals are states of the world that the agent works toward establishing. We follow the definition of [15] and represent them as follows.
⟨*Type, Success Condition, Failure Condition, Plans*⟩
Type is an alphanumeric string that uniquely identifies a goal. *Success Conditions* are the desired outcomes of the goal and when true indicates the goal has been successfully achieved. The *Failure Condition* when true indicates the goal cannot be achieved by the agent by any means possible, and hence the agent should no longer pursue it.

The *Plans* attribute is a set of plan names which achieves the success condition. When an instance of an achievement goal is created, the agent selects a plan from this set and if its pre-condition is satisfied, begins executing it. If the plan fails, then another plan from the plan set is selected and tried. If no applicable plan exists, the goal is dropped. This mechanism is typical of BDI agent systems.
An example of an achievement goal is:

Type: GatherSoilSample
Success Condition: $ObtainedSoilSamples = true$
Failure Condition: $GripperBroken = true || RoverImmobile = true$
Plans: $\{GatherSoilAtXPlan, GatherSoilAtYPlan\}$

This example describes a goal whose purpose is to obtain a soil sample, and it has two options in terms of the location which has soil (X and Y), hence two plans. If either plan succeeds, the rover will have obtained said sample, and the agent's belief of $ObtainedSoilSample == true$ will be satisfied. The goal can (and should) be abandoned if the gripper, used to gather soil, is broken, or if the rover is incapable of moving.

As discussed earlier, our representation of maintenance goals needs to support both proactive reactive aspects.

Maintenance goals are tuples of the form
⟨*Type, Maintenance Condition, Failure Condition, Recovery Plans, Preventative Goal* ⟩.
The *Type* and *Failure Condition* is similar to that of an achievement goal indicating the type of the goal and the condition which when true makes the goal no longer applicable.

The condition (belief) the goal requires the agent to maintain is given by *Maintenance Condition*. If this condition becomes false, the agent selects an applicable plan (i.e., one whose pre-condition is currently true) from *Recovery Plans*, in order to re-establish the maintenance condition.

In contrast to purely reactive maintenance goals, if the agent determines that the maintenance condition is currently true, but is anticipated to become false in the future, it will select and execute an applicable goal from *Preventative Goal*. These goals are designed to ensure that the maintenance condition remains true.

An example of a maintenance goal follows, which aims to ensure that the rover has an adequate battery level, recharging if the level drops below a certain threshold.

Maintenance Condition: $BatteryLevel \geq 20\%$
Failure Condition: $ChargerBroken = True$
Recovery Plans: {*RechargeBatteryPlan*} (Move a robot to a charging station and allow it to recharge its battery)
Preventative Goal: {*RechargeBatteryGoal*}

A second example of a maintenance goal ensures the speed of the rover does not exceed a maximum operating speed.

Maintenance Condition: $Speed \leq 100km/hr$
Failure Condition: {}
Recovery Plans: {*ReduceSpeedPlan*} (Slow down vehicle)
Preventative Goal: {}

A third example of a maintenance goal originates from the domain of robot soccer. An agent controls the behaviour of a 'defender' robot player, whose role is currently to prevent a opposition 'forward' from obtaining the ball.

Maintenance Condition: $OpponentHasBall == false$
Failure Condition: {}
Recovery Plans: {*TackleOpponentPlan*}
Preventative Goal: {*BlockOpponentGoal*}

In such a case, the maintenance goal behaviour of the defender differs depending on who has control of the ball. If its opponent has the ball, the agent must 'recover' its maintenance goal of preventing this, and so must tackle the opponent to cause it to lose control of the ball. However, if the opponent does not have control of the ball, one approach is to block the opponent, by moving to a location between the opponent and the ball, ready to intercept a pass or prevent the attacker from moving to the ball. In this example, the behaviour of the agent is different depending on if the agent is recovering from or preventing maintenance goal failure.

In addition to the above attributes, at run-time plan and goal instances will further require an *instance name* which is a label that provides a unique handle to them.

4. REASONING ALGORITHMS

In the BDI theory [11], goals of an agent are required to be consistent. Thangarajah et al.[12, 13, 14] provide mechanisms for ensuring this property holds for achievement goals. We build on this work to incorporate maintenance goals into the set of goals an agent may pursue, and provide mechanisms that ensure that all goals (maintenance and achievement) that an agent pursues are consistent with each other.

We consider two aspects in which an agent's maintenance goal may conflict; conflicts due to resource limitations, and conflicts due to the effects that goals produce. Our aim is to ensure that maintenance goals remain consistent with other goals with respect to these aspects.

In BDI-like systems, a goal can be decomposed into plans (that achieve the goal) and sub-goals. Often, there are several ways in which a goal can be realised, represented by a number of different plans and sub-goals. When a plan fails, a new plan is selected and tried. It is only when there are no applicable plans left that the agent declares a goal has failed. This leads to a natural tree structure which is termed a *goal-plan* tree [14], which indicates the possible ways in which the top level goal can be established. Every goal or sub goal in the tree has at least one plan as a child, and each plan may have a number of sub-goals. The plan nodes of a goal can be viewed as an "OR" since only one of the plans need to succeed for the goal to succeed, and the sub goal nodes can be viewed as "AND" nodes as they all must succeed for the plan to succeed.

Since there are a number of paths that an agent may take to satisfy a goal, the agent cannot be certain as to the resources consumed and the effects that are produced as a result of a goal. In [12, 13, 14], the notion of summary information is used to capture at an abstract level the resource requirements and the effects of executing a goal.They do this by requiring that each plan and goal in a goal-plan tree have information pertaining to the effects that the plan brings about, as well as the resources that each plan requires[1].

They use the goal-plan tree structure and the notion of summary information [3] to provide detailed algorithms that derive the *necessary* (but not sufficient) and *possible* resources for a goal [14], as well as the *definite* and *potential* effects of a goal [13].

Necessary resources are the resources that are common to all possible paths in achieving the top level goal. *Possible resources* are the maximum set of resources that may be required for an agent to achieve its goal, given any path through the goal-plan tree. The possible resource takes plan failure into consideration, and is often an exaggeration, but provides a clear upper bound. The necessary resources are a subset of the possible resources, which indicates that all necessary resources are possible resources as well. We refer to the combination of necessary and possible resources as *resource summaries*.

The *definite effects* of a goal are the effects that are certain to come about no matter which path through the goal-plan tree the agent follows in achieving a goal. Note that this process ignores the possibility of a goal becoming satisfied by some external process. The *potential effects* of a goal are a set of all the effects that could potentially occur, given that any plan could be selected, and that plan may fail and another be selected and so on. We refer to the combination of definite and potential effects as *effect summaries*.

The summary information is initially derived at compile time and then later dynamically updated as the agent completes goals and plans (or if they fail). With respect to resources, they also distinguish between *consumable* and *reusable* resources which are resources that once used are no longer available and those that once used can be re-used again respectively. Their algorithms consider these resource types as the behaviour varies between them.

In our work, we will use these notions of *resource summaries* and *effect summaries*, to determine if a maintenance goal is in conflict with an achievement goal. We rely on the methods developed in [13, 14] to obtain this summary information, and also on the conflict detection mechanisms between achievement goals. We do not reproduce these algorithms here, but show how they are incorporated into our own algorithms ahead.

If we consider an agent that supports proactive maintenance goals, several factors are required beyond that found in most agent systems. As described in Section 3, we require that each plan be tagged with additional information describing the effects the plans bring about, as well as the resources they use. The success condition of achievement goals defines the effects of the goal.

We assume that each agent has set of currently active goals $\mathcal{G}$ which includes both the set of maintenance goals ($\mathcal{G}_m$) and the set of achievement goals ($\mathcal{G}_a$) (i.e. $\mathcal{G} = \mathcal{G}_m \cup \mathcal{G}_a$). The list of goals which are waiting for another goal/plan to complete before it can be pursued, are stored in a *Suspended Goals (SG)* set, where each entry is the suspended goal and the list of goals/plans that it is waiting on. Finally, the agent requires a resource table to determine the current resources available for its goals. We call this resource table $\mathcal{R}$, containing tuples which are name, value pairs describing the current availability of its resources.

Goals are added to the agent system one at a time[2]. Of course once the goals have been added, when possible, the agent will execute goals in parallel. Section 4.1 looks at the mechanisms for ensuring that the goals are consistent when a new achievement goal is added to the system and section 4.2 investigates the addition of a new maintenance goal.

4.1 Adding Achievement Goals

Before an agent can pursue a new achievement goal, G_a, it must determine if it can be executed without causing any of the agent's other goals to fail. Four elements must be checked to ensure that this is the case – an agent must determine if a conflict can occur over (a) the effects the goal can cause, and (b) the resources that a goal requires. In each of these cases, the agent must check if conflicts arise between the new goal and existing achievement goals as well as existing maintenance goals.

The agent's first step is to determine the effects of G_a. This can be done using the algorithms developed in [13]. This algorithm results in the identification of two sets of effects – the *definite* effects(DE) of a goal, and the *potential*

[1]Note that the resources and effects that are the result of child-nodes are not included at the parent level.

[2]A goal set maybe given to an agent but it processes them one after the other.

effects(PE) of a goal. Using the algorithms in [13] this information can be used to determine if a conflict can arise between G_a and any achievement goals in $\mathcal{G}_a$. We then provide an algorithm to check for conflict between G_a and the set of maintenance goals $\mathcal{G}_m$.

G_a must be checked for consistency with every maintenance goal in $\mathcal{G}_m$. This requires the identification of the definite and potential effects of G_a. The effects of G_a are combined with the effects of $\mathcal{G}_a$, producing the set of definite and potential effects for the agent's current goals, and the new achievement goal. This set is represented as $< DE_A\prime, PE_A\prime >$. A maintenance goal is consistent if the definite and potential effects that the goal brings about, does not cause the maintenance condition of the maintenance goal to fail. A maintenance goal is inconsistent with the achievement goal if the definite effects alone cause the maintenance condition to fail. In situations where the definite effects do not cause the maintenance condition to fail, but some potential effects do cause failure, the agent is uncertain if the addition of G_a will cause the maintenance goal to become inconsistent.

function *check-effect-consistency* ($\mathcal{G}_m$, $< DE_A\prime, PE_A\prime >$)
```
    number of maintenance goals := 0
    number of maintenance goals satisfied := 0
    number of maintenance goals unsatisfied := 0
    for each maintenance goal mg in G_m
        mc := mg.maintenance condition
        if mc is consistent with DE_A' + PE_A' then
            number of maintenance goals satisfied ++
        if mc is ¬ consistent with DE_A' then
            number of maintenance goals unsatisfied ++
        number of maintenance goals ++
    if number-of-maintenance-goals-satisfied ==
        number-of-maintenance-goals then
        // all maintenance goals are definitely consistent
        return consistent
    else if number of maintenance goals unsatisfied > 0 then
        // some or all maintenance goals are not consistent
        return inconsistent
    else
        return uncertain
```

Goals can also conflict over the resources they require to execute successfully. The work in [12] provides a mechanism for determining the *necessary* (N) and *possible* (P) resource requirements of a goal. The necessary and possible resources of G_a (which is a tuple, $< N_a, P_a >$), are combined with the necessary and possible resources of all achievement goals in $\mathcal{G}_a$ ($< N, P >$), producing $< N_a\prime, P_a\prime >$ which describes the minimum and maximum resource needs if G_a is added to the goal set.

The agent must now determine if the maintenance goals in $\mathcal{G}_m$ are consistent with $< N_a\prime, P_a\prime >$ and the current set of available resources, $\mathcal{R}$. The achievement of all the goals in $\mathcal{G}_a$ will lead the reduction of $\mathcal{R}$ by *at least* $N_a\prime$, but *no more* than $P_a\prime$. These represent the upper and lower bounds to the resources consumed by the set of goals. It is rational for an agent to ensure that its maintenance goals will still hold after the achievement of its goals, i.e. after $\mathcal{R}$ has been reduced by at least $N_a\prime$, but no more than $P_a\prime$, the maintenance conditions in $\mathcal{G}_m$ still hold.

If a maintenance condition holds after $\mathcal{R}$ is reduced by $P_a\prime$, this indicates that the maintenance condition will still hold no matter which method of achieving the goals is selected by the agent. If the maintenance condition does not hold after $\mathcal{R}$ is reduced by $N_a\prime$, this indicates that the maintenance condition cannot be satisfied, no matter which method of achieving the goals are selected, i.e. the maintenance goal is inconsistent with $\mathcal{G}_a\prime$. A third alternative occurs when neither of the first two cases are met – this identifies a case where $\mathcal{R}$ is insufficient to ensure that $\mathcal{G}_m$ remains consistent after achieving all goals in $\mathcal{G}_a$, yet there are sufficient resources such that the goals may be achieved with the maintenance condition valid. All maintenance goals are required to be checked for consistency in this manner, and the results are aggregated in the following function. A set of maintenance goals that are inconsistent are also produced as a result of this function.

function *check-resource-consistency* ($\mathcal{G}_m$, $< N_A\prime, P_A\prime >$, R)
```
    number of maintenance goals := 0
    number of maintenance goals satisfied := 0
    number of maintenance goals unsatisfied := 0
    CFG := ⟨⟩
    for each maintenance goal mg in G_m
        mc := mg.maintenance condition
        if mc is consistent with R − P_A'
            number of maintenance goals satisfied ++
        if mc is ¬ consistent with R − N_A'
            number of maintenance goals unsatisfied ++
            CFG := ⟨ CFG ∪ mg ⟩
        number of maintenance goals ++
    if number-of-maintenance-goals-satisfied ==
        number-of-maintenance-goals then
        // all maintenance goals are definitely consistent
        return consistent
    else if number of maintenance goals unsatisfied > 0 then
        // some or all maintenance goals are not consistent
        return ⟨ in-consistent, CFG ⟩
    else
        return unknown
```

There are three possible outcomes of checking for consistency using the methods described above:

Case 1 *Consistent with all maintenance goals*

In the case that if after adding G_a
(a) the maximum possible resource requirements ($P_a\prime$) are consistent with the current maintenance goals; and
(b) the potential effects of all goals ($PE_a\prime$) are also consistent with the current maintenance goals,
then it is *safe* to add G_a as a goal to the system.

Case 2 *Not consistent with some or all maintenance goals*

In the case where after adding G_a if either
(a) the necessary resource requirements ($N_a\prime$); or
(b) the definite effects ($DE_a\prime$)
of the new goal set is not consistent with one or more maintenance goals then there will be a definite conflict if G_a is added to the system. To prevent this conflict, we can use the *preventative goal* associated with each maintenance goal as follows:
(1) For each maintenance goal G_m that is in conflict with G_a add the preventative goal of G_m to the system.
(2) Add the goal G_a to the suspended goals list with entry (G_a, {PG_1, ..., PG_i}) where PG_1, ..., PG_i are the preventative goals added in the previous step.

This method would then allow G_a to begin execution only after all the necessary preventative measures are taken.

In the case where there is no preventative goal prescribed for a conflicting maintenance goal then G_a has to wait until that maintenance goal is no longer in the system. We assume that maintenance goals take priority over achievement goals. Naturally, we could enforce any reasonable scheme for resolving such conflicts..

Case 3 *It is uncertain if there will be a conflict*

The *uncertain* case occurs when either:
(a) the resource requirements to be maintained are greater than the necessary resource requirements and less than the possible resource requirements; or
(b) the definite effects of the goal do not violate any maintenance condition but the potential effects do.

In this case we cannot determine for certain if there will be a conflict or not as it depends on the path chosen by the agent to achieve its goals.

In such situations it would appear rational for the agent to pursue G_a without performing additional actions, as there is a possibility that the appropriate path is chosen such that no conflicts occur. However, it may also be rational for an agent to execute some preventative goals that may make (more) certain that the agent's goals will be achieved successfully.

In [13], a process of *dynamic updates* was used to update the resource summaries of goal as the agent committed to certain plans (and sub-goals) at run-time. A more sensible approach for an agent would be to execute G_a when it is uncertain but monitor for situations, by dynamically updating the summary information and checking for inconsistencies, when the status changes to either definitely conflicting (case 2) or safe (case 1) and act accordingly.

There may be times, however, when the agent does not (or cannot) anticipate the future violation of its maintenance goals due to its pursuit of its achievement goals, and in these cases, the maintenance goals could be unsatisfied. The reactive nature of maintenance goals are still supported by our system. Hence, in such cases where the maintenance goal of the agent is spontaneously violated, the agent will trigger its recovery goal, which has the goal of re-establishing the maintenance condition of the goal.

We make a note here that preventative goals will most likely not be used in the case when an achievement goal conflicts with maintenance goals due to the effects that it brings about, rather than its resources. While our framework supports such a mechanism, we could not identify any cases where a preventative goal would be of use in avoiding conflicts over effects.

4.2 Adding Maintenance Goals

In the previous section, we have shown how achievement goals can be added whilst maintaining consistency between existing goals. We now look at the process of adding a maintenance goal to an agent system such that it is consistent with all other goals that the agent is pursuing. Maintenance goals present several challenges that separate them from the addition of achievement goals, as they do not achieve any effects per se but require conditions to be maintained.

The first step in adding a new maintenance goal G_m is to determine if it is consistent with the set of existing maintenance goals $\mathcal{G}_m$. This is done by simply checking the maintenance condition of each maintenance goal in $\mathcal{G}_m$ against the maintenance condition of G_m. If G_m conflicts with any other maintenance goals, then it is not added to the system. If the application demands it, G_m may be placed in the suspended goals list, waiting on the conflicting goals.

If G_m is consistent with $\mathcal{G}_m$ then the next step is to check if it is consistent with the current set of achievement goals $\mathcal{G}_a$. This is done by combining G_m with $\mathcal{G}_m$ and check that no achievements goal will violate the new set of maintenance goals, using the methods described in the previous section.

If there is no conflict then G_m is added to the system. In cases where adding G_m causes conflict, we prevent G_m from being added to the system, as it is difficult to determine the correct subset of achievement goals that are compatible with the set of maintenance goals. There may be a subset of achievement goals from the agent's current goal set that is consistent with the agent's maintenance goals (and current resources), but identifying such a subset can be difficult. It may be feasible, however, if there is a goal preferencing mechanism in place.

In these algorithms, we have assumed that the goal that is being added to the system is preferred less than the existing goals – that is, the agent does not drop goals that it is currently pursuing if a new conflicting goal is added. Further work is required in this area, and in the area of selecting between goals. itemize

5. EXPERIMENTAL EVALUATION

We have performed some preliminary experiments to quantify the benefits of proactive maintenance goals. X-JACK is an extension to JACK Intelligent Agents[3] developed in [14, 13] for evaluating their work. We further extend X-JACK to incorporate reactive and proactive maintenance goals and the algorithms developed in this work.

Our experimental test bed is that of a Mars rover capable of moving about a 2 dimensional environment, consuming energy from its battery as it moves. For simplicity, we assume that each unit of distance drains the battery by 1 unit and that there is a refuelling depot located in the middle of this environment.

A *goal* within the system is a particular location that the agent must visit. To test our system we created a number of goals with randomly generated locations. The agent was then given goals varying from a set of 10 to 100 for simulating an agent's typical workload. Each set was run a 20 times and the results mentioned below are the average of these runs.

Our experiments were conducted in 3 cases as follows:

- No maintenance goal (NMG) - In the first case, we assume that the rover's movements did not consume battery power, therefore, the rover had no need to recharge. This allows us to determine a baseline in our experiments.

- Reactive maintenance goal (RMG) - The second case involves a rover with a maintenance goal that is *reactive* to ensure that there is always enough charge in the battery to be able to reach the recharging station. Therefore, as the rover moves about, it will periodically compare its current distance from the recharge station with the amount of charge remaining in its battery. If there is not enough charge to continue, the rover must first head to the recharge station and recharge, prior to pursuing any other goal.

- Proactive maintenance goal (PMG) - The third case is when the rover has the same maintenance goal as case 2, but it is *proactive* in this case. This requires the

[3]www.agent-software.com.au

Goals	NMG	RMG	PMG
10	539.6667	595	542.3333
20	941.3333	1103.3333	951.3333
30	1098	1287.6667	1150.6667
40	1376.6667	1566.3333	1443.3333
50	1450.3333	1762	1514.3333
60	1713	2056.6667	1886
70	1793.3333	2178.3333	1960.3333
80	2553	3110.3333	2726
90	2606	2849.3333	2745
100	2551.3333	3012	2575

Table 1: Total time (milliseconds) to achieve all goals

Goals	NMG	RMG	PMG
10	308	439	341
20	704	946	797
30	814	1196	1034
40	1284	1851	1587
50	1460	2067	1812
60	1689	2512	2112
70	1881	2619	2350
80	2392	3410	3086
90	2973	4330	3717
100	3124	4660	3925

Table 2: Total distance (units) travelled

agent, when commencing a new goal, to anticipate its resource usage in achieving that goal. If the resources consumed in achieving this goal violate the agent's proactive maintenance goal, a preventative goal will be activated. The agent determines how much battery power will be left after achieving this goal, and if it is an insufficient amount, the recharge goal is executed first. In our experiment, to simulate the possible inaccuracies with estimating resource usage, the rover is able to estimate its resource usage to an accuracy of 10%.

The total time taken for the set of goals to complete in all 3 cases are shown in table 1. As expected, the case where the rover never needs to refuel (NMG) is the fastest. Note that there is little difference here between the reactive and proactive maintenance goals. In some instances the proactice maintenance is actually faster, due to less distance being travelled. As shown in Table 2, the total distance travelled by the rover in each of the cases is less for proactive maintenance than for reactive maintenance.

Tables 3 and 4 provides a breakdown of the rover's movements for the reactive and proactive cases respectively. They show the distance the rover spent successfully pursuing a goal (*total distance*), the distance the rover spent to refuel (*backtrack*), and the distance the rover spent trying to accomplish a goal that failed due to insufficient fuel (*waste*). The *excess* column is the sum of the waste and backtrack distance. Where this value is small indicates that the rover did not spend many resources on movement that was not directly attributable to achieving one of its goals. Therefore, smaller values are better. The comparison of the excess column shows that the excess amount for the proactive case is always less than that for the reactive case. This is because the proactive maintenance goal avoids consuming resources for goals that it can anticipate will fail. The reactive maintenance goal can only determine that the goal will fail once it has already consumed some resources in trying to achieve it.

Goals	Total Distance	Waste	Backtrack	Excess
10	439	120	69	189
20	946	282	151	433
30	1196	237	218	455
40	1851	593	315	908
50	2067	521	352	873
60	2512	650	437	1087
70	2619	558	442	1000
80	3410	831	574	1405
90	4330	1151	781	1932
100	4660	1106	887	1993

Table 3: Distance breakdown for Reactive Maintenance Goals

Goals	Total Distance	Waste	Backtrack	Excess
10	341	0	91	91
20	797	0	259	259
30	1034	0	280	280
40	1587	43	524	567
50	1812	92	510	602
60	2112	127	546	673
70	2350	48	685	733
80	3086	158	887	1045
90	3717	93	1181	1274
100	3925	140	1075	1215

Table 4: Distance breakdown for Proactive Maintenance Goals

If we compare the waste and backtrack values for the reactive and proactive cases, it is observed that often the reactive case has less backtrack than the proactive. This is because in the reactive case, some movement that becomes waste actually brings the rover closer to the refuelling point – hence, reducing the backtrack distance.

We also note that the value of waste for the proactive case is often much less than that for the reactive, because the objective of the proactive maintenance goal is to eliminate waste, by anticipating the failure of its maintenance condition at an earlier point than the reactive goal. For the first goal sets (10,20,30) for the proactive case, this works perfectly. However, some waste still occurs as the number of goals increases. This is because the estimation of resource usage was not 100% accurate. Our results show that even with inaccurate estimation, proactive maintenance goals can offer a reduction in resource consumed.

When resource estimation is perfectly accurate, there would be no waste at all for proactive maintenance, as it will never attempt to achieve a goal unless it is certain that it will be successful. In other words, the only way that waste gets introduced in the proactive case is by the inaccurate estimate of resources.

6. DISCUSSION AND FUTURE WORK

We have seen how the use of proactive maintenance goals can improve the rational behaviour of BDI agent systems. In particular, the ability to anticipate the failure of maintenance goals (rather than merely reacting to their failure) can result in some significant improvements in the decision-making ability of the agent. We see this as the first step towards providing a richer variety of goal types for agent systems, resulting in greater rationality for the agents concerned.

By utilizing existing techniques for reasoning about achievement goals, we are able to smoothly integrate proactive versions of maintenance goals, together with reactive maintenance goals and achievement goals. This provides a richer programming environment which can be readily developed as a plug-in for existing systems such as Jadex.

A line of further work that we intend to pursue is that of passive maintenance goals. A reactive maintenance goal leads to action only when the goal is violated. As we have seen, proactive maintenance goals may also lead to action in anticipation of violation. A passive maintenance goal is one that does not lead to actions per se, but is one that is never allowed to be violated. Such goals act purely as constraints on other actions; for example, if the Mars rover is currently travelling at its maximum speed and there is a passive maintenance goal to this effect, then there is no point in considering any action which will increase the rover's speed.

An extension of this idea is to prioritise maintenance goals, possibly via their urgency. For example, a maintenance goal of keeping the temperature of a gas tank below 100° will becoming increasingly urgent as the temperature climbs above 90°. This kind of rational behaviour is particularly important for safety-critical systems.

The relationship between achievement goals and maintenance goals also raises some questions, such as under what circumstances an achievement goal can trigger a maintenance goal. For example, the Mars rover may have to arrive at a particular location with at least 50% batter power (to perform a particular experiment), and doing so will require it to travel at a lower maximum speed than normal.

A further direction is to explore the ways in which violation of maintenance goals can be predicted. When the goals are about resources, it is inevitable that they will be exhausted; the key issue is to determine the optimal point of replenishement and to see how this can be specified by proactive maintenance goals. Clearly the system's ability to utilize maintenance goals is a direct function of its predictive power.

Acknowledgments

The authors of this paper would like to thank Michael Winikoff for discussions related to this material. The support of the Australian Research Council and Agent Oriented Software through grant LP0453486 is gratefully acknowledged.

7. REFERENCES

[1] L. Braubach, A. Pokahr, D. Moldt, and W. Lamersdorf. Goal Representation for BDI Agent Systems. In *Proceedings of the AAMAS Workshop on Programming in Multi-Agent Systems (PROMAS'04)*, pages 44–65, 2004.

[2] P. Busetta, R. Ronnquist, A. Hodgson, and A. Lucas. Jack Intelligent Agents - Components for Intelligent Agents in Java, 1999.

[3] B. J. Clement and E. H. Durfee. Theory for Coordinating Concurrent Hierarchical Planning Agents Using Summary Information. In *AAAI/IAAI*, pages 495–502, 1999.

[4] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed Requirements Engineering. *Science of Computer Programming*, 20(1-2):3–50, April 1993.

[5] M. Dastani, F. de Boer, F. Dignum, and J. Meyer. Programming Agent Deliberation: An Approach Illustrated Using the 3APL Language. In *Proceedings of the Second International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'03)*, pages 97–104, Melbourne, July 2003. ACM Press.

[6] M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *AAAI*, pages 677–682, 1987.

[7] F. Giunchiglia, J. Mylopoulos, and A. Perini. The Tropos Software Development Methodology: Processes, Models and Diagrams. In *Proceedings of the First International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, pages 35–42. ACM Press, July 2003.

[8] M. J. Huber. JAM: A BDI-theoretic mobile agent architecture. In *Proceedings of The Third International Conference on Autonomous Agents*, pages 236–243, Seattle, WA, 1999.

[9] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: Implementing a BDI-Infrastructure for JADE Agents. *EXP – in search of innovation*, 3(3):76–85, 2003.

[10] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In R. van Hoe, editor, *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Eindhoven, The Netherlands, 1996.

[11] A. S. Rao and M. P. Georgeff. BDI-agents: From Theory to Practice. In *Proceedings of the First International Conference on Multiagent Systems (ICMAS'95)*, San Francisco, 1995.

[12] J. Thangarajah, L. Padgham, and M. Winikoff. Detecting and Avoiding Interference Between Goals in Intelligent Agents. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'03)*, 2003.

[13] J. Thangarajah, L. Padgham, and M. Winikoff. Detecting and Exploiting Positive Goal Interaction in Intelligent Agents. In *Proceedings of the Second International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'03)*, Melbourne, July 2003.

[14] J. Thangarajah, M. Winikoff, L. Padgham, and K. Fischer. Avoiding Resource Conflicts in Intelligent Agents. In *Proceedings of the European Conference on Artificial Intelligence (ECAI'02)*, 2002.

[15] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative & Procedural Goals in Intelligent Agent Systems. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR'02)*, Toulouse, April 2002.

On the relationship between MDPs and the BDI architecture

Gerardo I. Simari
Department of Computer Science
University of Maryland College Park
College Park, MD 20742
gisimari@cs.umd.edu

Simon Parsons
Dept of Computer & Information Science
Brooklyn College, City University of New York
Brooklyn, NY 11210 USA
parsons@sci.brooklyn.cuny.edu

ABSTRACT

In this paper we describe the initial results of an investigation into the relationship between Markov Decision Processes (MDPs) and Belief-Desire-Intention (BDI) architectures. While these approaches look rather different, and have at times been seen as alternatives, we show that they can be related to one another quite easily. In particular, we show how to map intentions in the BDI architecture to policies in an MDP and vice-versa. In both cases, we derive both theoretical and related algorithmic mappings. While the mappings that we obtain are of theoretical rather than practical value, we describe how they can be extended to provide mappings that are useful in practice.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents*.

General Terms

Theory, Design

Keywords

Markov Decision Process, Policy, Intention.

1. INTRODUCTION

Markov decision processes (MDPs) can be regarded as an ideal approach to the implementation of intelligent agents provided that one can either assign utilities to states and probabilities to transitions between states, or that it is feasible to learn such values. Once these values are known, algorithms such as value iteration [10] will yield an MEU-optimal *policy*, that is a mapping from every state into the best action to execute in that state. Such a policy is a complete specification of what the agent should do in *every* state, based on the probable outcomes of *every* action possible in that state.

However, the very nature of these algorithms, and in particular the need to establish the outcome of every possible action in every state, means that finding policies is intractable in many practical cases. This has led to the search for approximate solutions to MDPs [1], but even state of the art approaches like those in [8] are not able to handle particularly large or complex problems.

A contrasting approach to building agents is the use of the BDI architecture [3, 12, 23]. Agents constructed in this way have a set of *belief*s about the state of the world and a set of *desire*s which, broadly speaking, identify those states of the world that the agent has as goals. From its beliefs and desires, and via a process of *deliberation*, and agent formulates one or more *intention*s. The precise semantics of an intention vary across the literature, but intentions can be broadly considered to be states that the agent has committed to bringing about. The agent then constructs a plan to achieve its intentions, typically through some form of *means-ends reasoning*, and executes it.

This is a heuristic approach, and naturally enough agents built using it are outperformed by the those programmed using MDPs when the full MDP solution is tractable [19]. However, the BDI model can scale to handle problems that are beyond the scope of a full MDP solution, and can outperform approximate MDP models [19] for some relatively small problems (such as a simplified Tileworld [9]).

Given this trade-off, we are interested in the formal relationship between the two approaches, and that is our subject in this paper. In particular we investigate the following questions:

1. Given a policy that is a solution to an MDP, how can we extract a BDI description that can be used to approximate the solution to the MDP?
2. Given a complete BDI description, how can we obtain a *policy* that an MDP-based agent can use to control its actions?

We start by briefly summarizing how actions, states, and transition functions for BDI and MDP descriptions can be related. Drawn from [16] this summary considers the formal descriptions provided by the BDI and MDP frameworks. Both descriptions consists of a state space S, a set of actions A, and a a state transition function T which depends on the current state and the action performed. In addition, an MDP description includes:

- a reward function R,
- a probability distribution P over the set of states, and
- a set of policies Π, each member of which identifies the best action to take in each state.

We will denote an MDP description as a tuple

$$\langle S, A, T, R, P, \Pi \rangle$$

An agent that uses an MDP description to decide how to act will be called an MDP agent.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ..$5.00

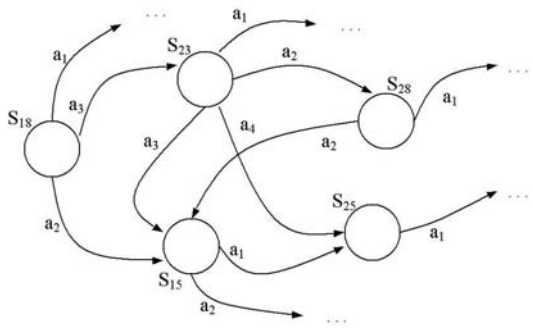


Figure 1: Part of a state-space

A BDI description consists, in addition to the state space, actions, and transition function, of:

- A set current beliefs B, desires D, and intentions I;
- a deliberation component Del; and
- a means-ends reasoning component M

We will denote a BDI description as a tuple:

$$\langle S, A, T, B, D, I, Del, M \rangle$$

An agent that uses a BDI description to decide how to act will be called a BDI agent.

For a given agent in a given environment, it is clear that the S, A and T will be the same for both the BDI and MDP descriptions. Furthermore, we will consider that P, the probability distribution in the MDP description and B in the BDI description do exactly the same thing — they identify which state in S the agent is currently in. While we deal with fully observable worlds, we can imagine both pick a single state (and we will revisit this when we talk about partially observable worlds).

With these equivalences in place, we can see that answering the two questions posed above comes down to relating rewards and policies, on the one hand, to desires, deliberation, means ends reasoning and intentions on the other hand. In fact, since rewards are merely a means of determining policies, and desires a step on the way to determine intentions, these components can be effectively ignored. Indeed, the relation we have to consider in detail is that between policies and intentions.

2. INTENTIONS, PLANS AND POLICIES

As mentioned above, the precise semantics for an intention varies across the literature. An intention is often taken to be "the agent's current focus", that a state that the agents is committed to bring about (for example by [23]). An intention can also be taken (as in PRS [5]), to be a linear plan that an agent has adopted to reach a state that the agent is committed to bring about. Here we will use the term *intention* to denote a state that an agent has committed to bring about, and use the term *intention-plan* (i-plan) to denote a sequence of actions built to reach a specific state, or, in other words, to achieve a specific intention. Since the precise sequence of actions selected will vary depending on what state the agent was in when it deliberated, an i-plan will depend on both state and intention — an agent will, in general, create different i-plans to achieve the same intention from different states, and will, in general, create different i-plans to achieve different intentions from the same state.

The reason that we are interested in i-plans is that they are key in the development of BDI agents. In practical implementations like PRS, once an agent has adopted an intention, it rifles through a pre-compiled set of plans (confusingly also called intentions, and hence i-plans in our terminology) to determine how to achieve the intention it has adopted. Building an efficient BDI system hinges on having a good plan library, and such a library will contain a set of i-plans tailored for the environment in which an agent operates. Thus, in order to answer our motivating questions, we need to relate i-plans to policies. We are interested in both establishing i-plans by solving MDPs, and using a set of i-plans to solve MDPs (hence the two questions we posed in the previous section).

We will adopt the following notation. An i-plan will be denoted by a lowercase Greek letter such as ψ, indexed $\psi^{i,s}$, where necessary, to denote the state s and intention i it relates to. I-plans are sequences of actions and ψ_i denotes the i-th action in ψ, while s_i^ψ denotes the i-th state that the agent plans to visit while executing ψ. Thus, for an i-plan ψ of length p, the agent starts at state s_0^ψ and plans on subsequently visiting states $s_1^\psi, s_2^\psi, ...s_p^\psi$. Of course, the agent may deviate from this sequence of states thanks to nondeterministic actions or changes in the environment. For example, in the case of the state-space in Figure 1, an agent in s_{18} might adopt an i-plan $\psi^{s_{18}}$:

$$s_{18} \xrightarrow{a_3} s_{23} \xrightarrow{a_4} s_{25} \xrightarrow{a_1} s_{79}$$

but find that executing a_4 in s_{23} takes it to s_{28}, where executing a_1 will take it back to s_{18} rather than on to s_{79}.

When such a deviation occurs, then execution of an i-plan will not have exactly its desired effect, though it may still help to bring the agent towards its goal. When the agent realizes that a deviation from the sequence of states laid down in the i-plan has occurred, then it needs to decide if it needs a new intention (which would typically require a new i-plan). This process of *intention reconsideration* [14, 15, 24] identifies whether or not the agent should keep its current intention, or whether it should deliberate to determine a new intention. Clearly if a new intention is adopted, the agent will need to create a new i-plan, and it may need a new i-plan even if it doesn't change intention.

Deliberation, reconsideration and the generation of i-plans can be done decision theoretically. Such approaches to intention reconsideration are explored in [17, 18]. The expected utility of an i-plan can be obtained exactly as one obtains the expected utility of a policy when solving MDP models — by establishing the expected value of executing each action in the state in which it is executed. The difference, however, is that in evaluating an i-plan in this way we only consider a single trajectory through the state-space rather than, as for policy evaluation, consider executing an action in *every* state. Typically, though, this is not the way that deliberation is done. Indeed, the BDI model was developed largely because it was felt [3] that decision-theoretic approaches were intractable (though see [11] for an example of a decision-theoretic approach to deliberation).

Whether reconsideration and deliberation are carried out decision theoretically or not, a BDI agent will have the same approach to traversing a state-space such as that in Figure 1. It will select an intention, identify an i-plan to achieve that intention, and execute that i-plan until it either it realizes that the current i-plan will no longer achieve its intention or until it realizes that the current intention is no longer achievable (or no longer the best possible intention). At such a point the agent will generate a new i-plan, or choose a new intention and generate an i-plan to achieve it, and the process will repeat.

We can contrast this process to that of an MDP agent in the same

state-space, which will always know, from its policy, what is the best action to take. The policy cannot fail to provide a suitable action — it is a universal plan [13] — and will eventually take an agent to its goal. The price paid by the MDP agent is that it has to compute the policy to begin with, a computation that is expensive compared with establishing a simple linear plan. The price paid by the BDI agent is that it has to compute what to do online (while the MDP agent can compute what to do offline) and has to suffer the sub-optimality of likely deviation from its chosen path through the state-space (though the loss of utility can be reduced by effective approaches to intention consideration [18]).

Now, despite the differences between the BDI and MDP models, we can establish a broad equivalence between them by considering how they work on the same state-space. Informally this can be done as follows. Given a policy, we have an action selection for every state, and so can establish one or more i-plans that summaries trajectories through the state-space from a given state to the goal. Thus such a policy incorporates a set of i-plans. Conversely, consider a set of i-plans, each of which captures a trajectory through a set of states. This set of i-plans identifies what action to take in all the states on the trajectory, and if one adds to that the null action for all states not on a trajectory, then one has a policy (albeit not a very good one).

The main contribution of this paper is to make this informal relationship between policies and i-plans more precise. In particular, Section 4 describes how i-plans can be used to create policies, and Section 3 details how policies can be used to derive i-plans.

3. FROM POLICIES TO I-PLANS

Assume we have a policy π that is the solution to a fully specified MDP. For now we assume that π is optimal — the results we obtain depend on this assumption — but we will see that this assumption can be dropped. From any π, it is possible to extract utility values for states that will induce π, and these can be used to establish i-plans.

For the purposes of this section, we shall consider the policy π to be the result obtained by the convergence of an algorithm such as Q-learning [20, 22] which gives a value for every state/action pair. BDI agents can also map states and actions into values. These values can be computed by assigning a value to each i-plan. Let ψ be an i-plan of length p, and ψ_i the i-th action involved in ψ. One way of assigning a value to ψ is:

$$V(\psi) = \sum_{i=1}^{p} \frac{R(s_{i-1}^{\psi}, \psi_i)}{i}$$

where $V(\psi)$ is the value of i-plan ψ, s_o^{ψ} is the initial state of ψ, s_{i+1}^{ψ} is the state to which the agent *expects* to arrive after executing action ψ_i, and $R(s_{i-1}^{\psi}, \psi_i)$ is the reward received for taking action ψ_i in state s_{i-1}^{ψ}. Therefore, the value assigned to an i-plan is the sum of rewards that will be achieved if all of the actions have the desired effect; the division of the reward by i in the above formula captures time discounting — rewards gained early are more valuable than the ones gained in the future.

It is important to note that this is only one of many possible ways of assigning a value to an i-plan. In general, we only require that for an i-plan ψ, every action that is added to the plan has a non-negative cost[1], and that these values depend on the rewards of the states that the agent plans to visit. For example, for a simple environment, the reward function could be defined by:

$$R(s_{i-1}^{\psi}, \psi_i) = \begin{cases} 1 & \text{if } \psi_i \text{ leads directly to a goal} \\ 0 & \text{otherwise.} \end{cases}$$

and $R(s, a) = 0$ for every other state/action pair.

We will now formally define the concepts of *i-plan* and *i-plan length*, and what it means for an i-plan to *obey* a given policy.

DEFINITION 1. *A sequence of actions $\psi = \psi_0, \psi_1, ..., \psi_p$ is called an* i-plan *if the ψ_i's ($0 \leq i \leq p$) were selected with the objective of executing them in turn in order to reach a given goal.*

DEFINITION 2. *Given an i-plan $\psi = \psi_0, \psi_1, ..., \psi_p$, we say that p is the* length *of ψ.*

DEFINITION 3. *An i-plan ψ of length p* obeys *a policy π if, and only if, $\forall i, 1 \leq i \leq p$, $\pi(s_{i-1}^{\psi}) = \psi_i$, where s_i^{ψ} is the state to which the agent is planning on arriving after executing action ψ_{i-1}, and s_0^{ψ} is the initial state.*

Definition 3 simply states that an i-plan obeys a policy if, and only if, the actions prescribed by the i-plan are the same as those prescribed by the policy through the i-plan's intermediate states. Remember that we are assuming that i-plans are *linear* plans, that is no considerations are made for unexpected outcomes of actions (which is typical for BDI implementations).

The dual of the notion of obedience is the notion of conformance:

DEFINITION 4. *A policy π* conforms *to an i-plan ψ of length p if, and only if, $\forall i, 1 \leq i \leq p$, $\pi(s_{i-1}^{\psi}) = \psi_i$ where s_i^{ψ} is the state that results from executing action ψ_{i-1} in state s_{i-1}^{ψ}, and s_0^{ψ} is the state in which the first action is executed.*

Since an i-plan is indexed by the intention that it will achieve when executed, we can extend the notions of obedience and conformance to intentions. A policy π conforms to an intention i if for all i-plans $\psi^{i,s}$, π conforms to $\psi^{i,s}$, and an intention i obeys a policy π if all i-plans $\psi^{i,s}$ obey π.

With the concepts that we have now introduced, we can state the following claim:

CLAIM 1. *Given a* BDI *agent and an* MDP *agent with an optimal policy π, if the* BDI *agent is in state s_i, then the i-plan ψ with the highest utility value will be such that ψ obeys π, starting at s_i.*

The remainder of this section explores this claim.

In general, the claim will only hold if states with the same reward are considered in the same order by both MDP and BDI approaches. Otherwise, even though the utilities are equivalent, the actions might not be exactly the same because the order in which states with the same reward are considered can affect the selection of actions. The proof of the claim can be made with respect to progressively more complex scenarios.

Our initial result relating policies to intentions and i-plans is established for the deterministic, fully accessible case (in other words the simplest):

PROPOSITION 1. *Let $\langle S, A, T, R, P, \Pi \rangle$ be an* MDP *agent, and let $\pi \in \Pi$ be a policy that is optimal under the maximum expected utility criterion. Let $\langle S, A, T, B, D, I, Del, M \rangle$ be a* BDI *agent. Let Del always select the intention with the highest reward and M select the i-plan with the highest reward. If the environment is observable and deterministic, so $\forall s \in S, \forall a \in A, |T(s, a)| = 1$, then $\forall i \in I$, $\forall s \in S$, it holds that $\psi^{i,s}$ obeys π.*

[1] This restriction, basically the same restriction that is required of the path cost function to ensure the optimality of A* search, means that the agent can follow a value gradient to the goal state.

```
Intention policyToIplan(Policy π, MDP m){
    Iplan i;
    s = getCurrentState(m);
    g = getGoalState(s,π,m);
    p = obtainPath(s,g,m);
    while not empty(p) do {
       i.addAction(p);
       p.deleteAction();
    }
    return i;
}
```

Figure 2: Pseudocode for mapping policies into intentions

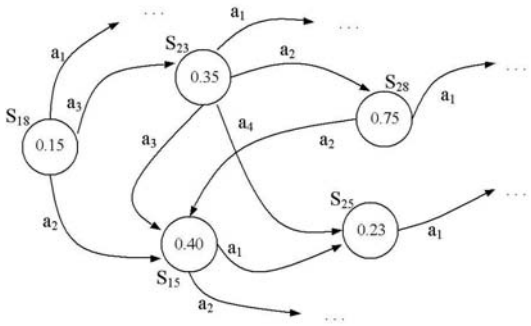


Figure 3: Part of the state space of an MDP

PROOF. *The result follows directly from the fact that we are assuming that D is optimal and that actions are completely deterministic in the environment. Because π is* MEU*-optimal, it will always select actions that take the agent to the best goal state in the best possible way. Because D picks the intention with the highest reward, and actions are deterministic, it will select the same intention/goal that π takes the agent to. Similarly since M picks the i-plan with the highest rewards it will pick an i-plan that traces the same path through the state space (from whichever state the agent is in) as π. Therefore, if we assume that states with equal rewards are considered in the same order by π and M, it is clear that the i-plans generated by M will obey π.* □

If actions are not deterministic, the utility of i-plans is not so clearly defined. Instead of a simple summation of rewards along the path of the plan, the *failure* of actions must be considered. Therefore, we must now assume that the deliberation and means-ends reasoning components are MEU-optimal as opposed to being able to pick the intention and i-plan (respectively) with the highest rewards.

PROPOSITION 2. *Let $\langle S, A, T, R, P, \Pi\rangle$ be an* MDP *agent, and let $\pi \in \Pi$ be a policy that is optimal under the maximum expected utility criterion. Let $\langle S, A, T, B, D, I, Del, M\rangle$ be a* BDI *agent where M and Del are* MEU*-optimal. If the environment is observable and non-deterministic, so $|T(s,a)| \geq 1$, then, then $\forall i \in I$ $\forall s \in S$, it holds that $\psi^{i,s}$ obeys π.*

PROOF. *Because we are now assuming that D is picking* MEU*-optimal intentions and that M is building* MEU*-optimal i-plans, the same argument as in Proposition 1 tell us that every i-plan will obey π even though actions are now non-deterministic.* □

This result also encapsulates the reason why the BDI approach struggles to generate optimal behavior. If we stick with the classic BDI model in which deliberation selects an intention and means-ends analysis then builds a plan to achieve it, in order to pick an optimal set of actions (which is what Proposition 2 is all about), the deliberation component has to be able to pick an intention that is MEU-optimal *before* means-ends analysis picks out a suitable i-plan (and thus before the agent has considered the cost and likelihood of achieving the intention it is selecting).

Now we turn to the case where the environment is not fully observable — in other words the agent doesn't know which state in S it is in, and must rely on its estimates of the current state of the environment [7]. MDP models (technically POMDP models under these conditions) handle this kind of situation by extending the notion of state. Rather than dealing (as we have until now) with a state that is some $s \in S$, the state space describing all the states of the environment, a state becomes a probability distribution across all the s. If we imagine enumerating all these possible distributions s'_i, then we can think of the policies and i-plans as being concerned with a new state space $S' = \cup_i s'_i$. If we call S' the *partially observable counterpart* to S and consider this new state space to be the space in which our BDI and MDP agents operate — so that both B and P identify some $s' \in S'$ as the current state that the agent is in — we can carry forward Proposition 2 as:

PROPOSITION 3. *Let $\langle S, A, T, R, P, \Pi\rangle$ be an* MDP *agent, and let $\pi \in \Pi$ be a policy that is optimal under the maximum expected utility criterion. Let $\langle S, A, T, B, D, I, Del, M\rangle$ be a a* BDI *agent where M and Del are* MEU*-optimal. If the environment is partially-observable, with S' the partially-observable counterpart of S, and non-deterministic, so that $|T(s,a)| \geq 1$, then $\forall i \in I$ $\forall s' \in S'$, it holds that $\psi^{i,s'}$ obeys π.*

PROOF. *This result generalizes the previous one because we are now considering a partially observable environment. However, because we have just expanded S into S' on both the* BDI *and* MDP *sides, the result follows directly from Proposition 2.* □

Thus it turns out that there is a simple formal correspondence between policies and i-plans, similar to the one informally discussed in Section 2. This correspondence holds under rather restrictive assumptions, in particular the optimality requirements, but ensures that the i-plans generated reflect an optimal policy. If we are willing to relax the requirement for a set of i-plans to correspond to an optimal policy, we can create i-plans under fewer restrictions.

Now, these formal results, though they tell us that we *can* extract i-plans that obey a policy don't tell us *how* we might do so. We consider how to do this next. We can obtain an i-plan from any policy in practice by means of a simple search through the state space, following policy π until a local maximum is reached — this state is then selected as the intention. In order to select a unique intention using this process, we assume that the agent's actions always have the most likely outcome; otherwise, a tree would result instead of a simple path. Figure 2 outlines an algorithm `policyToIplan` that can do this in a Java-like pseudocode.

By following policy π in this manner, we are able to obtain as many i-plans as desired: after reaching a intention state, simply continue following the policy from the state that results after achieving the previous intention. For example, in Figure 3 we illustrate this using a fragment of the same state space we considered before. Assume the agent is currently in state S_{18}, $\pi(S_{18}) = a_3$, and $\pi(S_{23}) = a_2$, and that state S_{28} is an intention for the agent. The i-plan that can be extracted in this situation is the linear plan $\langle a_3, a_2\rangle$. In this case, the i-plan that arises is obtained as the result of only one iteration of the process. The same is true in the case of the simplified TileWorld considered in [19], where agents only build plans for reaching one hole, not multiple hole tours.

The process that we have outlined here will construct a set of i-plans that obey an *arbitrary* policy. Such a policy will not necessarily be optimal, and so nothing can be inferred about the optimality or otherwise of the set of i-plans that are established, but the members of the set of i-plans will obey the policy.

With the procedure `policyToIplan`, and the fact that we can determine intentions from i-plans — intentions are, by definition, the states that i-plans lead to — we have answered the first of the questions that we set ourselves at the start of the paper.

4. FROM I-PLANS TO POLICIES

In this section, we will look to answer the second question. We consider how to use a set of i-plans to assign rewards to states in order to provide a policy for the underlying MDP that will mimic the behavior of an agent with the given i-plans. Such an approach makes it possible to use domain knowledge to solve problems that are intractable for existing MDP solution techniques (at the cost of providing sub-optimal solutions) — we use domain knowledge to construct i-plans and then use these i-plans to obtain a policy. Indeed, we can use intentions to construct i-plans and then i-plans to construct policies.

4.1 A single i-plan

We will start off by showing how rewards can be assigned in the state space in order to map *one* i-plan into a policy. Later on, we will see how the process is generalized to an arbitrary set of i-plans.

Assume the agent is currently in state s_a, and has adopted some i-plan ψ of length p. Then, we can assign a value to each state-action pair according to the following formula:

$$val(s^{\psi}_{i-1}, \psi_i) \stackrel{def}{=} i \cdot U(\psi)$$

$\forall i, 1 \leq i \leq p$, s^{ψ}_i is the i-th state involved in ψ, and ψ_i is the i-th action in ψ. In any other case, we have

$$val(s_j, a_k) \stackrel{def}{=} 0$$

$\forall s_j \neq s^{\psi}_i$, for any s^{ψ}_i such that $1 \leq i \leq p$, $\forall a_k$.

These values can be used induce a policy by selecting the action with the highest value given the current state. This process identifies a path through the state space assuming that nothing will go wrong. Such a policy only considers what to do in states that are involved in ψ, and therefore allows for no deviation from the path.

This mapping of states into values is then used as the reward function, which will clearly "mark the path" that the agent must follow in order to reach the goal, and executing the actions along this path will constitute a policy that conforms to the i-plan ψ and hence achieves the intention that ψ was constructed to achieve. In Figure 4, we show this assignment of values to states for the fragment of state space from Figure 3. The agent is initially in state S_{18}, and has formed the intention of reaching state S_{25} by executing actions a_2 and a_1. Therefore, if we assume that the intention has unit value, state S_{18} is given the value 1.0, S_{15} is given 2.0, and the goal is given 3.0; the rest of the states are assigned the value 0.

What we have so far is a mapping that turns an i-plan into its "equivalent" in terms of policies. That is, we have only obtained a correspondence from states to actions similar to the one described in Section 2, which doesn't consider what to do in cases in which the agent *drifts from the path* of its initial i-plan because of the failure of some action to reach the state that was planned for. To obtain a full policy — one that considers *every* possible state — we can extend the partial policy obtained from the i-plan using value iteration to establish a value for every state from the values that have been assigned to states that were part of the intention. The

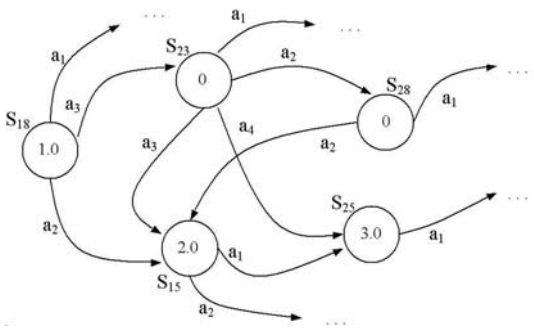


Figure 4: Part of a state space, with values assigned to the states leading to a goal.

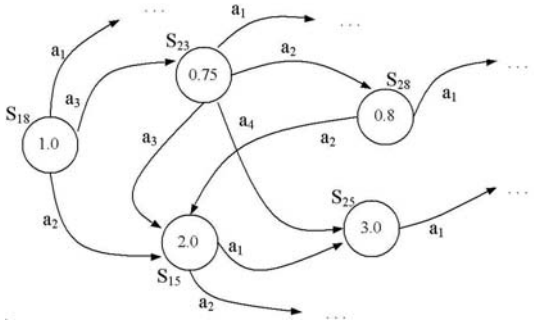


Figure 5: The state space from Figure 4, with values assigned to every state.

example in Figure 5 illustrates how doing this will assign the rest of the states an associated value, with value iteration working in exactly its normal fashion [10], assigning a state a value by suitably discounting the value of states to which it is connected by an action (with the discount determined by the expected cost of the action relating the two).

4.2 Sets of i-plans

So far we have only considered how to build a policy from a single i-plan. In order to use the same technique when an agent has more than one i-plan, only a minor change needs to be made. We must now take into account that the i-plan being considered is not necessarily the first. Therefore, we must keep count of the number of actions involved in the i-plans that have already been processed, which will be represented by the letter κ. When the process starts, we set $\kappa = 1$, indicating that the action to be considered is the first. The function val is now defined as:

$$val(s^{\psi}_{i-1}, \psi_i) \stackrel{def}{=} \kappa \cdot U(\psi)$$

$\forall i, 1 \leq i \leq p$, s_i is the i-th state involved in ψ, and ψ_i is the i-th action in ψ. After each action is considered, the value of κ is increased by 1. After all intentions have been processed, we have

$$val(s_j, a_k) \stackrel{def}{=} 0$$

$\forall s_j \neq s_i$, for any s_i involved in some i-plan, $\forall a_k$. Suppose, for example, that the agent has two i-plans ψ and φ, of lengths p and q, respectively. After processing ψ, κ will have the value $p + 1$ and, after processing φ, its value will be $p + q + 1$.

The process just described is captured by the algorithm `iplanToPolicy` outlined in the form of JAVA-like pseudocode in Fig-

```
Policy iplanToPolicy(IplanSet I, BDI A){
    MDP m; Policy π;
    ValueFunction val; int κ, j = 0;

    val.initialize(0); m.initialize(A);
    orderedIplan = I.obtainOrdering();
    for each i in orderedIplan do {
       j = 0;
       for each action a in i do {
          s = i.obtainInvolvedState(j++);
          val.setState(s,a) = κ*i.getUtil();
          κ++;
       }
    }
    π = valueIteration(m,val);
    return π;
}
```

Figure 6: Pseudocode for mapping i-plans into policies

ure 6. The following proposition formalizes the relationship between a set of i-plan and a policy as established by this algorithm:

PROPOSITION 4. *The algorithm* `iplanToPolicy` *obtains a policy which conforms to the given set of i-plans.*

PROOF. *The algorithm considers each i-plan in the set in turn, assigning a value to each state-action pair that is involved in the i-plan. The value that is assigned is the product of the current i-plan's utility and a monotonically increasing succession of integers. Therefore, each state-action pair will receive a monotonically increasing value with respect to the one before it. Up to this point we have an assignment of monotonically increasing values to the state-action pairs that the given set of intentions dictates, which allows us to obtain a straightforward policy using these values. If we feed the value iteration algorithm with these value assignments, the rest of the state-action pairs (not involved in the set of intentions) will receive values according to this assignment. The value iteration algorithm ensures us that the resulting policy will be* MEU *with respect to the input values.* □

`iplanToPolicy` generalizes the process described above for single i-plans, and to do so requires the i-plans to be ordered — the order in which the i-plans are processed. As we have seen above, i-plans can be assigned utilities in relation with how their execution will reward the agent by reaching certain goals, and this provides a suitable ordering.

Using `iplanToPolicy` to assign rewards to states, we can use value iteration as an *anytime* algorithm to construct a policy. With each iteration, more and more states will receive non-zero values, making the plan more universal. The example in Figure 7 shows how the policy evolves through three iterations of the algorithm on a specific state space.

Of course value iteration always works like this — with values backing-up across the state space from states with rewards — but it is not usually considered an anytime algorithm. That is because in early iterations most states have zero values, and so no useful policy can be extracted. However, in this case, an initial useful policy is provided by the values established by an i-plan. That policy can be used "as is". The worst that will happen is that whenever an agent strays off the "path" marked by states with non-zero values, it will try immediately to get "back on" that path by the shortest route rather than taking the optimal route that would be visible if all states had values. As more states have values filled in by value

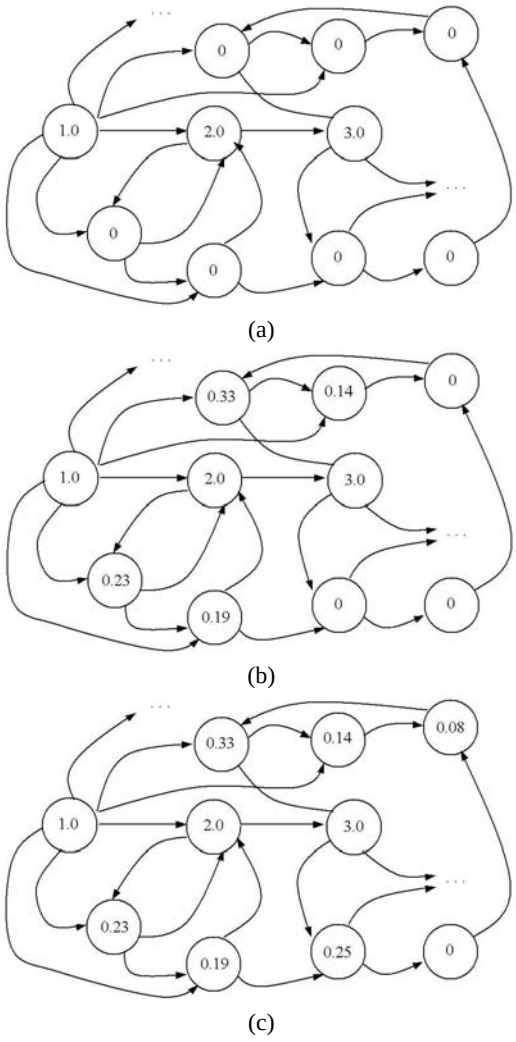


Figure 7: State values after the (a) first, (b) second, and (c) third iterations

iteration, the less likely this is to happen, and thus the better the policy that is computed.

4.3 Practical reasoning and policies

Up to now, we have only been considering the problem of how to map a single i-plan or a group of i-plans into a policy. Even though we have established a relationship between the most important components of the BDI and MDP models, the mapping is not yet complete because a policy obtained using the methods developed in the previous section corresponds to a "snapshot" of the BDI agent's state. If the agent switches intention (for example, due to reconsideration performed in order to profit from a change in the environment), the policy might no longer be valid.

It is possible to map the *complete* set of possible i-plans (all the i-plans for all its possible intentions) into a single policy by considering what intention, and hence what i-plans, the BDI agent would adopt in every possible state. If deliberation is performed in *every* state, followed by means-ends reasoning, a policy can be built by assigning actions to states corresponding to what the BDI planner dictates. It is clear that this process need not be initiated in every state in practice — if a state already has an action assigned to it

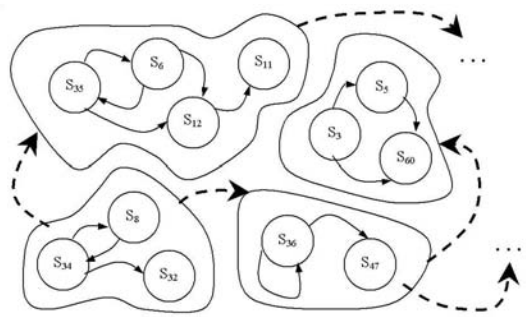


Figure 8: Grouping of states into "envelopes".

(because a previously built plan runs through it), then there is no sense in performing the same computations again.

The fact that we can do this, and that we can establish a set of i-plans from a set of intentions, gives us an answer to the second question we posed at the start of the paper.

It is interesting to point out that such a policy can be built in this way without considering rewards. Rewards played a fundamental role in previous sections because they were the basis for the construction of *contingencies* — once the main plans were laid, the actions performed in the rest of the states were based on these rewards. However, because we are now considering how the BDI agent behaves in every state, such an assignment of rewards is no longer necessary.

We should also point out that this mapping is initially of theoretical interest only, and its sole purpose is to show that a relationship exists between the universal behavior of a BDI agent and a policy for an MDP agent. The computational costs of constructing such a policy would be just as high as (or higher than) the actual process of solving the MDP directly.

5. TOWARDS HIERARCHICAL POLICIES

The analysis in the previous section established a relationship between BDI and MDP models based on the existence in the BDI agent's plan library of i-plans that lay out a precise sequence of actions that the agent has to carry out. It turns out that the i-plans that a BDI agent are typically equipped with aren't quite so detailed.

What happens in a typical BDI implementation is that the deliberation process decides *what* is to be achieved — what intention to adopt. Once a decision is made, the means-ends reasoning process then decides *how* this will be achieved — what i-plan will be used — picking from a set of relatively abstract i-plans and instantiating it to fit the specific situation. The result is to effectively groups states into "envelopes" which contain sets of states that are equivalent as far as the current intention of the agent is concerned but which might differ in detail. For example, considering Figure 8, imagine that each state represents an agent location. The agent is currently at s_{34} and needs to get to s_{60}, and its environment is split into a series of rooms that group states together — s_{34} with s_8 and s_{32}, all in the same room but at different locations within the room, say — and so on. Now the important thing as far as the agent is concerned, while navigating around, is which room it is in, so it can construct an i-plan at the "room level", figuring that it needs to get to the $\langle s_{36}, s_{47} \rangle$ room first, then to the $\langle s_3, s_5, s_{60} \rangle$ room.

The idea of this kind of state aggregation is not a novel one. In fact, much work has been dedicated to the grouping of states in MDPs and POMDPs, as can be seen in [2, 4, 6, 21], with the aim of improving the tractability of solving problems with ever larger state-spaces. However, it does tie in very neatly with the BDI model, and we can easily see that it will be possible to establish similar results to those we have obtained above, but where the correspondence is not between whole-environment policies and i-plans at the level of individual states, but between i-plans and policies at the state-aggregation level.

Now, being able to construct policies at the state-aggregation level is a useful thing to do. Open questions in work on state-aggregation are how to aggregate states, what level states should be aggregated at, and which states should be grouped with which states. The methods we have developed here potentially provide answers — i-plans identify what states to group together, and these can then be turned into high-level policies for a partial MDP solution that can then be refined, filling in policies that tell the agent how to act within grouped states (which is a smaller, and hence more tractable problem than determining a whole-world policy).

6. DISCUSSION

The BDI model was established at least in part [3] because it was felt that decision-theoretic models were too computationally intractable to use in practice. This has led to a perception that the two models are somehow cast in opposition with one another, that one adopts the BDI model only if one believes that decision theoretic models are somehow wrong or impractical, and that to use the BDI is to somehow settle for less than perfect performance (it is, after all, a heuristic approach). Our aim in this work is to steer a middle course.

We have previously shown [19] that on a particular task there are cases where using an MDP model provides the best solution, while in other cases an BDI approach works best — as the task grows in size beyond the range of problems that can be optimally solved using an MDP, the BDI approach outperforms the best one can do with an MDP. This suggests that some aspects of the BDI approach may be worth considering in more detail. Here we provide some of that detail.

The results from Section 3 show that the BDI model is not inherently sub-optimal. If we can, as the results there show we can in theory, construct a set of intentions that obey an optimal policy, then the BDI model will give us optimal performance. That it doesn't in practice, is because intentions are constructed not from optimal policies but from domain knowledge, and it is because of the mismatch of the two that sub-optimality creeps in.

The results in Section 3 don't help us in practice since there is no need to build intentions from an optimal policy — if we have an optimal policy we can just execute it directly. However, the results in section Section 4 can help us in practice. They show us how to construct policies from i-plans, and we can (as in [19]) create i-plans in situations where the state-space is so large that it isn't possible to generate anything like an optimal policy from first principles.

Finally, the discussion in Section 5 suggests that we can use the relationships we have developed to construct policies at a sufficiently abstract level that we can gain some computational advantage over directly solving an MDP. Of course, so far we have only demonstrated that it is possible to do this in theory. Some practical demonstration will be necessary to be completely convincing, and providing empirical results that do demonstrate this, using the same Tileworld testbed as we used in [19], is what we are currently working towards.

7. SUMMARY

In this paper, we have presented a series of relationships that exist between certain components of the BDI model and those of

MDPs. We have extended the work of [16] by exploring how *i-plans* on the BDI side and *policies* on the MDP side can be related to one another. To our knowledge, this is the first time that such a relationship has been proposed.

We have mapped from policies to i-plans by proving that the i-plans derived from an optimal policy are those adopted by a BDI agent which selects i-plans with the highest utility, and an optimal reconsideration strategy; furthermore, we presented a computational version of this mapping, by means of a search algorithm based on following the given policy through the state space. The other part of the mapping, obtaining policies from i-plans and thus intentions, was also characterized both theoretically and algorithmically. First, we established the mapping declaratively, by assigning rewards to states in such a way as to induce the selection of actions in the same way as they are selected by the intentions. This allows the derivation of a policy by means of any of the MDP solution algorithms. A simple algorithmic version of this mapping was also presented, indicating that one possible solution is a policy that simply follows the actions as dictated by the i-plan, and says nothing about the rest of the possible states. The use of value iteration as an anytime algorithm which refines and completes the policy with each i-plan was also proposed as an alternative way of deriving a policy.

This work constitutes the first steps in a line of research that aims to unify the two approaches. In the future we will test these findings empirically in pursue of the development of agent design methodologies that make use of the best of both the BDI and MDP models.

8. ACKNOWLEDGMENTS

The work described in this paper was partially supported by a grant from the City University of New York PSC-CUNY Research Awards Program and partially supported by the Air Force Office of Scientific Research under Grant Nr. FA95500510298, by Army Research Office grant number DAAD190310202, by NSF grants IIS0329851 and 0205489, and by the Joint Institute for Knowledge Discovery

9. REFERENCES

[1] D. Aberdeen. A (revised) survey of approximate methods for solving partially observable markov decision processes. Technical report, National ICT Australia, Canberra, Austalia, 2003.

[2] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1–2):49–107, 2000.

[3] M. E. Bratman, D. Israel, and M. Pollack. Plans and resource-bounded practical reasoning. In R. Cummins and J. L. Pollock, editors, *Philosophy and AI: Essays at the Interface*, pages 1–22. The MIT Press, Cambridge, Massachusetts, 1991.

[4] T. Dean, R. Givan, and S. Leach. Model reduction techniques for computing approximately optimal solutions for Markov decision processes. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, pages 124–131, 1997.

[5] M. P. Georgeff and F. F. Ingrand. Decision-making in embedded reasoning systems. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, pages 972–978, 1989.

[6] J. Goldsmith and R. H. Sloan. The complexity of model aggregation. In *Artificial Intelligence Planning Systems*, pages 122–129, 2000.

[7] W. S. Lovejoy. A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 28(1–4):47–66, Apr. 1991.

[8] J. Pineau, G. Gordon, and S. Thrun. Policy-contingent abstraction for robust robot control. In *Proceedings of the Conference on Uncertainty in AI*, Acapulco, Mexico, 2003.

[9] M. Pollack and M. Ringuette. Introducing the Tileworld: experimentally evaluating agent architectures. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 183–189, 1990.

[10] M. L. Puterman. *Markov decision processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, Inc., New York, 1994.

[11] A. S. Rao and M. P. Georgeff. Deliberation and its role in the formation of intentions. In *Proceedings of the 7th Annual Conference on Uncertainty in Artificial Intelligence*, pages 300–307, 1991.

[12] A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First International Conference on Multiagent Systems*, 1995.

[13] M. J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 1039–1046, 1987.

[14] M. Schut and M. Wooldridge. Intention reconsideration in complex environments. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 209–216, 2000.

[15] M. Schut and M. Wooldridge. Principles of intention reconsideration. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 340–347, 2001.

[16] M. Schut, M. Wooldridge, and S. Parsons. On partially observable MDPs and BDI models. *Lecture Notes in Computer Science*, 2403:243–??, 2002.

[17] M. Schut, M. Wooldridge, and S. Parsons. The theory and practice of intention reconsideration. *Journal of Theoretical and Experimental AI*, 16(4):261–293, 2004.

[18] M. C. Schut. *Intention Reconsideration*. PhD thesis, University of Liverpool, 2002.

[19] G. I. Simari and S. Parsons. On approximating the best decision for and autonomous agent. In *Proceedings of the Sixth Workshop on Game Theoretic and Decision Theoretic Agents*, pages 91–100, 2004.

[20] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA, 1998.

[21] J. Tsitsiklis and B. van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1/2/3):59–94, 1996.

[22] C. J. C. H. Watkins. *Learning with delayed rewards*. PhD thesis, Cambridge University, 1989.

[23] M. Wooldridge. Intelligent Agents. In G. Weiss, editor, *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*, chapter 1, pages 27–78. The MIT Press, Cambridge, Massachussetts, 1999.

[24] M. Wooldridge and S. Parsons. Intention reconsideration reconsidered. In J. Müller, M. P. Singh, and A. S. Rao, editors, *Agent Theories, Architectures, and Languages V*, pages 63–80. Springer-Verlag: Heidelberg, Germany, July 1999.

Modelling Situations in Intelligent Agents *

John Thangarajah
RMIT University
Melbourne, Australia
johthan@cs.rmit.edu.au

Lin Padgham
RMIT University
Melbourne, Australia
linpa@cs.rmit.edu.au

Sebastian Sardina
RMIT University
Melbourne, Australia
ssardina@cs.rmit.edu.au

General Terms

Design, Theory

Keywords

Agent and multi-agent architectures; Frameworks, infrastructures and environments for agent systems

1. INTRODUCTION AND MOTIVATION

BDI agent systems and languages such as PRS, JAM, JACK, 3APL, and AGENTSPEAK have been widely used in developing robust and flexible applications in dynamic domains. However, one criticism of these systems is that the modelling of how agent reasoning progresses is too reliant on the rather low level notion of individual *events*. In our own work in a number of application areas, we have consistently noticed a need for a more abstract concept, which we call a *situation*. Recognition by the agent that it is in a particular situation may affect the goals that it has, may place overarching constraints on how it operates, or may influence the way that it chooses to achieve its goals.

Work on "situation awareness" [1] also supports the position that aggregation of lower level perceptions into meaningful chunks allowing for understanding is a crucial part of intelligent agent systems. Work on "situation spaces" [2] collects together information about the environment into a set of meaningful high-level abstractions that provide the context for the agent's cognitive activity (in this case planning). Situation awareness has also more recently been proposed as a means of interpreting pro-active behaviour in agents [3].

While situations can in a sense be captured simply as a collection of beliefs, there is an advantage in introducing them as a conceptual entity which can be modelled declaratively.[1] Firstly, this allows for developing a semantics that directly captures the meaning of situations within the agent architecture (for example, we shall require that the reasoning about situations takes priority over all other agent reasoning.) Secondly, it allows for domain experts to more readily specify the characteristics of the situations particular to the domain. Thirdly, it can allow for general purpose infrastructure which supports the recognition of situations and the appropriate adjustment of agent attitudes.

As an initial step in capturing and reasoning about situations, we provide an operational semantics describing the agent's reasoning when it recognises that a particular situation exists, or ceases to exist. We do this within the context of the previously presented BDI agent language CAN (**C**onceptual **A**gent **N**otation) [4], which is similar in style to AGENTSPEAK, 3APL, and other BDI languages. The distinguishing feature of CAN is its goal construct and its account of failure handling typically found in implemented BDI systems, which provides goal persistence.

Although our initial work is assuming situations will be recognised by individual agents, the work is generalisable to multi-agent systems also.

2. SITUATION RULES

We define the rules for managing situations as an addition to the CAN language defined in [4]. As previously indicated, situations are (conceptually) at a coarser level of granularity than individual events, and the agent's reaction to a situation may involve changing its mental attitudes and focus of attention, rather than directly producing behaviour. Consequently, we introduce the notion of a *meta* program/plan (MP) whose objective is not to produce behaviour (directly), but to change such things as attitude, focus, understanding, and priorities of an agent. It is possible to conceive a range of things a meta plan could do, such as change the agent's goals; drop active intentions; change the plan library; start protecting certain conditions; or engage in specialised reasoning such as diagnosis or prediction. For the purpose of the current work, though, we shall restrict meta plans to add goals and to set and release system constraints.

In the CAN language, an *agent configuration* is a tuple $\langle \mathcal{N}, \Pi, \mathcal{B}, \mathcal{G}, \Gamma \rangle$ where $\mathcal{N}$ is the agent name, Π is a plan library, $\mathcal{B}$ is a belief base, $\mathcal{G}$ is a goal base, and Γ is the

*We thank Bao Quoc Vo, within the RMIT agents group, for contributions to this work. We also gratefully acknowledge the support of the Australian Research Council, The Bureau of Meteorology and AOS (especially Ralph Rönnquist), under Grant LP0347025 and Grant LP0453486.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

[1] We note that plan types in the BDI plan library can also be modelled simply as beliefs, However, it is well accepted that there is both a computational and semantic advantage in modelling plans as special kinds of beliefs. We claim the same is true of situations.

set of current intentions. The belief base of an agent often contains ground belief atoms in the form of first-order relations. However, CAN makes no commitments about the form of the belief base; sophisticated logics and reasoning about action formalisms can be used to model more expressive belief bases. We simply require that well-defined operations exist to check whether a condition follows from a belief set ($\mathcal{B} \models \phi$), to add a belief b to a belief set ($\mathcal{B} \cup \{b\}$), and to delete a belief b from a belief set ($\mathcal{B} \setminus \{b\}$). Observe that, in order to capture interesting situations, a temporal formalism for beliefs is likely to be desirable. We will discuss this below when talking about the implementation.

Next, we introduce a *constraint* construct, $\mathsf{Constraint}(\alpha, P)$, with the intended meaning that no action is to be performed if it would make the agent not believe α. If at some point α does not hold, initially or due to some factor beyond the agent's control (e.g., other agent's actions), then plan P ought to be executed. Program P is intended to encode a recovery procedure to re-establish α. We denote the set of constraints in the system as $\mathcal{C}$.

In the CAN language, actions are primitives that always succeed. In order to reason about constraints we require this notion of actions to be extended, as we want to restrict an agent from performing any action that violates any system constraint. To that end, we assume that agents are equipped with a STRIPS-like *action description library* Λ containing rules of the form: $act : \psi_{act} \leftarrow \overset{-}{}_{act}; \overset{+}{}_{act}$, one for each action type in the domain, where ψ_{act} is the pre-condition of the action, and $\overset{+}{}_{act}$ and $\overset{-}{}_{act}$ are the add and delete lists of atoms, respectively. Figure 1 shows the derivation rules for actions. Note that rule *act* ensures that the agent only performs an action if the system constraints are not violated. If the action does violate a system constraint, then the action does not fail but is not allowed to execute. This means that the action in question would be re-considered later on. Hence, actions only *fail* if the pre-conditions are not met.

So, we next define a *meta plan* MP as follows:

$$MP ::= +\mathsf{Goal}(\phi_s, P, \phi_f) \mid +\mathsf{Constraint}(\alpha, P) \mid \\ -\mathsf{Constraint}(\alpha, P) \mid MP_1; MP_2 \mid true$$

where P is a CAN plan-body and MP_1, MP_2 are meta plans.

A *situation library* Π_{Sit} contains all the situations the agent knows of. Concretely, Π_{Sit} includes *situation tuples* of the form $\langle N, \phi_a, MP_a, \phi_e, MP_e \rangle$, where N is the name of the situation, ϕ_a and ϕ_e are *entry* and *exit conditions* respectively, and MP_a and MP_e are the meta plans that the agent executes on situation entry and exit.

We extend the notion of an agent configuration to be $\langle \mathcal{N}, \Pi, \Lambda, \Pi_{Sit}, \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{S}, \Gamma \rangle$, where Π_{Sit} is a situation library, $\mathcal{C}$ is the set of current system constraints, and $\mathcal{S}$ is the set of situations that the agent currently believes to be active. Each entry in $\mathcal{S}$ is the name of the situation and the object bindings that enabled the entry condition of the situation to be true. Figure 2 depicts the derivation rules for capturing when a particular situation becomes active and when a situation ends (Sit_{active} and Sit_{end}), as well as for handling meta-plans. Rule Sit_{active} states that for any situation $\langle N, \phi_a, MP_a, \phi_e, MP_e \rangle$, the situation becomes active when: (a) ϕ_a is true with the bindings ; and (b) the situation in question is not already active (that is, N is not already in $\mathcal{S}$.[2] When a new situation becomes active, then the agent: (a) creates an entry N in the active situations set; and (b) executes the steps of the meta plan MP_a with the corresponding bindings to ensure that the plan acts upon the same object instances that formed the situation.

The Sit_{end} rule states that for any situation that is currently active, if the exit condition is true, then the agent: (a) removes the relevant entry from the active situations set; and (b) executes the exit meta plan with the object bindings that enabled the exit condition to hold. We shall require that the Sit_{active} and Sit_{end} rules take precedence over all other rules of the agent, i.e., they must be applied whenever applicable in every BDI execution cycle.

3. REPRESENTING AND MONITORING

We have indicated that situations will be characterised by some sort of belief-set formulae indicating entry to and exit from a situation, along with meta plans which will be executed on entry and exit. Situations often include a recognition of trends and changes that have occurred over a period of time such as "the thunderstorm's intensity has been increasing steadily in the last 30 minutes" or "the thunderstorm has been moving towards the fire." In order to support this kind of reasoning, we choose to maintain a temporal database. The challenge then is to know what should be stored in this temporal database in order to support the relevant belief-set queries, and when we should query this database.

Noting that the temporal queries we encounter generally have to do with *appearances* of objects over time, we choose to explicitly identify those objects which are involved in temporal relations in the recognition of the entry or exit condition of some situation. We refer to this set of objects as the *core* of a situation as it contains the core objects which require monitoring over time for situation recognition. This identification enables us to maintain historical information regarding only the relevant object types, and to query the temporal database at selected points to recognise entry to and exit from a situation.

The entry condition for the situation is then specified as a *temporal entry formula* (TEF) and a *current entry formula* (CEF). If the TEF is true with some bindings, the CEF will then also be checked with these same bindings. These are conceptually part of the same formula, but are separated out to facilitate using the temporal database only where necessary, if that is desired. The exit condition for a situation is specified in a similar manner. Along with the entry and exit conditions, we specify the meta-plans that ought to be executed when these conditions hold.

We also introduce the notion of a set of *active situations*, which are those situation instances where the entry condition has been true, and the exit condition has not yet become true. Situation instances are added to and removed from this set dynamically. This set is important to ensure that the agent does not repeatedly act upon the same situation instance, and to keep track of which situation instances the end condition should be monitored for.

During execution we need to ensure that relevant temporal (and other) information is stored, while also ensuring that there is not a blow-out in space consumption. We then need to use this information to recognise when a situation starts and ends. We chose to use a temporal database

[2]This is necessary to avoid the agent repeatedly reacting to the very same situation.

$$\frac{a: \quad \leftarrow \quad {}^{+}; \quad {}^{-} \in \Lambda \quad a\theta = act \quad \mathcal{B} \models \quad \theta \quad \forall \mathsf{Constraint}(\alpha, P) \in \mathcal{C} : (\mathcal{B} \setminus \quad {}^{-}\theta) \cup \quad {}^{+}\theta \models \alpha}{\langle \mathcal{B}, \mathcal{G}, \mathcal{C}, act\rangle \longrightarrow \langle (\mathcal{B} \setminus \quad {}^{-}\theta) \cup \quad {}^{+}\theta, \mathcal{G}, \mathcal{C}, true\rangle}\ act$$

$$\frac{a: \quad \leftarrow \quad {}^{+}; \quad {}^{-} \in \Lambda \quad a\theta = act \quad \mathcal{B} \not\models \quad \theta}{\langle \mathcal{B}, \mathcal{G}, \mathcal{C}, act\rangle \longrightarrow \langle \mathcal{B}, \mathcal{G}, \mathcal{C}, fail\rangle}\ act_f$$

Figure 1: Derivation rules for actions over basic configurations.

$$\frac{\langle N, \phi_a, MP_a, \phi_e, MP_e\rangle \in \Pi_{Sit} \quad \mathcal{B} \models \phi_a\theta \quad (N, \theta) \notin \mathcal{S} \quad \langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \Gamma, MP_a\theta\rangle \stackrel{*}{\Longrightarrow}_{Sit} \langle \mathcal{B}', \mathcal{G}', \mathcal{C}', \Gamma', true\rangle}{\langle \Pi_{Sit}, \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{S}, \Gamma\rangle \longrightarrow \langle \Pi_{Sit}, \mathcal{B}', \mathcal{G}', \mathcal{C}', \mathcal{S} \cup \{N, \theta\}, \Gamma'\rangle}\ Sit_{active}$$

$$\frac{\langle N, \phi_a, MP_a, \phi_e, MP_e\rangle \in \Pi_{Sit} \quad \mathcal{B} \models \phi_e\theta\theta' \quad (N, \theta) \in \mathcal{S} \quad \langle \mathcal{B}, \mathcal{G}, \Gamma, \mathcal{C}, MP_e\theta\theta'\rangle \stackrel{*}{\Longrightarrow}_{Sit} \langle \mathcal{B}', \mathcal{G}', \Gamma', \mathcal{C}', true\rangle}{\langle \Pi_{Sit}, \mathcal{B}, \mathcal{G}, \mathcal{C}, \mathcal{S}, \Gamma\rangle \longrightarrow \langle \Pi_{Sit}, \mathcal{B}', \mathcal{G}', \mathcal{C}', \mathcal{S} \setminus \{N, \theta\}, \Gamma'\rangle}\ Sit_{end}$$

$$\frac{\langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \Gamma, MP_1\rangle \Longrightarrow_{Sit} \langle \mathcal{B}', \mathcal{G}', \mathcal{C}', \Gamma', MP_1'\rangle}{\langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \Gamma, (MP_1; MP_2)\rangle \Longrightarrow_{Sit} \langle \mathcal{B}'', \mathcal{G}'', \mathcal{C}'', \Gamma'', (MP_1'; MP_2)\rangle}\ ;_1 \qquad \frac{}{\langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \Gamma, (true; MP_2)\rangle \Longrightarrow_{Sit} \langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \Gamma, MP_2\rangle}\ ;_2$$

$$\frac{}{\langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \Gamma, +\mathsf{Constraint}(\alpha, P)\rangle \Longrightarrow_{Sit} \langle \mathcal{B}, \mathcal{G}, \mathcal{C} \cup \{\mathsf{Constraint}(\alpha, P)\}, \Gamma, true\rangle}\ +\mathsf{Constraint}$$

$$\frac{}{\langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \Gamma, -\mathsf{Constraint}(\alpha, P)\rangle \Longrightarrow_{Sit} \langle \mathcal{B}, \mathcal{G}, \mathcal{C} \setminus \{\mathsf{Constraint}(\alpha, P)\}, \Gamma, true\rangle}\ -\mathsf{Constraint}$$

$$\frac{}{\langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \Gamma, +\mathsf{Goal}(\phi_s, P, \phi_f)\rangle \Longrightarrow_{Sit} \langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \Gamma \cup \{\mathsf{Goal}(\phi_s, P, \phi_f)\}, true\rangle}\ +\mathsf{Goal}$$

Figure 2: Rules for situations and meta plans. $\Longrightarrow_{Sit}$ is a transition relation between situation configurations of the form $\langle \mathcal{B}, \mathcal{G}, \mathcal{C}, \Gamma, MP\rangle$ ($\stackrel{*}{\Longrightarrow}_{Sit}$ stands for the usual reflexive transitive closure of $\Longrightarrow_{Sit}$.)

to store object information, as they provide an infrastructure for storing and retrieving objects with temporal data. The particular temporal database system that we are currently investigating is RapidBase[3] [5]. There are a number of reasons for this choice, some of which we outline below. The RapidBase Query Language (RQL) is based on SQL, which allows programmers with sufficient SQL skills to be able to write RQL queries. RapidBase allows tables to be defined with history columns, such that only histories of those columns will be maintained. This helps in avoiding redundant data and keeping the size of the database to a minimum.

We use RapidBase to store and track objects over time. For efficiency reasons, we only record objects whose types are part of the *core* of some situation known to the agent. Objects are stored by creating a table for each object type and specifying the attributes that require histories to be stored, as history columns.

When specifying a known situation, entry and exit conditions are expressed as RQL temporal queries. If a query returns an empty result set, then the condition in question is false; otherwise, when the result set is not empty, the answer set will contain the bindings for the objects for which the corresponding condition is true. These bindings are necessary for the entry condition, because, as mentioned previously: (a) the same bindings must be used to check the exit condition for that situation instance; and (b) the meta-plan that executes on activation should also use the same bindings as they should reason about the same object instances that formed the situation. The manner in which the object bindings are obtained is another useful and convenient feature of using a temporal database.

4. DISCUSSION AND CONCLUSION

We have introduced a notion of a situation as a special type of entity of which agents should be aware. We have added some rules to a previously developed agent specification language which provide for some initial aspects of responding to situations. We have also presented a representation for situations at the application programming level using a temporal database to capture the important temporal comparisons that commonly occur. We have currently only addressed a small number of things to be done upon recognition of a situation. This is an area where future work is required.

5. REFERENCES

[1] M. R. Endsley. Toward a theory of situation awareness in dynamic systems. *Human Factors*, 37(1):32–64, 1995.

[2] C. F. Schmidt, J. L. Goodson, S. C. Marsella, and J. L. Bresina. Reactive planning using a "situation space". In *Proceedings of the Annual AI Systems in Government Conference*, pages 50–55, March 1989.

[3] Raymond So and Liz Sonenberg. Situation awareness in intelligent agents: Foundations for a theory of proactive agent behavior. In *IEEE/WIC/ACM IAT*, 2004.

[4] Michael Winikoff, Lin Padgham, James Harland, and John Thangarajah. Declarative & procedural goals in intelligent agent systems. In *Proceedings of KR*, 2002.

[5] A. Wolski, J. Kuha, T. Luukkanen, and A. Pesonen. Design of RapidBase - an Active Measurement Database System. In *Proceedings of IDEAS 2000*, pages 75–82. IEEE Computer Society Press, 2000.

[3] `http://www.vtt.fi/tte/projects/rapid/`

Beliefs, Desires and Intentions Through the Environment

Tom Holvoet
AgentWise, DistriNet, Dept. Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven, Belgium
Tom.Holvoet@cs.kuleuven.be

Paul Valckenaers
PMA, Dept. Mechanics
Katholieke Universiteit Leuven
Celestijnenlaan 300B
3001 Leuven, Belgium
Paul.Valckenaers@mech.kuleuven.be

ABSTRACT

BDI-based agent architectures have proven their usefulness in building MASs for complex systems - their explicit attention for coping with dynamic environments is one obvious explanation for this. One large and quite interesting family of MAS applications is characterized (1) by their large scale in terms of number of agents and physical distribution, (2) by their very dynamic nature and (3) by their complex functional and non-functional requirements. This family includes a.o. manufacturing control, traffic control and web service coordination. In this paper, we present an innovative approach to BDI agents which alleviates agent complexity through "delegate MASs". Delegate MASs consist of light-weight agents, which are issued either by resources for building and maintaining information on the environment, or by task agents in order to explore the options on behalf of the agents and to coordinate their intentions.

1. INTRODUCTION

The term "coordination and control applications" can be coined to refer to a large family of applications which share a number of characteristics. First, in the applications, one can distinguish (1) an underlying physical or software environment, and (2) a software system that is connected to this environment. The underlying environment contains fixed entities or "resources" capable of performing particular operations, as well as mobile entities which can move in the environment. Second, the coordination and control software system is able to observe and direct the entities in the underlying environment. Third, the purpose of the application is to execute "tasks". Executing a task requires moving through the environment and performing operations by resources. The purpose of the software system is to manage the underlying environment by controlling entities that live in the environment, and coordinating the collective behavior of these entities. Fourth, the underlying system evolves several orders of magnitude slower than the coordination and control software. This allows the software to observe the environment and to plan ahead. Fifth, the environment itself is highly dynamic. Resources may crash, new resources may be added, connections between resources may be added, lost, or their characteristics (e.g. throughput, speed) may change. Members of this family of coordination and control applications include manufacturing control, traffic control and web service coordination. As an example, in manufacturing control, the environment is the physical world equipped with resources such as machines and conveyor belts, and the tasks are the client orders for fabricating particular products. The software in manufacturing control is responsible for controlling the resources and for guiding the orders through the factory floor. Both the mobile entities (partially fabricated products, vehicles, client software) and the fixed entities or resources in the environment (machines or conveyor belts, roads and intersections, web services) are obvious candidates to be represented as agents - which we call task agents and resource agents respectively.

In this paper, we propose a particular BDI-based approach [1, 4] that aims to avoid part of the internal agent complexity by exploiting the environment [6]. In the approach, so-called "delegate MASs" are issued by the different agents. Delegate MASs consist of light-weight agents (called ant agents). These ant agents explore the environment and bring relevant information back to their responsible task agent, evaluate optional paths, and put the intentions of their task agent as information in the environment. This allows delegate MASs of different agents to coordinate by aligning or adapting the information in the environment according to their own tasks.

2. BASIC COMPONENTS

The basic architecture comprises the task agents, resource agents, and the environment. The environment of the targeted applications is a dynamic directed graph. The nodes in the graph correspond to the resources in the environment and the edges represent connections between different resources.

A resource agent represents a resource in the environment and contains an information processing part for controlling the resource. In manufacturing control systems, a resource agent is an abstraction of the production means such as machines, conveyors, tool holders, material storage or even personnel. A resource can abstractly be described as a set of capabilities, i.e. the operations that the resource is able to perform. Resource agents need to be able to make schedules based on requests from task agents, and must be able to answer "what-if questions": a task agent may ask a re-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

source agent when and according to what quality standards a particular operation could be performed if the task would arrive at a particular moment in the future.

A task agent represents and controls a task in the coordination and control application, and resides on a (physical or virtual) mobile entity in the environment. A task agent is responsible for performing its task by guiding its mobile unit through the environment, and communicating with resource agents in order to perform operations on the unit. Every task agent is aware of the schemes or plans that can be followed in order to reach its goal. For manufacturing control, the task agents represent (unfinished) client orders and are associated with the pallets with partially fabricated products. The production schemes describe possible sequences of operations on the product in order to obtain the final product. Task agents are obvious candidates to be modeled as BDI agents. Task agents need to deal with the observations of the environment and its entities (beliefs), consider possible options on how to proceed (desires), choose a particular option (intention) and communicate and coordinate this with the other task agents.

Both resource and task agents control the entities in the environment, and obviously need to interact to achieve the goal of the application. A typical interaction amongst these agents goes as follows. When a new "task" enters the system, a task agent is created and connected to the appropriate unit. The agent is aware of the initial state of the task and investigates possible next operations that can be performed. The agent searches and selects a combination of a next processing step and a suitable resource that has the capability to execute the step. When the selected processing step is executed by the resource, the resource reports on the new state of the task. This may or may not be the expected outcome of the operation. Based on this state, the task agent investigates possible next steps. The process repeats until the task is finished.

3. DELEGATE MULTI-AGENT SYSTEMS

A typical approach would be to use direct communication protocols, knowledge engineering and means-ends reasoning to achieve the required coordination. Here, we use delegate MASs for (1) obtaining feasibility information, (2) exploration and (3) propagation of intentions toward the resource agents. The following interesting principles are recognized in "ant-like multi-agent systems": (1) make the environment part of the solution to handle a complex environment without being exposed to its complexity - ants are quite simple agents; (2) place relevant information as signs in the environment ensuring that locally available data informs about remote system properties; (3) limit the lifetime of this information (evaporation) and refresh the information as long as it remains valid - this allows the system to cope with changes and disturbances. We exploit these principles in our approach and define three types of light-weight agents, which each represent a different delegate MAS and which share a common environment for indirect communication. To distinguish them from task and resource agents, we call the light-weight agents "ant agents" or ants further on. Delegate MASs consist of ant agents that reside in a virtual software environment which reflects the application environment, and in which ant agents can navigate.

Feasibility Ants. Feasibility ants form a delegate MAS that is issued by resource agents. Their purpose is to roam

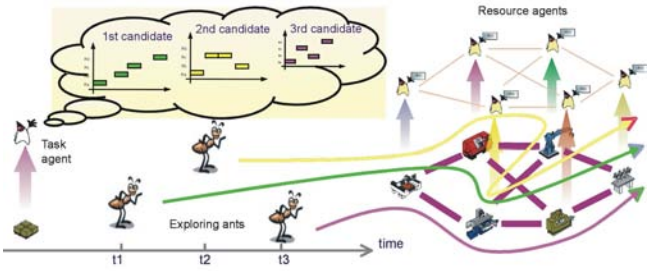


Figure 1: ***Exploration ants scout feasible paths***

the environment and, at each node they pass, drop information on feasible paths that start from this node. Resource agents which have not seen a feasibility ant passing by for a particular period will create new feasibility ants themselves at a certain frequency. The behavior of a feasibility ant is as follows. A feasibility ant communicates with the resource agent at its current node, and asks for its capabilities. The ant observes the environment locally and finds out from which other nodes the current node can be reached. The ants can be seen as moving upstream. When arrived at the new node, the ant asks the local resource agent for its capabilities, and merges this information with information of previous nodes. Now this ant knows that from this node, a sequence of operation that requires a capability from the current node followed by an operation that can be performed at its previous node is currently feasible. This information is dropped at the local information space of the current node, as a kind of road sign. Then the ant clones itself for every node upstream, and the process repeats itself. A feasibility ant dies if there are no nodes upstream. The information that is stored in local information spaces is time-stepped, and if not refreshed by another feasibility agent in time, that information disappears.

Exploration Ants. Using the feasibility information available locally, a task agent is able to find out which paths are physically or virtually feasible for achieving the goal of its task. A task agent generates exploration ants at a certain frequency which explore feasible paths. These exploration ants are scouts that each explore a feasible route through the underlying system and evaluate this route. To make the evaluation, an exploration ant follows a path through the environment, and interacts with the resource agents at the different nodes by asking the resource agent *what* the timing, quality or costs would be *if* a task in a particular state would arrive at a particular moment in time. The exploration ant collects this information, and then proceeds to the next node in the path, in which this behavior is repeated. When arrived at the end of the path, the exploration ant returns and reports back to its base, i.e. the task agent that created the exploration ant. Figure 1 illustrates this process for a simple factory (see right-hand side of figure, cf. www.mech.kuleuven.be/benchmarking). The task agent on the left hand side creates three exploration ants for scouting feasible paths.

The list of candidates get refreshed regularly as exploration ants are sent out regularly. Candidates that are not refreshed are removed over time, assuming that these candidates have become invalid or infeasible because of changes in the environment.

Intention Ants. Exploration as described above requires the resource agents to possess an adequate estimate of their future workload. To serve this purpose, task agents generate intention ants, which propagate the intention of task agents through the environment. When a task agent has constructed a set of valid paths to follow, the task agent selects one candidate path to become its intention. The criteria used for this selection depends on the requirements of the task, and is application-specific. Then, the task agent creates intention ants, at a certain frequency, to inform the resource agents that are involved in this intended path. The intention ants follow the selected path, and virtually execute the routing and processing of their selected candidate solution. On their virtual journey, the intention ants acquire travel, queuing, and processing times from the resource agents on their path. Similar as exploration ants, intention ants report back to their task agent to inform the agent about the schedule and performance of the bookings.

The intention information at the resource agent - the booking - evaporates. Task agents must create intention agents to refresh their intention at a frequency that is sufficiently high to maintain their bookings at the resources. While refreshing, a task agent observes the evolution of the expected performance of its current intentions through the reports on the estimated performance that intention ants bring back. This performance estimate is compared to the estimates of the candidate solutions that are found and refreshed by the exploration agents. When the estimated performance of the current intention drops significantly below the estimated performance of other candidate solutions, the task agent may revise its intention. When the task of a task agent reaches the point where a decision needs to be executed, the task agent triggers the action in the underlying system in accordance with the intention.

Task and Resource Agents. Let us now list the responsibilities of task and resources agents. These lists should guide a developer of a coordination and control application to define a concrete software architecture for these agents.

A resource agent is responsible for (1) answering what-if questions from exploration ants, (2) making schedules based on requests for bookings by intention ants - the schedules must obviously respect resource constraints as well as follow a predefined policy for re-scheduling, e.g. when high-priority tasks make a booking, this may reject earlier reservations, (3) keep an up-to-date view on the resource that the resource agent is managing (e.g. observe operation quality and status), and (4) send out feasibility ants if no feasibility ants have contacted this resource agent for a while. These guidelines should suffice to produce a concrete architecture according to the requirements of a concrete application. The resource agent could execute each of these responsibilities in sequence, or one may opt to define concurrent execution of some of these responsibilities.

The responsibilities of the task agents are the following. First, a task agent must have knowledge about its task, about the initial state of the task, and about task schemes. Task schemes describe one or several plans to achieve the goal of the task, and given any intermediate state of the task, the task scheme should provide one or more sub-plans to fulfill the task. Second, a task agent must manage its beliefs. Beliefs on feasibility are readily available from the information space in the environment. Beliefs about explored paths are gathered by exploration ants. Third, this information needs to be filtered out, yielding valid paths - the agents desires. Fourth, based on the beliefs about the task and the options, a task agent then chooses or revises its intention. Fifth, at appropriate times and frequency, exploration ants as well as intention ants are sent out. Finally, a task agent is responsible for interacting with resource agents in order to perform operations on its task. Again, these responsibilities are either executed as a sequence, or one may choose for concurrent execution of some of these responsibilities.

4. EVALUATION AND CONCLUSION

In the approach we propose, we stick to the basic philosophy of belief-desires-intention agents, but exploit the environment and delegate MASs, inspired by ant behavior, to realize beliefs, desires and intentions. Based on simulation experiments for a real-world case of manufacturing control, we can state that this innovative approach yields very interesting results for the targeted application domains, both in terms of the reduction of the agent software complexity and in overall performance in a highly dynamic environment [5].

One important difference between an approach using delegate MAS and traditional BDI approach for coordination and control systems is that a delegate MAS design extends Brooks concept of having the world as its own best model, while traditional BDI approaches rely on maintaining world models. The basic idea of having the "world as its own best model" only discusses the present state of the world. In contrast, the delegate MAS approach in this paper extends this idea toward the future state, using exploration and intention ants, while keeping modeling efforts acceptable. Indeed, resource agents only need to be knowledgeable and intelligent about their own small section of the world. Known scalable MAS coordination and control developments are myopic [2, 3], while our approach allows to take future state into account.

5. REFERENCES

[1] M. E. Bratman. *Intentions, Plans, and Practical Reason*. Harvard, Cambridge, MA, USA, 1987.

[2] S. Bussmann, N. Jennings, and M. Wooldridge. *Multiagent systems for manufacturing control: A design methodology*, volume XIV of *Springer Series on Agent Technology*. Springer-Verlag, 2004.

[3] H. Parunak, A. D. Baker, and S. J. Clark. The aaria agent architecture: From manufacturing requirements to agent-based system design. *Integrated Computer-Aided Engineering*, 8(1):45–58, 2001.

[4] A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995.

[5] P. Valckenaers and H. V. Brussel. Holonic manufacturing execution systems. *CIRP Annals-Manufacturing Technology*, 54(1):427–432, 2005.

[6] D. Weyns, H. V. D. Parunak, F. Michel, T. Holvoet, and J. Ferber. Environments for multiagent systems state-of-the-art and research challenges. In D. Weyns, H. V. D. Parunak, and F. Michel, editors, *E4MAS*, volume 3374 of *Lecture Notes in Computer Science*, pages 1–47. Springer, 2004.

User Defined Monitoring Strategies for BDI Agent Programs *

Timothy William Cleaver
Griffith University
Southport, Australia
Tim.Cleaver@student.griffith.edu.au

Abdul Sattar
Griffith University
Brisbane, Australia
a.sattar@griffith.edu.au

Raihana Ferdous
Griffith University
Southport, Australia
Raihana.Ferdous@student.griffith.edu.au

ABSTRACT

An agent needs to balance its time between reflecting on its past behaviors, the execution of its current behaviors and the recognition of opportunities for future action. The optimal strategy to achieve this balance is heavily dependent on the environment in which the agent is situated. In this paper we extend a BDI agent programming language called AgentSpeak(L) in two ways. Firstly we refine the use of execution monitoring to enable agents to reflect on their past endeavors. Secondly, we extend the semantics and syntax of the language to allow for user defined monitoring strategies. This allows agent designers to utilize execution monitoring in a way that balances the focus of an agents behavior in a way that is appropriate for its application domain.

Keywords: Agent and Multi-Agent Architectures, Agent Programming Languages

1. INTRODUCTION

For an intelligent agent, behaving in complex and real-time environments is a balancing act. There are three primary tasks that must be managed in order to achieve this balance. An agent must be able to recognize opportunities for future conduct. Simultaneously, the agent must devote resources, including its cognitive abilities, towards the execution of its current behaviors. All the while the agent must be watchful to collate its perceptions with the outcomes of its previous behaviors so as to maintain an accurate view of its progress towards its goals. This must all be done while remaining responsive to real-time constraints on its behavior.

This paper proposes the use of execution monitoring to allow BDI systems to correlate their perceptions with their previous behavior. More specifically we argue that providing the ability for agent developers to integrate user defined monitoring strategies is the only way to do this efficiently given the variety of domains for which BDI systems are applicable. AgentSpeak(L) was chosen as the prototypical BDI system for reasons outlined in Section 3 and it is the framework into which we have integrated execution monitoring. However, prior to providing an overview of AgentSpeak(L) and our extensions (Section 4) we describe the execution monitoring problem for BDI systems.

*This work is supported by the SIT-CRC

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. THE MONITORING PROBLEM

Monitoring is the mental act of validating expectations through the inspection of ones world model. There are three main things that a BDI system should be monitoring for: success, validity and failure. In dynamic and complex environments, monitoring for one cannot be reduced to monitoring for another[5]. Ideally an agent should continuously monitor the environment to determine the outcomes of its behaviors. However, in resource-bounded practical agents the monitoring process has a cost. Therefore an agent needs to schedule the times at which it monitors. This way an agent makes a trade-off between its introspection capabilities and the time constraints imposed upon it by the environment in which it is embedded. Thus, an agent needs to minimize the cost of monitoring. Be it the cost resulting from not monitoring when monitoring would have been beneficial or the cost of the monitoring process itself when the monitoring process has no effect on the agent's behavior.

3. WHAT IS AGENTSPEAK(L)?

AgentSpeak(L)[9] is an abstract architecture and programming language that aims to embody the BDI philosophy[4] and logic[10] in a computationally attractive way. It has a number of semantics ranging from the Z specification language [7] to structural operational semantics [8, 3, 1] and a sub-set of the language has been demonstrated, through model-checking, to satisfy a number of specifications expressed in BDI logic formula[2]. Due to its simplicity and elegance, AgentSpeak(L) is easy to extend and is designed in such a way that customizing the architecture and language to suit the application domain is straight forward.

4. MONITORING IN AGENTSPEAK(L)

Recently, the AgentSpeak(L) language and architecture was extended to facilitate the specification of and reasoning with success and failure conditions of plan execution[5, 6]. However, one significant drawback of the approach proposed was that the architecture utilized a naïve approach to monitoring execution. Each success and failure condition of each intended means in each intention stack was checked once each reasoning cycle. This results in significant proportions of the agents cognitive faculties being devoted to the monitoring process at the expense of its other activities. In order to balance the monitoring process with action execution, opportunity recognition and belief revision an agent needs to be able to make decisions as to when to monitor for the success and failure of its

plans. We believe that the optimal strategy is dependent on both the application domain and the deployment environment. To cater for this we incorporate monitoring strategies by way of an additional user function in the architecture of the agent. We build upon the structural operational semantics and syntax developed in [1] and extended in [5, 6]. The primary contribution of this paper required no modification to the extended syntax of the language. However it was necessary to add new definitions, auxiliary functions and rules to the semantics of the interpreter.

4.1 Definitions

We extend the definition of an agent to include two new user defined functions: $F_M$and F_{ConRes}. $F_M$encapsulates the agents monitoring strategy. Given that both implemented BDI systems and BDI logic[10] represent time discreetly, this function takes the current state of the agent, a mental attitude to monitor and maps them to true or false ($M : S \times At \mapsto [\top, \bot]$). True indicates the agent needs to check the condition this reasoning cycle and false indicates the agent need not. $F_{ConRes} : \xi \times B \mapsto [\top, \bot, \dashv]$defines the agents conflict resolution strategy. $F_{ConRes}$is necessary because the agent may come to believe evidence for both the success and failure of an effect of a plans execution. It is used to alleviate the ambiguity of whether to conclude success or failure in these cases. $F_{ConRes} : \xi \times B \mapsto [\top, \bot, \dashv]$maps an effect ($\xi$) of a plan and the current beliefs (B) of the agent to one of three values: $\top$ if the agent should conclude true given this ambiguity, $\bot$ if the agent should conclude false under uncertainty and $\dashv$ if the agent can conclude nothing this cycle and should continue monitoring this effect.

One additional temporary data store is also required: η. During each reasoning cycle, this set is generated based on the user function $F_M$to include all the effects that the agent needs to check for either success or failure. Being temporary (the contents do not persist across reasoning cycles), this data store is referred to in the semantics as T_η.

4.2 Auxiliary Functions

Here we define the filtering function ($\mathbf{FilterInt} : C_I \mapsto \{\pi\}$). It returns the sub-set of the current set of intentions (C_I) such that each included intention has elements remaining in the body. This function is used to prevent intentions that have completed from executing but have effects outstanding from being candidates for processing on this reasoning cycle.

$$\mathbf{FilterInt}(C_I) = \{\pi | \pi = \pi'[l \rightarrow te : c \leftarrow \{i\}h.e] \wedge h \neq \emptyset\}$$

This function ($\mathbf{GenEff} : T \times C \times AG \mapsto \{\langle \pi, \varrho, \xi \rangle\}$) generates the set of effects that should be monitored this reasoning cycle. This is achieved by iterating over each intention in the set of intentions ($\forall\pi$), each plan on each intention stack ($\forall\varrho$) and each effect within each plan ($\forall\xi$). This combination, in addition to the current state of the agent (T, C), is passed onto the user defined effect monitoring strategy function ($F_M(T, C, \pi, \varrho, \xi)$). This function defines whether this effect should be checked in this cognitive cycle. The return value of this function is the set of tuples $\langle \pi, \varrho, \xi \rangle$ for which the ($F_M(T, C, \pi, \varrho, \xi)$) function returns true ($\top$).

$$\mathbf{GenEff}(T, C, AG) = \left\{ \langle \pi, \varrho, \xi \rangle \;\middle|\; \begin{array}{l} \forall \pi \varrho \xi \text{ s.t. } \pi \in C_I \wedge \varrho \in \pi \wedge \\ l \rightarrow te : c \leftarrow \{i\}h.e = \varrho \wedge \\ \xi \in e \wedge AG_{F_M(T,C,\pi,\varrho,\xi)} = \top \end{array} \right\}$$

Because we regenerate the set of effects to monitor each reasoning cycle, we need to remove those effects that have sufficient evidence for the agent to conclude their success from their encompassing plan. Otherwise, the agent will continue to monitor for these effects and the plan will never be considered complete. We do this using the update effects function ($\mathbf{UpdEff} : \pi \times \varrho \times \xi \mapsto \pi'$). This function maps an intention, plan and effect into a new intention. Because an agent can monitor for and consequently conclude the success of an effect in any level of an intention stack, this function retrieves the input plan ($l \rightarrow te : c \leftarrow \{i\}h.e$) from the intention ($\pi$)[1], removes the input effect (ξ) from the input plan ($l \rightarrow te : c \leftarrow \{i\}h.e$) and reconstructs the intention stack (π') to be returned.

$$\mathbf{UpdEff}(\pi, l \rightarrow te : c \leftarrow \{i\}h.e, \xi) = \pi'$$

where: $\pi = \pi''[l \rightarrow te : c \leftarrow \{i\}h.e\backslash\xi]\pi'''$ s.t. $\pi''[l \rightarrow te : c \leftarrow \{i\}h.e]\pi''' = \pi$

4.3 Rules

The first rule that we must alter from the original is that of **SelInt**. We do so in order to integrate the intention filtering function (as described above) into the intention selection process. The first rule is applicable in the case where there are intentions for which further processing is required. The second handles the case where there are not.

$$\mathbf{SelInt}\frac{\mathbf{FilterInt}(C_I) \neq \emptyset \qquad F_I(\mathbf{FilterInt}(C_I))) = T_\iota}{\langle AG, C, T, \mathbf{SelInt}\rangle \longrightarrow \langle AG, C, T', \mathbf{ExecInt}\rangle}$$

$$\mathbf{SelInt}\frac{\mathbf{FilterInt}(C_I) = \emptyset}{\langle AG, C, T, \mathbf{SelInt}\rangle \longrightarrow \langle AG, C, T, \mathbf{GenEff}\rangle}$$

We do not repeat the rules for executing the top most plan of the selected intention as they remain identical to those in [3]. The next process is that of generating the set of effects to check. We employ two rules to do this. The first handles the case where the set of intentions is empty. If the set of intentions is empty then there can be no effects to be monitoring for and we skip the procedure and transition to clearing the intentions.

$$\mathbf{GenEff}\frac{C_I = \emptyset}{\langle AG, C, T, \mathbf{GenEff}\rangle \longrightarrow \langle AG, C, T, \mathbf{ClrInt}\rangle}$$

However, should there be elements in the set of intentions the second rule passes the current set of intentions as a parameter to the effect generation function and stores the result in the set of effects to be checked (T_η).

$$\mathbf{GenEff}\frac{C_I \neq \emptyset \qquad \mathbf{GenEff}(T, C, AG) = T_\eta}{\langle AG, C, T, \mathbf{GenEff}\rangle \longrightarrow \langle AG, C, T', \mathbf{UpdEff}\rangle}$$

Four rules are required to process the current effects. The first three are repeatedly applied until there are no elements left in the effect set. The first is applicable when there are outstanding effects to be checked($T_\eta \neq \emptyset$) and the selected effect is deemed successful ($\top = F_{ConRes}(\xi, B)$) by the conflict resolution strategy (F_{ConRes}). It removes the effect from the set of effects awaiting monitoring and from the plan from which it originated. It also updates the intention using the ($\mathbf{UpdEff}$) function.

$$\mathbf{UpdEff}\frac{T_\eta \neq \emptyset \qquad \langle \pi, \varrho, \xi \rangle \in T_\eta \wedge \top = F_{ConRes}(\xi, B)}{\langle AG, C, T, \mathbf{UpdEff}\rangle \longrightarrow \langle AG, C', T', \mathbf{UpdEff}\rangle}$$

where:

$T'_\eta = T_\eta \backslash \langle \pi, \varrho, \xi \rangle$

$C'_I = (C_I \backslash \pi) \cup \mathbf{UpdEff}(\pi, \varrho, \xi)$

[1]producing $\pi''[l \rightarrow te : c \leftarrow \{i\}h.e']\pi'''$ where π'' and π''' represent the elements of the intention both below and above the required plan respectively

The second rule is also applicable when there are outstanding effects to be checked, but requires the conflict resolution strategy to judge the selected effect to have failed. This rule creates the appropriate failure event and adds it to the event set. Then the intention is removed from the set of intentions to prevent its further processing until the failure is handled.

$$\mathbf{UpdEff}\frac{T_\eta \neq \emptyset \qquad \langle \pi, \varrho, \xi\rangle \in T_\eta \wedge \bot = F_{ConRes}(\xi, B)}{\langle AG, C, T, \mathbf{UpdEff}\rangle \longrightarrow \langle AG, C', T', \mathbf{UpdEff}\rangle}$$

where: $C'_E = \begin{array}{ll} C_E \cup \{(+g, \pi)\} & \text{if } te = -g \text{ (where g is a goal)} \\ C_E \cup \{(-g, \pi)\} & \text{if } te = +g \text{ (where g is a goal)} \end{array}$

$C'_I = C_I \backslash \pi$

$T'_\eta = T_\eta \backslash \langle \pi, \varrho, \xi\rangle$

The third rule handles the case where there are effects to audit, one of these is selected, yet the conflict resolution strategy can not conclude that the effect has been satisfied or invalidated. All that must be done in this case is to remove the effect from the effect set so that it will not be tested again.

$$\mathbf{UpdEff}\frac{T_\eta \neq \emptyset \qquad \langle \pi, \varrho, \xi\rangle \in T_\eta \wedge \dashv = F_{ConRes}(\xi, B)}{\langle AG, C, T, \mathbf{UpdEff}\rangle \longrightarrow \langle AG, C, T', \mathbf{UpdEff}\rangle}$$

where: $T'_\eta = T_\eta \backslash \langle \pi, \varrho, \xi\rangle$

Finally, the last (**UpdEff**) rule handles the case where there are no elements in the set of effects to review.

$$\mathbf{UpdEff}\frac{T_\eta = \emptyset}{\langle AG, C, T, \mathbf{UpdEff}\rangle \longrightarrow \langle AG, C, T, \mathbf{ClrInt}\rangle}$$

There is only one primary modification required to the original (**ClrInt**) function and that is to retain plans that have outstanding effects. This prevents the continued processing of these intentions until all the effects have been realized or demonstrated to have failed.

$$\mathbf{ClrInt}\frac{C_I \neq \emptyset}{\langle AG, C, T, \mathbf{ClrInt}\rangle \longrightarrow \langle AG, C', T', \mathbf{SelEv}\rangle}$$

where: $C'_I = \begin{cases} C_I \backslash T_\iota & \text{if } \begin{array}{l} T_\iota = [l \rightarrow te : c \leftarrow \{i\}h.e] \\ \wedge i = h = e = \emptyset \end{array} \\ (C_I \backslash T_\iota) \cup \{T'_\iota\} & \text{if } \begin{array}{l} T_\iota = \pi[l \rightarrow te : c \leftarrow \{i\}!at; h.e] \\ [l' \rightarrow te' : c' \leftarrow \{i'\}h'.e'] \wedge \\ i' = h' = e' = \emptyset \end{array} \\ C_I & \text{otherwise} \end{cases}$

$T = \emptyset$

5. CONCLUSION

In conclusion, through our studies of existing BDI systems we discovered that these systems do not have strong capabilities for monitoring the outcomes of their behaviors. Particularly, existing systems have no methodology for utilizing user defined monitoring strategies that facilitate effective balance of the time an agent spends monitoring and doing its other activities. We chose as our prototypical BDI implementation, AgentSpeak(L), and made extensions to its semantics in an attempt to address this issue. Due to the sensitivity of the performance of an agent to the environment in which it is situated, utilizing user defined strategies that are tuned to the particular application domain is essential. We believe that agents utilizing these extensions will perform better due to their ability to better balance the time spent correlating their current beliefs with their past behaviors, the time spent recognizing opportunities and executing current behaviors.

6. REFERENCES

[1] R. Bordini, M. Dastani, J. Dix, and A. Seghrouhni. *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, 2005.

[2] R. Bordini, M. Fisher, C. Pardavila, and M. Wooldridge. Model checking AgentSpeak. In *Proceedings of the second International Joint Conference on Autonomous Agents and Multi-agent Systems*, pages 409–416, 2003.

[3] R. Bordini and A. Moreira. Proving BDI properties of agent-oriented programming languages: The asymmetry thesis principles in agentspeak(l). *Annals of Mathematics and Artificial Intelligence*, 42:197–226, 2004.

[4] M. Bratman. *Intention, Plans and Practical Reason*. Harvard University Press, 1987.

[5] T. Cleaver, A. Sattar, and K. Wang. Reasoning about success and failure in intentional agents. In *Proceedings of the 2005 Pacific Rim International Workshop on Multi-Agents*, 2005.

[6] T. Cleaver, A. Sattar, and K. Wang. Reasoning with plan outcomes in intentional agents. In *Proceedings of the eighteenth Australian Joint Conference on Artificial Intelligence (to appear)*, 2005.

[7] M. d'Inverno and M. Luck. Engineering AgentSpeak(L): A Formal Computational Model. *Journal of Logic and Computation*, 8(3):1–27, 1998.

[8] A. Moreira, R. Vieira, and R. Bordini. Extending the Operational Semantics of a BDI Agent-Oriented Programming Language for Introducing Speech-Act based Communication. In J. Leite and A. Omicini and L. Sterling and P. Torroni, editor, *Proceedings of the first International Workshop on Declarative Agent Languages and Technologies*, volume 2990 of *Lecture Notes in Computer Science*, pages 135–154. Springer, 2003.

[9] A. Rao. Agentspeak(L): BDI Agents Speak out in a Logical Computable Language. In W. van de Velde and J. Perrame, editors, *Proceedings of the seventh European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, volume 1038 of *Lecture Notes in Artificial Intelligence*, pages 42–55. Springer-Verlag, 1996.

[10] A. Rao and M. Georgeff. Decision procedures for BDI logic. *Journal of Logic and Computation*, 8(3), 1998.

Mixed-integer Linear Programming for Transition-independent Decentralized MDPs

Jianhui Wu
EECS Department, University of Michigan
Ann Arbor, MI 48109 USA
jianhuiw@umich.edu

Edmund H. Durfee
EECS Department, University of Michigan
Ann Arbor, MI 48109 USA
durfee@umich.edu

Categories and Subject Descriptors

I.2.8 [**ARTIFICIAL INTELLIGENCE**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms

Keywords

MDP, transition-independent decentralized MDP, piecewise linear approximation, mixed integer linear programming

1. INTRODUCTION

Markov decision process (MDP) provides a framework for computing optimal policies for individual agents operating in uncertain environments. However, extending single-agent MDP techniques to multiagent problems is not straightforward. Previous complexity analyses have shown that the general decentralized Markov decision process (Dec-MDP) is NEXP-complete, which means that optimally solving a Dec-MDP is extremely difficult. The class of problems studied in this paper is a subclass of Dec-MDP in which two or more cooperative agents are tied together through the rewards of completing joint tasks but the actions taken by one agent do not impact other agents' transitions. Although this reduces the complexity class to NP-complete [4], efficiently solving such transition independent Dec-MDPs is still nontrivial.

Optimal algorithms, such as the coverage-set algorithm [2], slow down dramatically as the size of the problem increases. Approximation algorithms, such as the JESP algorithm [5], can quickly find a locally optimal joint policy, but the solutions usually lack quality guarantees. Other collaborative multiagent research often makes the simplifying assumption that each agent has the same (complete) world view. Thus, there remains a critical need for new algorithms that can efficiently build optimal or near-optimal policies for transition independent Dec-MDPs. In this paper, we present a novel mathematical programming algorithm for this purpose.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. MDPS AND LINEAR PROGRAMMING

A classical MDP can be defined as a four-tuple $\langle S, A, P, R\rangle$, where S is a finite state space, A is a finite action space, $P = \{p_{i,a,j}\}$ is the state transition probability where $p_{i,a,j}$ is the probability that the agent reaches state j if it executes action a in state i, and $R = \{r_{i,a}\}$ is the reward function where $r_{i,a}$ is the reward that the agent receives if it executes action a in state i.

One approach to solving MDPs is based upon linear programming (LP) [1]. Let $x_{i,a}$ represent the expected number of times action a is executed in state i. Then the objective function $\sum_i \sum_a x_{i,a} \times r_{i,a}$ represents the total expected reward. Now, consider the linear program shown in Eq.1.

$$\max \sum_i \sum_a x_{i,a} \times r_{i,a} \left| \begin{array}{l} \sum_a x_{j,a} = \alpha_j + \sum_i \sum_a p_{i,a,j} \times x_{i,a} \\ x_{i,a} \geq 0 \end{array} \right. \quad (1)$$

where α_j is the probability that the system is initially in state j. The constraint $\sum_a x_{j,a} = \alpha_j + \sum_i \sum_a p_{i,a,j} \times x_{i,a}$ indicates that the expected number of times state j is visited must equal the initial probability distribution at state j plus the expected number of times state j is entered via all possible transitions.

If we solve Eq.1, it is trivial to derive the optimal policy. Specifically, assigning a probability of executing action a in state i as $\pi_{i,a} = \frac{x_{i,a}}{\sum_a x_{i,a}}$ will maximize the total expected reward.

3. ALGORITHM

In the class of problems studied in this paper, multiple cooperative agents aim to maximize their total expected reward in a stochastic environment where actions taken by one agent cannot impact other agents' transitions. Each agent can fully observe its own local state but can only have a partial view of the global environment. An agent receives rewards by completing its individual tasks. Agents can also work together to complete joint tasks. The accomplishment of a joint task needs participation from multiple agents. That is, if every (relevant) agent completes its subtask of the joint task, then the group receives a joint reward; otherwise they get zero reward for that joint task. Since agents might not be able to complete all available tasks due to limitations (such as time), they need to determine the subset of tasks they should pursue, and align their policies.

Let $p^m_{i,a,j}$, $r^m_{i,a}$, and α^m_j, respectively, represent the transition probabilities, rewards (or costs) for individual activities, and initial probability distribution for agent m, let $p^{m,k}_{i,a}$

represent the probability that agent m completes its subtask of joint task k when executing action a in state i, and let R^k represent the reward for accomplishing joint task k (i.e., $R^k > 0$). Our problem can then be formulated as

$$\max \sum_m \sum_i \sum_a x_{i,a}^m \times r_{i,a}^m + \sum_k \rho^k \times R^k$$
$$\left| \begin{array}{l} \sum_a x_{j,a}^m = \alpha_j^m + \sum_i \sum_a p_{i,a,j}^m \times x_{i,a}^m \\ \rho^{m,k} = \sum_i \sum_a p_{i,a}^{m,k} \times x_{i,a}^m \\ \rho^k = \prod_m \rho^{m,k} \\ x_{i,a}^m \geq 0 \end{array} \right. \quad (2)$$

where $x_{i,a}^m$ is the expected number of times that action a is executed in state i for agent m. $\rho^{m,k} = \sum_i \sum_a p_{i,a}^{m,k} \times x_{i,a}^m$ is the expected number of times agent m completes its subtask of joint task k. Note that it is generally not difficult to restrict an autonomous agent from completing a particular task more than once, either through an additional bit in the state representation or through a fixed acyclic ordering for completing the tasks, and so $\rho^{m,k}$ can also represent the probability of agent m completing its subtask of joint task k. $\rho^k = \prod_m \rho^{m,k}$ is the probability that the group successfully completes joint task k.

The term $\sum_m \sum_i \sum_a x_{i,a}^m \times r_{i,a}^m$ in the objective function of Eq.2 is the total expected reward (or cost) of individual activities, and the term $\sum_k \rho^k \times R^k$ is the total expected reward from accomplishing joint tasks. The constraint $\sum_a x_{j,a}^m = \alpha_j^m + \sum_i \sum_a p_{i,a,j}^m \times x_{i,a}^m$ is a multiagent version of the constraint in Eq.1.

It is important to emphasize that, unlike an LP formulation (e.g., Eq.1), Eq.2 uses nonlinear constraints to represent cooperative activities of agents (instead of using the joint state space) and thus avoids exponential increase in the size of the mathematical formulation. In the following two subsections, we present discretization and piecewise linear approximation techniques, and then reformulate Eq.2 into a mixed integer linear program (MILP) that can be efficiently solved by state-of-art algorithms [3].

3.1 Discretization

We now introduce a discretization technique that, although it cannot eliminate the nonlinearities in Eq.2 by itself, paves the way for using piecewise linear approximation technique presented in Section 3.2.

Let $\mathbb{H}$ denote an ordered set of constants (named *thresholds*) in which $0 \leq h_1 < h_2 < \ldots < h_i < \ldots \leq 1$, and define binary variables Θ_h^k. Then, with additional constraints Eq.3,

$$\left| \begin{array}{l} \frac{\rho^k}{h} \geq \Theta_h^k \\ \sum_h \Theta_h^k \leq 1 \\ \Theta_h^k \in \{0,1\} \end{array} \right. \quad (3)$$

the objective function of Eq.2 can be approximated as

$$\sum_m \sum_i \sum_a x_{i,a}^m \times r_{i,a}^m + \sum_k \sum_{h \in \mathbb{H}} h \times \Theta_h^k \times R^k \quad (4)$$

The constraint $\frac{\rho^k}{h} \geq \Theta_h^k$ is equivalent to $\Theta_h^k = 0$ when $\rho^k < h$, and $\Theta_h^k \in \{0,1\}$ when $\rho^k \geq h$. The constraint $\sum_h \Theta_h^k \leq 1$ guarantees that, for each joint task k, there is at most one nonzero integer in the set $\{\Theta_h^k \mid h \in \mathbb{H}\}$. These two constraints work together to make $\sum_{h \in \mathbb{H}} h \times \Theta_h^k \leq \rho^k$. Since rewards of joint tasks are always positive, $\sum_{h \in \mathbb{H}} h \times \Theta_h^k$ equals the largest h in the set $\{h \mid h \in \mathbb{H}$ and $h \leq \rho^k\}$ when Eq.4 is maximized. That is, $\sum_{h \in \mathbb{H}} h \times \Theta_h^k$ can nicely approximate ρ^k when the maximum gap between two adjacent thresholds in $\mathbb{H}$ is sufficiently small.

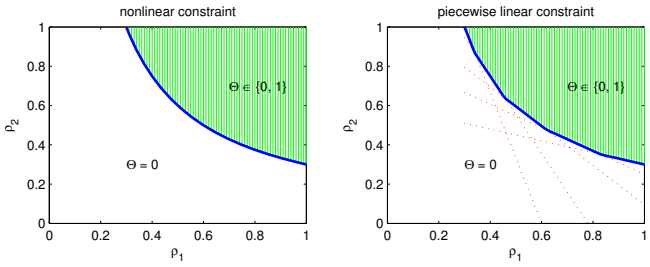


Figure 1: An example of piecewise linear approximation ($h = 0.3$, two agents).

3.2 Piecewise Linear Approximation

With the discretization technique, we are ready to remove the nonlinearities. Note that, in our new formulation (i.e., adding Eq.3 into Eq.2, and replacing Eq.2's objective function with Eq.4), ρ^k are trivial variables, which can be taken away by combining constraints $\rho^k = \prod_m \rho^{m,k}$ and $\frac{\rho^k}{h} \geq \Theta_h^k$ into a new constraint $\frac{\prod_m \rho^{m,k}}{h} \geq \Theta_h^k$.

In general, the nonlinear constraint $\frac{\prod_m \rho^{m,k}}{h} \geq \Theta_h^k$ can be closely approximated with a sufficient number of linear constraints. Figure 1 shows a simple example where a nonlinear constraint $\frac{\rho^1 \times \rho^2}{h} \geq \Theta_h$ ($h = 0.3$) is approximated by five linear constraints.

Explicitly, for M agents, we can build a piecewise linear matrix for threshold h as:

$$\mathbb{A}_h = \begin{bmatrix} \frac{1}{\varrho_1^1 \times M} & \cdots & \frac{1}{\varrho_1^m \times M} & \cdots & \frac{1}{\varrho_1^M \times M} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{1}{\varrho_i^1 \times M} & \cdots & \frac{1}{\varrho_i^m \times M} & \cdots & \frac{1}{\varrho_i^M \times M} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{1}{\varrho_n^1 \times M} & \cdots & \frac{1}{\varrho_n^m \times M} & \cdots & \frac{1}{\varrho_n^M \times M} \end{bmatrix}$$

where $\overrightarrow{\varrho_i} = \langle \varrho_i^1, \ldots, \varrho_i^M \rangle$ represents a point on the function $\prod_m \rho^m = h$.[1] $\frac{\prod_m \rho^{m,k}}{h} \geq \Theta_h^k$ can then be approximated as $\mathbb{A}_h \times \overrightarrow{\rho}^k \geq \Theta_h^k$, where $\overrightarrow{\rho}^k = \langle \rho^{1,k}, \ldots, \rho^{M,k} \rangle$.

3.3 MILP-HCS Algorithm

Using the approximation techniques presented above, a transition-independent Dec-MDP problem can then be formulated as the following mixed integer linear program.

$$\max \sum_m \sum_i \sum_a x_{i,a}^m \times r_{i,a}^m + \sum_k \sum_{h \in \mathbb{H}} h \times \Theta_h^k \times R^k$$
$$\left| \begin{array}{l} \sum_a x_{j,a}^m = \alpha_j^m + \sum_i \sum_a p_{i,a,j}^m \times x_{i,a}^m \\ \rho^{m,k} = \sum_i \sum_a p_{i,a}^{m,k} \times x_{i,a}^m \\ \mathbb{A}_h \times \overrightarrow{\rho}^k \geq \Theta_h^k \\ \sum_h \Theta_h^k \leq 1 \\ x_{i,a}^m \geq 0 \\ \Theta_h^k \in \{0,1\} \end{array} \right. \quad (5)$$

[1] In this work, we use a greedy algorithm to find n points $\overrightarrow{\varrho_i}$ for $\mathbb{A}_h$.

We can then use state-of-art MILP solvers for it, and derive a joint policy in a similar way to that described in Section 2. That is, agent m executes action a at state i with probability $\pi^m_{i,a} = \frac{x^m_{i,a}}{\sum_a x^m_{i,a}}$.

Not surprisingly, with the approximation techniques, our MILP-based algorithm cannot guarantee optimal solutions. However, we can prove that, in the worst case, the algorithm suffers a loss of $\sum_k (e + g) \times R^k$ from the optimal total expected reward, where g denotes the maximum error caused by the discretization, which can be reduced by increasing the size of $\mathbb{H}$, and e denotes the maximum error caused by the piecewise linear approximation, which can be reduced by using more points $\overrightarrow{\varrho_i}$ in $\mathbb{A}_h$.

To further improve performance, we integrate the MILP algorithm with hill-climbing search, and iterate MILP and hill-climbing search with gradually increasing approximation accuracy. The details are presented in Algorithm 1.

Algorithm 1 MILP algorithm integrated with hill-climbing search (MILP-HCS)

```
1: Initialize approximation accuracy parameters e, g, and
   randomly choose a joint policy randomSol
2: initSol ← HillClimbSearch(problem, randomSol)
3: repeat
4:    milp ← CreateMILP(problem, e, g)
5:    nearOptimalSol ← SolveMILP(milp, initSol)
6:    sol ← HillClimbSearch(problem, nearOptimalSol)
7:    initSol ← sol
8:    e, g ← ImproveAccuracy(e, g)
9: until computational time uses up or user interrupts
10: return sol
```

4. EMPIRICAL RESULTS

In this section, we give a preliminary empirical evaluation of our algorithm through comparing its performance with two other well-known approaches: the coverage-set algorithm [2] and the random restart JESP algorithm [5].

We evaluate these algorithms on problem instances with two cooperative agents where each autonomous agent has a ordered set of 40 tasks, half of which (randomly chosen) are its individual tasks and the others are its subtasks for joint tasks. The rewards for the individual tasks are evenly distributed in the range $[0, 1]$ and the rewards for the joint tasks are evenly distributed in the range $[0, 3]$. The cooperative agents try to maximize their total expected reward within 20 time steps. The state for each agent is composed of its current task and the remaining time before the deadline. At each state, each agent can choose to execute the current task or to skip it. If the agent chooses to skip, then it moves to the next task immediately. The uncertainty in the problem is in the time required to execute the tasks. In this paper, we assume that, if an agent decides to execute a task, it has 25% probability of accomplishing it within one, two, three, and four time steps respectively.

We ran experiments on a Pentium IV machine, and used *cplex* 9.1 (www.ilog.com) as our MILP solver. The empirical results shown in Figure 2 were collected from 100 randomly generated test problems. Figure 2 illustrates anytime performance of these three algorithms, in which we can see that MILP-HCS is able to find good solutions much faster than the other two algorithms.

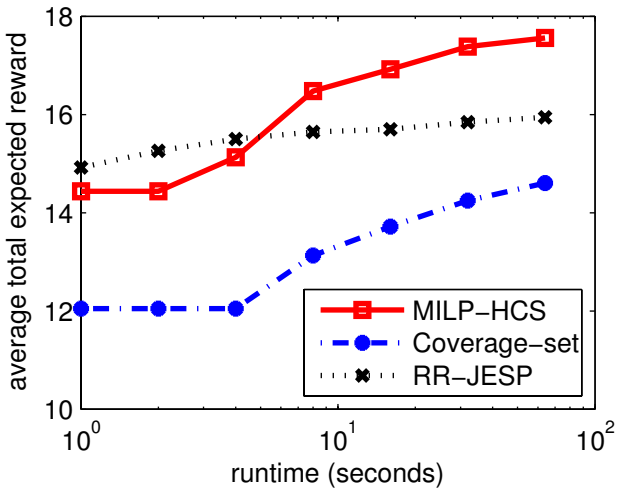


Figure 2: Anytime performance of three algorithms.

5. CONCLUSION

We have described a novel MILP-HCS algorithm for solving transition-independent Dec-MDPs, and shown through experiments that this algorithm is able to find a satisfactory solution much faster than previous approaches. This work links transition-independent Dec-MDPs with mixed integer linear programming, and lays the foundation for further efforts of exploiting established MILP techniques.

6. ACKNOWLEDGEMENTS

This material is based upon work supported in part by the DARPA/IPTO COORDINATORs program and the Air Force Research Laboratory under Contract No. FA8750–05–C–0030. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

7. REFERENCES

[1] E. Altman. *Constrained Markov Decision Processes*. Chapman and HALL/CRC, New York, 1999.

[2] R. Becker, S. Zilberstein, V. R. Lesser, and C. V. Goldman. Solving transition independent decentralized Markov decision processes. *J. Artif. Intell. Res. (JAIR)*, 22:423–455, 2004.

[3] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley & Sons, New York, 1998.

[4] C. V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *J. Artif. Intell. Res. (JAIR)*, 22:143–174, 2004.

[5] R. Nair, M. Tambe, M. Yokoo, D. V. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*, pages 705–711, 2003.

Scalability, security and performance analysis

Performance Analysis of Untraceability Protocols for Mobile Agents Using an Adaptable Framework

Rafał Leszczyna
European Commission
Joint Research Centre
Institute for the Protection and security of the Citizen
Via Enrico Fermi
21020 Ispra (VA), Italy
Rafal.Leszczyna@jrc.it

Janusz Górski
Gdańsk University of Technology
Faculty of Electronics, Telecommunications and Informatics
Department of Software Engineering
G. Narutowicza 11/12
80-952 Gdańsk Poland
jango@pg.gda.pl

ABSTRACT

Recently we had proposed two untraceability protocols for mobile agents and began investigating their quality. We believe that quality evaluation of security protocols should extend a sole validation of their security and cover other quality aspects, primarily their efficiency. Thus after conducting a security analysis, we wanted to complement it with a performance analysis. For this purpose we developed a performance evaluation framework, which, as we realised, with certain adjustments, can be applied to evaluate performance of a whole class of security protocols for mobile agents. This class contains the protocols in which each host on the agent's route is responsible for providing a security function, and, inter alia, includes a big variety of integrity checking schemas and accountability mechanisms for mobile agents such as Lee's Partial Results Authentication Codes or family of Karjoth's et al protocols for protection of computation results of free-roaming agents. Finally, we used the framework to evaluate efficiency of our Untraceability Protocol I. In this paper we present the framework and the results of the evaluation.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; K.4.1 [**Public Policy Issues**]: Privacy; H.3.4 [**Systems and Software**]: Performance evaluation (efficiency and effectiveness)

General Terms

Security, Performance

Keywords

ACM proceedings, mobile agents, security, privacy, anonymity, untraceability, security protocols, performance evaluation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

We proposed two security protocols for untraceable migration of mobile agents [1, 2] which have this advantage over other existing untraceability protocols for agents, that they support agents' autonomy in choosing their route [1, 2]. We performed a basic security analysis of the protocols [1, 2] by checking their resistance to known generic and traffic analysis attacks, and currently we are working on extending this analysis. We also applied the protocols to an e-health service to verify their applicability to real IT services [3].

Although in [2] we had presented some implementation suggestions aiming at improving performance of the protocols, the general question of their performance was left open. In this paper we present the results of our recent work aiming at comprehensive coverage of the issue.

To evaluate performance of our protocols we developed a framework. The framework is composed of three modules: an analytic study, related to computation of time complexities; a performance benchmark based on standard performance testing; and a protocols' target environment overhead estimation. The objective of the first part was to obtain the indicator of the protocols' performance, independent of a concrete computer architecture, agent environment and implementation details [4]. The empirical part aims at complementing the analytical results with validation of the simplifying assumptions of the analytical model which was introduced to derive the final complexity equations; and providing a direct view on the protocols' efficiency in a real context. The third module is to obtain a picture of the overhead which will be brought in by the protocols to their target architecture.

In the next sections we will describe the framework and we will present the results of evaluation of our Untraceability Protocol I performed on its basis. Because the description of the framework often refers to our real case, we decided to precede it with a brief presentation of the protocols.

2. UNTRACEABILITY PROTOCOLS

Anonymity is the property that ensures that a user may use a resource or a service (the items of interest) without disclosing his/her identity [5]. *Untraceability* is a subclass of communication anonymity [6] assuring that the identities of the message sender and the receiver can not be inferred (and linked) by tracing the message. In case of mobile agents this

means that the agent's owner and the target (a particular location or just a statement of the goal) can not be identified by tracing the agent.

Anonymity plays a crucial role for various activities conducted in the Internet. Gülcü et al [7] describe four categories of the activities[1]. These are: discussion of sensitive and personal issues, information searches, freedom of speech in intolerant environments and polling/surveying. For some of these uses, the anonymous access was an enabler and contributed to increasing the popularity of the Internet.

To support anonymity in agent environments we proposed an untraceability protocol for mobile agents – the *Protocol I* which has this advantage over the solutions of Westhoff et al and Enzmann et al [8, 9] that it does not constrain agents' autonomy of choosing a migration path.

In brief, the concept of the protocol is that an agent, while migrating, at each visited host encrypts the identifier of the last visited host (using the public key of the present host) and puts it in the LIFO queue stored in its state. Then, after achieving a goal, when the agent wishes to come back to its base host, it uses the queue to find its way back. Down the route back the identifiers are subsequently decrypted using each host's private key [2].

We investigated security [1] of the protocol and figured out that the protocol, while being effective, is prone to the costly, nonetheless possible to perform – *cordoning-off attack* – i.e. the attack in which the attacker compromises all hosts surrounding the source one [1]. Consequently we introduced the *Protocol II* which eliminates this vulnerability at the cost of addition of some computation workload to the source host and restriction of the agent's autonomy at the beginning of its route [3]. Hereby we gave choice to the users: they can use more effective but slightly less secure Protocol I, or choose the Protocol II, to receive stronger protection.

Because we do not refer to the second protocol in this article, we decided to skip its details. An interested reader may find them in the [3].

3. FRAMEWORK

The framework consist of three modules: an analytic study, a performance benchmark and a target environment overhead estimation.

The first module, aiming at estimation of performance of protocols which abstracts of any underlying technological architecture [4], focuses on calculation of time complexity of a protocol as a function of the length of the agent's route.

The second, provides a practical view at the performance of the protocols, allowing to confront the analytic results with the figures obtained during the tests performed in a testing architecture.

The third module gives an image of the overhead brought in by the protocols to their target architecture.

We developed the framework to evaluate performance of our untraceability protocols but we believe it can be applied to the whole class of security protocols for mobile agents.

This class contains the protocols in which each host on agent's route is responsible for providing a security function. For example in case of our Protocol I, each host has to compute the hash value of a concatenation of the preceding container identifier $\mathsf{ID}_{\mathrm{m}-1}$, the own identifier and the successor identifier; then it has to encrypt the $\mathsf{ID}_{\mathrm{m}-1}$, the hash value and a nonce with its secret key; so the agent could add it to the end of its queue of encrypted platform identifiers [3].

Another example is the Simple Partial Results Authentication Codes (Simple PRAC) schema of Lee et al [10]. In the schema a digital signature (and optionally a hash function) has to be computed at each host to assure that the partial results collected at the host are not-repudiable and undeniable.

To the group also belong other variations of Yee's PRAC such as MAC-based PRAC with One-Way Functions [10], and techniques presented by Karjoth et al [11] as well as the Vigna's tracing schema [12]. Apparently most of the methods aiming at detection of tampering with data gathered by a mobile agent will fall into the category since they are based on the similar concepts.

Thus, our framework, although primarily targeting our untraceability protocols, eventually may become very useful to evaluate performance of various other security protocols for mobile agents.

4. ANALYTIC STUDY

The analytic study focuses on calculation and comparing time complexities of security protocol -enabled (protected) and -disabled (plain) agent's migrations. These complexities are calculated as sums of partial times spent on each involved host. Because specific protocols may differ from each other with relation to numbers of these hosts between protected and plain migration (e.g. in our case untraceable migration needs twice as many nodes as plain migration) or other than Java environments might be required to serve as a reference, the study should be performed individually for a particular protocol. For this reason, we do not provide any common template, yet instead, we present derivations corresponding to our untraceability protocol, which we encourage to use as a reference point.

Since Java is the most popular environment for implementation of agent platforms [13] the Java-founded migration mechanisms served as a reference for our analysis. Relying on our experience, we claim that the general code-transfer principles of Java are equally applicable to other environments, and where they are not, they differences are just slight.

In the next subsection we present the reasoning for the protected (untraceable) migration and then the results of adequate deduction applied to the plain migration. Comparing the results will allow us to derive the equation for the overhead induced by employing our security protocol which we present in section 4.3. Implications of this specific formula we will discuss in further part of the paper.

In this section, the term 'time' is used interchangeably with 'time complexity'.

4.1 Protected migration

During the initialisation of its migration, the Untraceable Agent (UA) is given a route to be traversed in order to achieve its goal. Let N be a length of the route. The total migration of UA is composed of $2N$ steps since, according to the Protocol, during the actual migration, the given route is covered twice – first when the agent performs its task, and second when the agent is coming back to the source location. If time of each step performance is denoted as $\dot{t}[i]$ then the

[1]This is important to note that the authors don't claim this set to be exhaustive.

total migration time of UA $\ddot{T}[N]$ is specified as:

$$\ddot{T}[N] = \sum_{i=1}^{2N} \ddot{t}[i] \ . \tag{1}$$

A single migration step roughly involves – suspension of the execution, then serialization of the agent's state and code, and finally sending the resulting data package to the new location where it is de-serialized and the execution resumes. This implies that the time $\ddot{t}[n]$ is the combination of the following times:

$\ddot{t}_T[n]$ – time of UA's transfer (T) to the container n (from the container $n-1$) through the communication channel,

$\ddot{t}_S[n]$ – time of UA's serialization and de-serialization (S) on the container n,

$\ddot{t}_{TR}[n]$ – time of UA's task realization (TR) on the container n,

$\ddot{t}_{UO}$ – time of UA's untraceability operations (UO) on one container.

The time $\ddot{t}_{UO}$ does not depend on N and is fixed since the untraceability operations always operate on fixed amount of data. The other times can vary: the agent can perform a different task on each location, the agent transfer time depends on characteristics of a transmission medium and the agent (de-)serialization may last longer or shorter proportionally to size of the agent's state. While the first two times are the N-independent, the third includes a component which depends on N – the information about agent's route is maintained in the agent's state.

Because the migration times while heading to the target location and returning to the source can differ significantly, the time $\ddot{t}[i]$ distinguishes between them as follows:

$$\ddot{t}[n] = \begin{cases} \overrightarrow{t}_{UA}[n], & \text{if } 1 \leq n < N; \\ \overleftarrow{t}_{UA}[n-N], & \text{if } N < n \leq 2N; \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

where $\overrightarrow{t}_{UA}[n]$ denotes the step performance while going ahead and $\overleftarrow{t}_{UA}[n]$ represents step performance while going backwards.

The time $\ddot{t}_S[n]$ is a combination of the two components:

$$\ddot{t}_S[n] = \ddot{t}_{RS}[n] + \ddot{t}_{RESS}[n] \ , \tag{3}$$

where $\ddot{t}_{RS}[n]$ is the time of UA's route serialization (RS) and $\ddot{t}_{RESS}[n]$ – the time of UA's route excluded state serialization (RESS).

The time $\ddot{t}_{RS}[n]$ can be expressed by the times of atomic serialization operations:

t_{LS} – time of one location serialization (LS),

t_{ELS} – time of one encrypted location serialization (ELS),

– and distinguished between $\overrightarrow{t}_{UARS}[n]$ – the time of route serialization (RS) while UA is heading the target location and $\overleftarrow{t}_{UARS}[n]$ – the time of route serialization (RS) while the UA is coming back to the source:

$$\overrightarrow{t}_{UARS}[n] = [N-n]t_{LS} + nt_{ELS} \ , \tag{4}$$

$$\overleftarrow{t}_{UARS}[n] = [N-n]t_{ELS} \ . \tag{5}$$

Consequently:

$$\begin{aligned} \overrightarrow{t}_{UA}[n] = \ddot{t}_{TR}[n] + \ddot{t}_{RESS}[n] + (N-n)t_{LS} \\ + nt_{ELS} + \ddot{t}_{UO} + \ddot{t}_T[n] \ , \end{aligned} \tag{6}$$

$$\overleftarrow{t}_{UA}[n] = \ddot{t}_{RESS}[n] + (N-n)t_{ELS} + \ddot{t}_{UO} + \ddot{t}_T[n] \ , \tag{7}$$

$$\ddot{T}[N] = \sum_{i=1}^{N} \overrightarrow{t}^{UA}[i] + \sum_{i=1}^{N} \overleftarrow{t}^{UA}[i] \ , \tag{8}$$

and the final expression for $\ddot{T}[N]$ is:

$$\begin{aligned} \ddot{T}[N] = N^2 t_{ELS} + 2N\ddot{t}_{UO} + \frac{N(N-1)}{2} t_{LS} \\ + \sum_{i=1}^{N} \ddot{t}_{TR}[i] + \sum_{i=1}^{2N} \ddot{t}_{RESS}[i] + \sum_{i=1}^{2N} \ddot{t}_T[i] \ . \end{aligned} \tag{9}$$

4.2 Plain migration

After applying the analogous reasoning and notation to the Traceable Agent (TA), and keeping in mind that TA needs only to cover a one-way route (N locations), after which it moves back directly to the source location (1 migration step), the following equations describe the relevant times for TA:

$$\dot{T}[N] = \sum_{i=1}^{N} \overrightarrow{t}_{TA}[i] + \overleftarrow{t}_{TA} \ , \tag{10}$$

$$\overrightarrow{t}_{TA}[n] = \dot{t}_{TR}[n] + \dot{t}_{RESS}[n] + (N-n+1)t_{LS} + \ddot{t}_T[n] \ , \tag{11}$$

$$\overleftarrow{t}_{TA}[n] = \dot{t}_{RESS}[n] + \ddot{t}_T[n] \ . \tag{12}$$

And from the above we can convert (10) into:

$$\begin{aligned} \dot{T}[N] = \frac{N(N+1)}{2} t_{LS} + \sum_{i=1}^{N} \dot{t}_{TR}[i] \\ + \sum_{i=1}^{N+1} \dot{t}_{RESS}[i] + \sum_{i=1}^{N+1} \dot{t}_T[i] \end{aligned} \tag{13}$$

4.3 Overhead

The overhead introduced by the untraceability protocol can be calculated as a ratio:

$$\frac{\ddot{T}[N] - \dot{T}[N]}{\dot{T}[N]} \cdot 100\% \ . \tag{14}$$

where the value of $\ddot{T}[N] - \dot{T}[N]$ is given as following:

$$\begin{aligned} \ddot{T}[N] - \dot{T}[N] = N^2 t_{ELS} + 2N\ddot{t}_{UO} \\ + (N-1)\ddot{t}_{RESS} + \sum_{i=1}^{N-1} \dot{t}_T[i] - Nt_{LS} \end{aligned} \tag{15}$$

It is assumed that the parameters not related to untraceability are equal, i.e. n: $\dot{t}_{TR}[n] = \ddot{t}_{TR}[n]$ and $\dot{t}_T[n] = \ddot{t}_T[n]$ and $\dot{t}_{RESS}[n] = \ddot{t}_{RESS}[n] = t_{RESS} = const.$

5. PERFORMANCE BENCHMARK

The testing module is composed of a testing architecture and a series of tests to be performed upon it.

5.1 Testing architecture

To obtain meaningful test results, tests should be performed either directly in the target environment or in an environment that is sufficiently close to the target one. Since at security protocols' design and validation stages authors of protocols usually do not know the target environment, and yet because security protocols are usually designed to cover various areas of problems (both cases applied to us), only the second option remains available.

To reach the feature of sufficient proximity between testing and possible target environments, when designing our testing architecture, we paid particular attention to its flexibility and configurability. In other words, we aimed at providing an architecture, which with one setting of 'switches' would be similar to one predicted target environment, while with another setting it would become close to another target environment.

For this reason, firstly we had to identify the characteristics of computational environments which have the strongest impact on performance of the environments.

Each agent environment is composed of three layers: the agents layer, comprising agents of any type; the underlying containers layer – a middleware allowing agents' operation; and the hosts layer which aggregates hosts interconnected via a computer network.

When extracting the characteristics of interest we examined each of the three layers respectively. Results of the study are listed in the table 1[2].

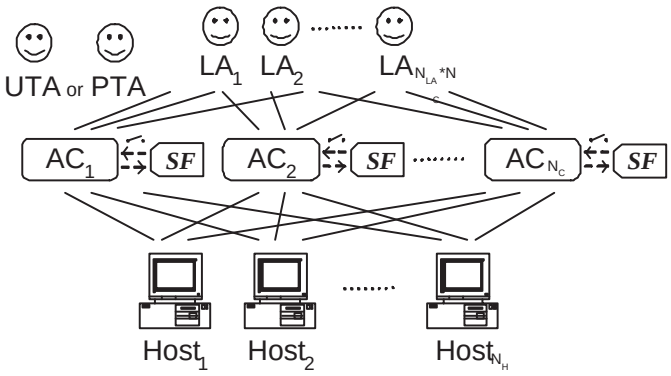


Figure 1: Benchmarks architecture.

After receiving the above characteristics we developed a three-layered testing architecture (Figure 1) in which we can control a subset of the characteristics. The architecture is realised in Java Agent DEvelopment Framework (JADE – `http://jade.tilab.com/`) since it appears to be most popular agent middleware [13].

In our architecture, the agents layer is composed of Load Agents (LAs), amount of which is flexibly adjustable. As the name indicates, the main aim of LAs is to load the environment and consume its resources. This must be done quite similarly to as original agents would do. Thus, each LA actively operates in the system – it migrates and 'realises tasks'. The task realisation is simulated by stopping LA on a container for a random period in time. Despite the fact that the container resources are not consumed directly in sense of being utilised by a working agent, the LA's presence on a container requires reservation of an execution space for LA. The stay-time is chosen with equal probability from the range between 10 milliseconds and 10 seconds – corresponding to various kinds of agent tasks. Another type of diversity of agents' behaviours, is simulated by introducing two types of LAs – Protected LAs (PLAs) and Unprotected LAs (ULAs), what allows to examine the behaviour of the environment depending on the level of use of protection (usually only a certain subset of agents requires protection).

Since our untraceability protocols do not involve any agent intercommunication and a user-control we decided not to include these characteristics of agents layer into our architecture. We also did not examine nuances of implementation since our protocols straightforwardly follow specifications and do not leave a space for performance-focused improvements.

In the containers layer we can configure a number and deployment of containers and to enable or disable protection. Originally we used untraceability but it may be conveniently replaced by any other measure by means of JADE add-ons. We can also enable another services, which could consume (or provide) additional resources of environment.

The hosts layer is very independent of the actual implementation of our architecture and can be flexibly configured according to the hardware resources available to testers.

In such simulated environment test agents are run. Each such agent can be either an Unprotected Test Agent (UTA) or a Protected Test Agent (PTA). They are identical except that the first one utilises and the second one does not utilise the untraceability service.

During a test each test agent has to follow the route of N_{IC} length N_R times. The UTAs and PTAs are following the same routes. The time of passing every round is measured and recorded as well as the time of passing all N_R rounds. The precision of time measurements is 10 milliseconds [3].

5.2 Tests

The objectives of the tests were to:

- examine if and to which extent enabling protection affects performance of agents which do not use it,
- compare performance of protected agents (PAs) and unprotected agents (UAs),
- analyse performance of PAs as a function of length of the agent route and the agent environment load.

While comparing performance of UAs and PAs the two most common agent migration scenarios were taken into consideration[4]:

Agent Migration Scenario I – the agent has its goal formulated in general terms so the it has to traverse mul-

[2]It is important to note that the authors do not claim this set to be exhaustive.

[3]We have used java `System.currentTime()` – being the most precise of the program execution measuring methods which do not require hardware devices [14].

[4]In figures we use the notation: N_C – total number of containers, N_{IC} – number of Intermediate Containers (IC).

layer	characteristics
Agents Layer	number of agents present in the environment, diversity of agents behaviours (including agent relocation), agents intercommunication, agents implementation, agents resources, agents control (autonomous agents, user-controlled agents etc)
Containers Layer	number of containers, containers deployment, services resources
Hosts Layer	number of hosts, hosts efficiency, inter-host communication (communication protocols, communication medium)

Table 1: Characteristics of agent environments of the strongest influence on performance of the environments.

tiple locations in order to satisfy the goal (e.g. web crawling)

Agent Migration Scenario II – the agent goal is to visit its target location and to perform its task there (e.g. bandwidth conservation scenarios – in which an agent is sent to a remote host in order to reduce multi-host communication)

Figures 2 – 5 illustrate the scenarios applied to our untraceability protocol.

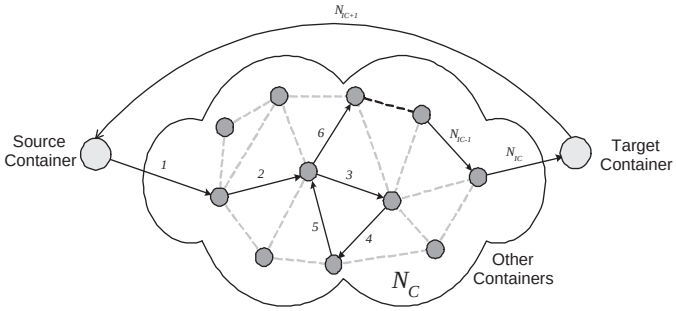


Figure 2: Agent Migration Scenario I realized by TA: TA traverses N_{IC} containers and in N_{IC+1} step returns to the source container.

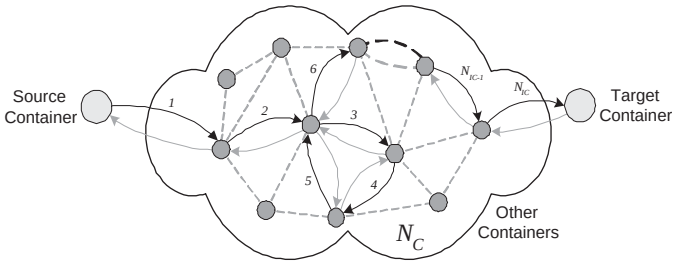


Figure 3: Agent Migration Scenario I realized by UA: UA traverses N_{IC} containers and then has to cover the traversed route again (in reversed order) to return to the source container.

In each case, after achieving the goal the agent returns to the source container.

6. TARGET ENVIRONMENT OVERHEAD ESTIMATION

As a third module of the evaluation framework we propose estimation of overhead brought in by a security protocol to the foreseen target environment.

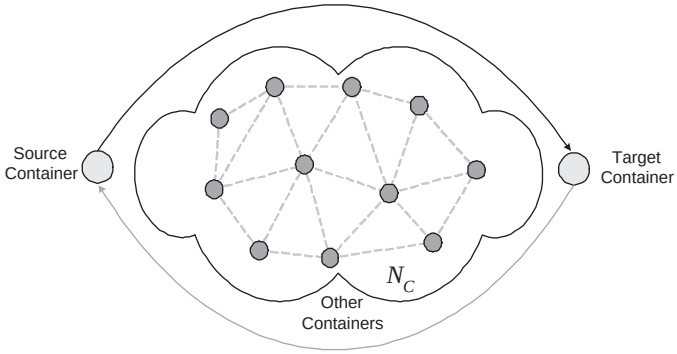


Figure 4: Agent Migration Scenario II realized by TA: TA simply 'jumps" directly to the target container and after realizing its task returns to the source container.

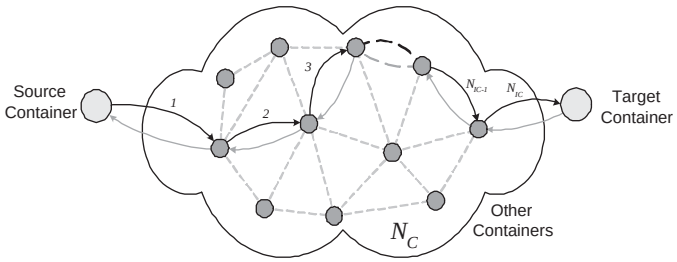


Figure 5: Agent Migration Scenario II realized by UA: Before reaching the target container the UA has to cover an intermediate route of a length N_{IC} in order to hide a migration trace. This route must be traversed again by the UA when returning to the source container.

At this stage it is necessary to obtain the specification of the target environment allowing the most precise determination of functional types of agents required in the environment and their quantitative proportions. With these figures it is possible to assess the amount of the agents actually utilising the security protocol, and on its ground, to evaluate the real overhead which the security protocol will bring to the environment.

We will illustrate this on the example of our protocols.

One of the applications where users anonymity (so untraceability) plays crucial role is anonymous e-health counselling. Anonymous e-health counselling is a service allowing IT-based provision of health related advices on an anonymous basis. Another words, it provides all functionalities for performing the scenarios where a patient using an IT system asks for a health advice, yet wishes to do it anonymously (because, for example, they suffer from an embar-

rassing disease or from an addiction) [15].

This application, which for the first sight seems to be quite dedicated, in reality it represents a large area of (often used) applications in which the schema of agent-based accessing multiple knowledge sources in order to obtain a comprehensive information is implemented [3].

Thus we developed an experimental application providing the e-health counselling service [3] which takes advantage of untraceable agents – to provide ourselves the opportunity to observe how our protocols behave in a real settlement.

And the e-health environment we treated as the target environment when estimating the level of the total untraceability overhead brought in by enabling the untraceability service.

The target health environment involves many stakeholders participating in the process of medical service delivery: patients, health professionals, pharmacists, dietitians etc. In the technological e-health platform, a stakeholder is represented by an agent, – its *Virtual Ego*. Apart of Virtual Ego agents, various other agents are present in the platform [3]: Knowledge Discovery Agents (KDA), Information Agents (IA) and a compound group of Auxiliary Agents (AA) – encompassing different types of agents responsible for performing supportive tasks (Figure 6).

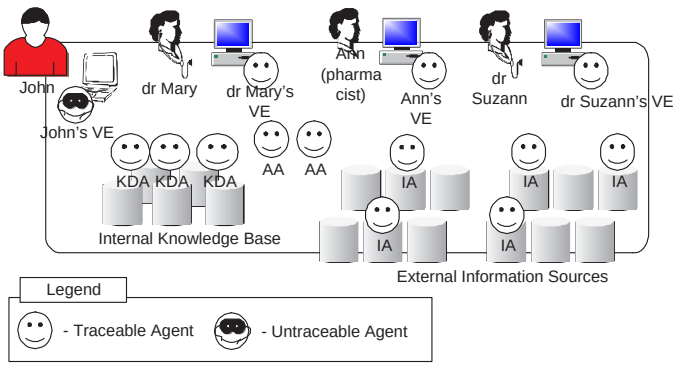


Figure 6: Mobile agent based health platform.

For such platform we estimated the proportion of active (running) agents in the system as a function of N_{PU} – the number of patients using the system at a particular moment ($N_{PU} \leq N_P$, where N_P is the number of all agents registered in the system). It is worth to note that besides these active agents also temporarily suspended agents exist in the system, which wait for dispositions of their owners. Our prognoses are based on the report of Royal College of Physicians of London [16] saying that average number of practising physicians per 1000 inhabitants in Europe equals 3.4 and the surveys conveyed under the International Network Health Policy & Reform project indicating average 6.19 health consultations per person taking into account ten European countries [17]; and our experience in design of IT systems.

The results (assuming $N_{PU} = 10000$) are presented in the table 2.

With the figures we estimated the overhead. The results of the estimation we present in the next section.

7. EVALUATION RESULTS

7.1 Analytic study

The equation (15) for the overhead introduced by the untraceability protocol, indicates polynomial (quadratic) dependence of $\ddot{T}[N] - \dot{T}[N]$ on N. The Untraceability-Related Component (URC) ($N^2 t_{ELS} + 2N\ddot{t}_{UO}$ – with a higher significance of the first component of the sum) grows faster than the Migration-Related Component (MRC) ($(N-1)\ddot{t}_{RESS} + \sum_{i=1}^{N-1} \dot{t}_T[i]$). However the real value of t_{ELS} is at least four orders of magnitude smaller than the value of MRC, which means that not until N reaches very high values ($N \geq 10000$), the influence of URC becomes visible. In practical agent applications, agents scarcely (if ever) have to cover routes of such lengths. It must be also remembered that assigning a route of that large length to a traceable agent, leads to dominance (because of the quadratic growing with reference to location serialization) of the route-storage component as well.

7.2 Performance benchmarks

The following benchmarks were performed basing upon our testing architecture:

Test I – Comparing the non-protected agent (the traceable agent – TA) performance in the untraceability enabled and untraceability disabled environments, $N_{IC} = N_C = 100$, the random stay-time is turned off,

Test II – Comparing migration times of the traceable agent (TA) and the untraceable agent (UA) as the function of N_{IC}, $N_{IC} = NC \in 20, 40, 60, 80, 100, 120, 140$ (refers to Migration Scenario I),

Test III – UA performance dependent on N_{IC}, $1 \leq N_{IC} \leq 64$, $N_C = 150$ (refers to Migration Scenario II),

Test IV – Comparing UA performance in variably loaded environments. Due to identified problems with JADE this test was performed for $N_C = 60$ and only three $N_L A$ values – $N_L A \in 60, 120, 180$.

At the current stage of our research all benchmarks were performed on one host[5], but we consider extending them to a higher number of hosts in the future.

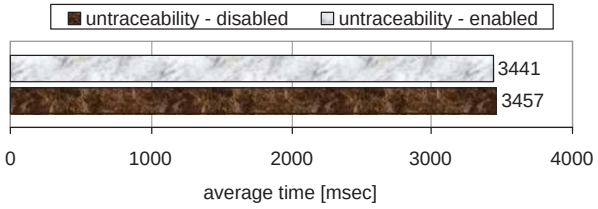


Figure 7: Test I – Comparing TA performance in untraceability enabled and untraceability disabled environment, $N_{IC} = N_C = 100$.

The results of Test I (see Fig. 7) empirically prove that enabling the untraceability service does not influence the per-

[5]Intel®Pentium®4, 3 GHz, 1.5 GB RAM. Java™2 Runtime Environment, Standard Edition (build 1.4.2_01-b06), Java HotSpot™Client VM (build 1.4.2_01-b06, mixed mode) – run with default settings.

active patients	patients VE	health physicians VE	KDA	IA	AA
10000	10000	34	5000	5000	30000

Table 2: Numbers of different mobile agents running (active) in the health system for 10000 active patients.

formance of traceable agents[6]. This is not surprising since the mobile agents do not use any functions from the service. However the question was if the fact that the service is enabled on the containers (and takes some resources) might affect the performance.

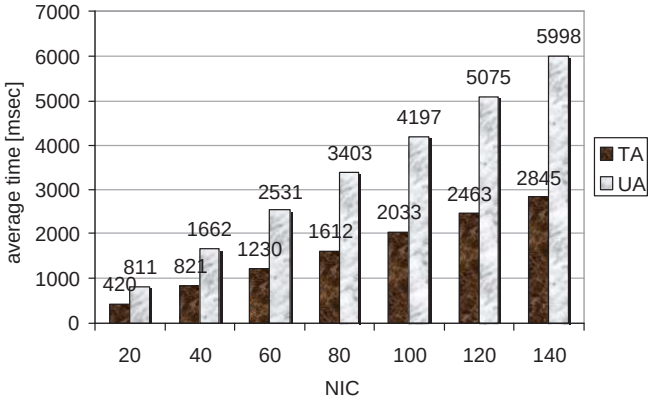


Figure 8: Test II – Comparing migration times of TA and UA dependent on N_{IC} (refers to Migration Scenario I).

The results of Test II (Figure 8) confirm the intuition, that for realistic values on N_{IC}, the dominant factor influencing the performance is agent migration – the approximately linear growth of time consumption in dependence on N – practically the same for TA and UA (the linearity coefficient is roughly equal to 1 – to cover two times longer route agent needs approximately two times more of time) – as it could be inferred from the estimation of complexity. Consequently, the dominant role of the migration factor affected the overhead, which oscillates on the level of around 100% – UA must again cover the already traversed route whereas TA just 'jumps' from the target location back to the source.

The meaningful result is obtained from Test III – when sending UA the user has to be conscious that its performance is directly proportional to the length of untraceability route (Figure 9).

Test IV confirms the intuition that if the number of the load agents approaches the limit resulting from the capacity of the agent environment, the performance falls significantly (as the result of the shortage of the operational resources) (Figure 9).

7.3 Target Environment Estimated Overhead

Assuming that every tenth patient want's to take advantage of a medical service untraceably, only 1000 of total 50034 active agents will introduce the overhead related to untraceability to the platform. Having in mind that each

[6]The slight difference of values (this time, surprisingly –) in favour of the untraceability enabled environment is in the range of standard deviation which is equal to 110 [ms].

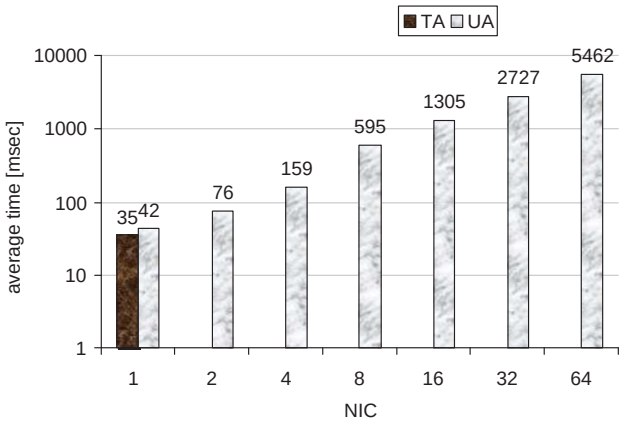


Figure 9: Test III – UA performance dependent on N_{IC} (refers to Migration Scenario II).

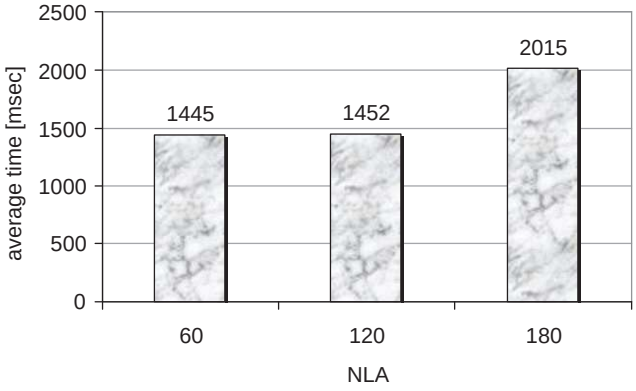


Figure 10: Test IV – Comparing UA performance in variably loaded environments.

individual UA introduces around 100% of the TA overhead, and representing a performance of each TA as 1 Unit of Performance (UP), for 50034 UP's – related to the presence of 50034 agents in the system – the 1000 UA's will induce additional 1000 UP's. Consequently the total overhead (we must underline that the overhead is related solely to a time of migration and not with any other agent activities) will be almost 2%.

8. SUMMARY

The paper presented a performance evaluation framework which although originally designed for our untraceability protocols, may be applied to a much wider category of security protocols for mobile agents. This category include all the protocols where during migration of agent, a security function is computed at each host passed. The framework is based on three modules. The first, analytic module aims

at deriving a mathematic formulae for the overhead imposed by the employment of the protection, and involves computation of time complexities of standard and protected agents. In the empirical part we provide a testing architecture and a series of tests to be performed upon it, in order to obtain 'real' performance characteristics of protected and unprotected agents. The third module requires forecasting the structure of the environment to which the protection will be added and it results in estimation of the overall overhead brought in by the protection.

With the architecture, we carried out the evaluation of our Untraceability Protocol I. We derived the universal (not tied to any implementation) symbolic equation for the overhead as a function of the length of the agent's route. Then we performed the tests, results of which confirm the analytical estimates and provide more concrete data related to the Java enabled agent platforms. The general result is that untraceability cost (in terms of time overhead resulting from traversing a given route) is twice as much as the cost without untraceability. A good news is that this cost does not depend on the tasks performed by the agent. It should be also remembered that the real agent applications would likely be composed of both – traceable and untraceable – agents, so the resulting total overhead would be relatively small (see Section 7.3).

9. REFERENCES

[1] Rafał Leszczyna and Janusz Górski. Untraceability of mobile agents. Technical report, European Commission, Joint Research Centre, Institute for the Protection and security of the Citizen, December 2004.

[2] Rafał Leszczyna and Janusz Górski. Untraceability of mobile agents. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '05)*, volume 3, pages 1233–1234, Utrecht, the Netherlands, July 2005.

[3] Rafał Leszczyna. The solution for anonymous access of it services and its application to e-health counselling. In *Proceedings of the 1st 2005 IEEE International Conference on Technologies for Homeland Security and Safety (TEHOSS '05)*, volume 1, pages 161–170, Gdańsk, Poland, September 2005. Gdańsk University of Technology.

[4] Marek Kubale. *Introduction to computational complexity and algorithmic graph coloring.* Gdańsk Scientific Society, 1998.

[5] National Institute of Standards and Technology (NIST). *Common Criteria for Information Technology Security Evaluation - Part 2: Security Funtional Requirements.* U.S. Government Printing Office, 1998.

[6] Andreas Pfitzmann and Marit Hansen. Anonymity, unobservability, and pseudonymity - a proposal for terminology. draft v0.23. August 2005.

[7] Ceki Gülcü and Gene Tsudik. Mixing email with Babel. In *Proceedings of the 1996 Symposium on Network and Distributed System Security (SNDSS '96)*, page 2. IEEE Computer Society, 1996.

[8] Dirk Westhoff, II Markus Schneider, Claus Unger, and Firoz Kaderali. Protecting a mobile agent's route against collusions. In *Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography*, pages 215–225. Springer-Verlag, 2000.

[9] Matthias Enzmann, Thomas Kunz, and Markus Schneider. Using mobile agents for privacy amplification in the trade with tangible goods. In *6th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI)*, volume IV, Orlando, Florida, USA, July 2002.

[10] Bennet S. Yee. A sanctuary for mobile agents. In *Secure Internet Programming*, pages 261–273, 1999. `citeseer.ist.psu.edu/article/yee97sanctuary.html`.

[11] Günter Karjoth, N. Asokan, and C. Gülcü;. Protecting the computation results of free-roaming agents. In *MA '98: Proceedings of the Second International Workshop on Mobile Agents*, pages 195–207, London, UK, 1999. Springer-Verlag.

[12] G. Vigna. Protecting mobile agents through tracing. In *Third Workshop on Mobile Object Systems*, 1997. Available at `https://citeseer.ist.psu.edu/vigna97protecting.html`.

[13] Rafał Leszczyna. Evaluation of agent platforms. Technical report, European Commission, Joint Research Centre, Institute for the Protection and security of the Citizen, Ispra, Italy, June 2004.

[14] David B. Stewart. Measuring execution time and real-time performance. In *Embedded Systems Conference*, San Francisco, April 2001. Embedded Research Solutions. `http://www.embedded-zone.com/Publications/mezexec.pdf`.

[15] EU IST-2002-507591 PRIME. Requirements version 0 part 3: Application requirements.

[16] Royal College of Physicians of London. European working time directive: European dimensions. Internet, November 2002. Available at `http://www.rcplondon.ac.uk/college/statements/doc_ewtd_european.asp`. Last accessed: October 2005.

[17] International Network Health Policy & Reform. Health policy monitor: Country facts. Internet, April 2005. Available at `http://www.healthpolicymonitor.org/en/index.html`. Last accessed: October 2005.

Scalable and Reliable Data Delivery in Mobile Ad Hoc Sensor Networks

Bin Yu, Paul Scerri, and Katia Sycara
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{byu, pscerri, katia}@cs.cmu.edu

Yang Xu, Michael Lewis
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA 15260, USA
{xuy,ml}@sis.pitt.edu

ABSTRACT

This paper studies scalable data delivery algorithms in mobile ad hoc sensor networks with node and link failures. Many algorithms have been developed for data delivery and fusion in static microsensor networks, but most of them are not appropriate for mobile sensor networks due to their heavy traffic and long latency. In this paper we propose an efficient and robust data delivery algorithm for distributed data fusion in mobile ad hoc sensor networks, where each node controls its data flows and learns routing decisions solely based on their local knowledge. We analyze the localized algorithm in a formal model and validate our model using simulations. The experiments indicate that controlled data delivery processes significantly increase the probability of relevant data being fused in the network even with limited local knowledge of each node and relatively small hops of data delivery.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Distributed Artificial Intelligence*

General Terms

Algorithms, Reliability, Experimentation,

Keywords

sensor data fusion, data delivery, reinforcement learning

1. INTRODUCTION

The mobile ad hoc sensor networks of the near future are envisioned to consist of hundreds of UAVs (unmanned aerial vehicles) or robots. These networked mobile sensors play strong roles in military and civilian operations, e.g., battlefield surveillance, disaster search and rescue [9, 15, 13]. One phenomenon in mobile sensor networks is that the raw data from each sensor usually has low confidence. The raw data cannot be used directly for team coordination and has to be fused with other relevant data [11]. Various statistical inference techniques have been studied for distributed data fusion in mobile sensor networks, but these approaches do not have any control on the data flows [12]. Uncontrolled data flows may cause large amount of conflicting plans and have severe effects on the performance of the whole network when mobile sensors work together to conduct high-level tasks. Thus, how to optimize communication strategies for team coordination is an important problem for mobile ad hoc sensor networks [18]

Many protocols have been developed for data delivery and fusion in microsensor networks with small and battery-powered "motes", where a node, called a sink, attempts to gather data from a number of data sources [21]. For example, in directed diffusion, the sink floods a query toward the sensor nodes in a specific area [7]. The source nodes within the specific area then send data back to the sink along the best path. In other words, a source node will not send any data back to the sink until it receives the query. For this reason, these routing protocols are called *reactive protocols*.

Reactive protocols are mainly designed for microsensor networks, where the sensors are stationary after deployment. They are not appropriate for mobile sensor networks for the following two reasons:

- Heavy traffic: Since locations of collected data are not correlated with existing positions of mobile sensors, effective routing of either query or data is usually infeasible. Sinks have to flood the queries to the whole network when they need the data each time for a specific area. This could overload one or more sinks and cause a lot of traffic in the network.

- Long latency: Sinks usually do not know when source nodes will have the data in mobile sensor networks. If the source nodes follow reactive protocols, they have to wait for the queries from sinks to send the data. This causes long latency and cannot meet the requirements of time-critical missions in many defense and civilian applications for mobile sensor networks [2].

In this paper we present a scalable and reliable approach to data delivery in mobile ad hoc sensor networks with failures. While failures could happen at either team coordination level or data communication level, our approach focuses on communication failures for sensor data delivery among

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

mobile sensors [8]. We assume these mobile sensors such as UAVs are "blind" in some sense and they cannot monitor the activities of other sensors directly.

In our approach there is no querying process and each node proactively forwards the data to one of its neighbors. The constraint here is that nodes have no global knowledge of network topology. Each node has to intelligently deliver data solely based on the knowledge about itself and its neighbors. Moreover, the data delivery processes need to be survivable to failures. The challenge is, with various decisions made by individual sensors, how to minimize the communication overhead and maximize the probability of relevant data being fused in the network.

We propose an efficient and failure-resistant localized algorithm over mobile ad hoc networks, where each node learns routing decisions from its history of data delivery via *Q-learning*. We study the process of neighbors selection for each sensor node and the emergence of a spanning tree in the network for reliable data delivery. Moreover, we analyze the probability of relevant data being fused in a random network in a formal model and validate our model using simulations. The experimental results show that controlled data flows significantly increase the probability of relevant data being fused, where the probability is improved by two to five times for the same hop of data propagation.

The rest of this paper is organized as follows. Section 2 summarizes the relevant literature. Section 3 briefly describes the background of mobile ad hoc sensor networks. Section 4 investigates the design of protocols and localized algorithms for data delivery and fusion. Section 5 provides a formal analysis of the data delivery algorithm in mobile sensor networks. Section 6 presents some experimental results for the algorithms. Section 7 discusses the main themes and some directions for future research.

2. RELATED WORK

Mobile sensor networks have been studied by Howard et al. [6], but their work focuses on the coverage problem of a multi-robot system, and does not consider the effective data delivery for fusion. Q-learning and ants algorithms have been studied by [1, 19] for network routing problem. Routing problem is different from ours since the sensor data does not have explicit destination during the data delivery processes. Moreover, we need not consider alternative paths for surplus sensor data. The data propagation paths are deterministic in our approach, while paths in ants algorithms are probabilistic.

Multisensor data fusion has been studied for a long time, where both the data fusion and control algorithms are centralized [3, 4]. We will not discuss the literature on multisensor data fusion since they fall outside of the scope of this paper.

Distributed data fusion in sensor networks has been focused on statistical inference techniques. For example, Niu et al. investigate an optimum local fusion detection threshold for binary hypothesis testing by local sensors [14]; Rosencrantz et al. present a Bayesian technique for decentralized state estimation using particle filters [16]; and Durrant-Whyte et al. consider distributed data fusion as a state estimation problem using information filter (a variant of Kalman filter) [11, 12]. These approaches focus on detection and tracking performance of distributed sensor networks. They do not have any control on the data flows in the network. In this paper we propose an efficient data delivery algorithm for distributed data fusion. Moreover, we study the tradeoff of the efficiency and fusion probability in a network with node and link failures.

3. BACKGROUND

This section provides some background on mobile ad hoc sensor networks.

We assume mobile sensors communicate on a point-to-point basis [17]. A network of these mobile sensors is modeled as a connected undirected graph $G = (V, E)$, where $V = \{a, b, \ldots\}$ is a set of sensor nodes and E consists of edges between any two nodes a and b that can communicate directly. $N(a)$ is the set of a's neighbors and $b \in N(a)$ is any neighbor of node a. Note that neighbors in mobile sensor networks need not be in the proximity and they are neighbors only in the logical sense.

There are two types of communication failures in a mobile ad hoc sensor network: *node failure* (the loss of sensor nodes) and *link failure* (unreliable communication channels). Link failure is temporary and is modelled as a fixed probability for any two nodes in our scenario. Node failure is permanent and cannot be recovered once the node is dead. Following [5], we model the reliability $R_a(t)$ of a node a as a Poisson distribution

$$R_a(t) = e^{-\lambda_a t} \tag{1}$$

where λ_a is the failure rate of the sensor node and t is the time period. $R_a(t)$ captures the probability of not having a failure within the time interval $(0, t)$. $1 - R_a(t)$ describes the probability of a node failure. The higher the failure rate λ_a, the lower the reliability of a sensor node.

Specifically, we consider a UAVs network for target detection, tracking, and classification in the battlefield. We assume there are multiple stationary or moving targets $T_1, T_2, \ldots, T_M$ on the ground. A target $T_K (1 \leq K \leq M)$ may be detected by multiple UAVs at the same time, but each UAV cannot initialize its plans based on its own raw sensor data about the target, e.g., engagement of its missile with a target. The reason is that the raw sensor data from a UAV is uncertain and noisy. Sometimes, a UAV with SAR (Synthetic Aperture Radar) may even confuse friendly targets ($T80$ tanks) with enemy targets ($M1$ tanks). Hence, the low quality sensor data cannot be used directly for high-level plans and has to be delivered to other nodes for fusion in the network [4].

If sensor a detects a target T_K, it will generate an event e_i about target T_K. An event e_i about T_K can be denoted as a tuple $\langle sender, identity, location, TTL, pedigree \rangle$, where

- *sender* is the ID of the sensor that detects the target.
- *identity* is the decision about the target from the sensor. For example, the target could be classified as $\langle T80, 0.4 \rangle$, where $T80$ is the possible vehicle type and 0.4 is the confidence level.
- *location* is the location of the target T_K being detected.
- *TTL (time-to-live)* is the maximal number of hops allowed for the event propagation in the network.
- *pedigree* is the list of nodes event e_i has been visited, denoted as L. Note that pedigree is used to avoid cycles during event propagation.

The processes of sensor data delivery can be described as follows. When sensor a receives or generates an event e_i about T_K, it first tries to fuse e_i with its previous events about target T_K using statistical inference techniques. [1] If the confidence of fused events meets a threshold, sensor node a will stop the propagation of event e_i and become a sink node for relevant events of T_K. Otherwise, sensor a will choose one of its neighbors to continue the delivery unless its TTL is zero. Note that, sensor a will create an information token about target T_K when it becomes a sink for target T_K. Information tokens are then used for high-level team coordination [18].

4. ALGORITHMS

Obviously, an event does not need to visit all sensor nodes to meet with other relevant events. However, visiting too few sensors can lead to low probability of relevant events being fused. Here two events e_i and e_j are relevant if and only if they are referring to the same target, denoted as $rel(e_i, e_j) = true$. The key is how to minimize communication overload and maximize the probability of relevant data being fused. In this section, we discuss the efficiency and robustness of three algorithms: *random walks*, *path reinforcement* algorithm and *path learning* algorithm.

Random walks are a simple data delivery technique in mobile sensor networks. In random walks, sensor node a with event e_i first decides if it will stop the propagation of e_i. If not, sensor node a just forwards the event e_i to one randomly chosen neighbor $b \in N(a)$. Once the neighbor receives the data, it repeats the same process until the event is successfully fused with other relevant events or the TTL of the event reaches zero.

Obviously, random walks are not efficient for data delivery, since they do not consider the relationships of events and just pass events randomly in the network. Hence, we design other two algorithms — path reinforcement algorithm and path learning algorithm.

4.1 Path Reinforcement Algorithm

The intuition behind path reinforcement algorithm is that relevant events are likely to be fused earlier if they will follow the same path after they meet. Specifically, if sensor node a has sent an event e_i to one of its neighbors b before, it will send b any events e_j if $rel(e_i, e_j) = true$.

Figure 1 shows an example of data flows in the network using path reinforcement algorithm. The solid lines correspond to directed communication channels between sensor nodes. The arrows in dashed lines represent information flows of relevant events e_i, e_j, and e_k. Event e_i and e_j meet at node b and then they follow the same path (event e_j reinforces the path $\langle b, c\rangle$). As shown in Figure 1, the three events are fused at node c.

From the figure it is easy to understand why path reinforcement algorithm is more efficient than random walks for data delivery. For example, if event e_i and e_j meet at node b and they continue to walk randomly in the network, there will be a very small chance that they can meet again and be fused with event e_k at node c.

[1]Commonly used frameworks for processing uncertain sensor data fusion are Bayesian inference method and Dempster-Shafer theory, but the details are outside of the scope of this paper. Readers may refer to [3] for more information.

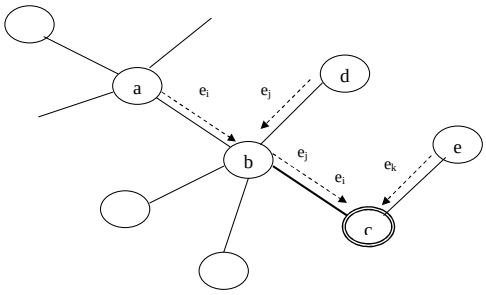


Figure 1: An example of data flows in the network using path reinforcement algorithm, where e_i, e_j, and e_k are three relevant events.

4.2 Path Learning Algorithm

Path reinforcement algorithm is quite efficient when there is no failure in the network. One phenomenon in path reinforcement algorithm is that a sensor never checks if its neighbor has successfully received the data. This will cause severe data loss problem and decrease the probability of relevant data being fused when node or link failures happen.

To remedy to the problem we can allow each node to retransmit the data to another neighbor when a failure is detected. Path reinforcement algorithm with retransmission is robust, but it is not efficient when failures happen. The reason is that sensor a may choose a failed neighbor again when a selects its neighbor randomly. Hence, the key is how to model the reliability of each neighbor, so we can reduce the number of retransmission. Here we describe another algorithm, called path learning algorithm, where each node keeps track of statistics on which neighbor leading to successful data delivery.

Formally, the probability of successfully passing events to a neighbor $b \in N(a)$ at time t is defined as $Q^t(a, b)$ and $0 < Q^t(a, b) < 1$. For any neighbor $b \in N(a)$, $Q^t(a, b)$ is initialized as 0.5 when $t = 0$. Given an event e_i, node a will send e_i to a neighbor $b \in N(a)$ with maximal Q value. Node a will randomly choose a neighbor for e_i if there are any two neighbors b, $c \in N(a)$ and both have maximal Q value at time t, e.g., $Q^t(a, b) = Q^t(a, c)$.

After node a sends out an event e_i to b, a waits for a feedback from b in a period of $\mathcal{T}$. The feedback from b can be defined as

$$feedback = \begin{cases} 1 & \text{if } b \text{ is a sink node for } e_i \\ 0 & \text{if no feedback} \\ max_{z \in N(b)} Q^{t-1}(b, z) & \text{otherwise} \end{cases}$$

where $Q^{t-1}(b, z)$ is the feedback from b and z has the maximal probability from the point of b at time $t - 1$. Here a sink node means b successfully fuses e_i with other relevant events.

The probability to deliver event e_i is updated as

$$Q^t(a, b) = (1 - \alpha)Q^{t-1}(a, b) + \alpha * feedback \qquad (2)$$

where α is the learning rate and $\alpha = 0.5$ in this paper if not specified otherwise. In the case of no response, node a will decrease the value of $Q^t(a, b)$ and choose another neighbor

Algorithms	Efficiency	Robustness
Random walks	bad	bad
Path reinforcement algorithm without retransmission	good	bad
Path reinforcement algorithm with retransmission	average	good
Path learning algorithm	above average	good

Table 1: The comparison of four data delivery algorithms.

to resend the event. The adjusted possibility reduces the chance of choosing b in the future.

Algorithm 1 presents the path learning algorithm, where e_i is an event for target T_K from either node a or other nodes and $\Theta_a(T_K)$ is the set of events about T_K that node a has. Other parameters include: L the pedigree of event e_i and $N(a)$ the set of neighbors of sensor a. The algorithm has two basic parts: (1) deciding if node a will stop propagating event e_i (line 2-4); (2) choosing a reliable neighbor $z \in N(a)$ to deliver e_i and z is not in e_i's pedigree L (line 6-19). Finally, sensor a updates its Q value about neighbor z if $feedback \neq 0$ (line 24).

Algorithm 1 The path learning algorithm

```
1: Suppose sensor a receives or generates an event e_i and
   Θ_a(T_K) is the set of events about T_K at t − 1.
2: Θ_a(T_K) = Θ_a(T_K) ∪ e_i
3: if Θ_a(T_K) has high confidence then
4:    Return success
5: else
6:    repeat
7:       Choose z ∈ N(a) with maximal Q^t(a, z) and z ∉ L
8:       TTL = TTL − 1
9:       L = L ∪ {z}
10:      Send event e_i to neighbor z
11:      if No response from z within time T then
12:         Q^t(a, z) = (1 − α)Q^{t−1}(a, z)
13:         if Q^t(a, z) < ε then
14:            N(a) = N(a) − z
15:         end if
16:         TTL = TTL − 1
17:         L = L − {z}
18:      end if
19:   until feedback ≠ 0 or |N(a)| = φ
20: end if
21: if |N(a)| = φ then
22:    Return fail
23: else
24:    Q^t(a, z) = (1 − α)Q^{t−1}(a, b) + α ∗ feedback
25:    Return success
26: end if
```

Path learning algorithm also helps to maintain the topology of the network when node failures happen. Node a will remove the dead node z from its neighbors when $Q^t_{z \in N(a)}(a, z) < \epsilon = 0.1$ (line 12-13). Intuitively, node a cannot locally distinguish a link failure from a node failure for one time data loss, but it can recognize a neighbor b as a dead node if sensor b continuously fails to deliver data for several times. Also, in order to avoid network partition, node a needs to pass a request in the network when $N(a)$ is small. Any node with small number of neighbors can accept the request and become a neighbor of node a.

4.3 Spanning Tree

From the local view of node a, the feedback from b seems only reinforces the path $\langle a, b\rangle$. But, if we look at the data delivery processes in the whole network, we can find that data flows generate multiple trees, where each tree represents a successful data delivery process for a set of relevant events. The probability $Q^t(a, b)$ serves as the gradient of the path $\langle a, b\rangle$ and controls the direction of data flows.

Once we have multiple trees in the network, there will be two cases for sensor a to deliver an event: (1) if node a is on a tree and is not the root node, it just forwards the event according to the gradient of the path; (2) otherwise, node a just chooses a neighbor with maximal Q value to pass the event. The latter may extend the existing tree or connect the tree with another one.

An important question is how to avoid cycles when a tree connects with another one. A simple way is to check the pedigree of each event before it was sent out. For example, if node x attempts to send event e_i to node a and a already appears in the pedigree of event e_i, x will first reset the gradient between them to 0.5, e.g., $Q^t(x, a) = 0.5$ and then pass e_i to another neighbor y with the highest Q value, where $y \in N(x)$ and $y \neq a$.

Multiple trees will evolve and merge with each other and finally one spanning tree may emerge in the network. The advantage of a spanning tree is that the tree keeps track of the node and link failures and connects most of nodes for reliable data delivery. Note that, in mobile sensor networks, any node on the spanning tree can fuse relevant events and stop the data delivery processes. Hence, our approach is different from other work on building spanning trees in static microsensor networks, e.g., [10], in which nodes on the spanning tree transmit all data to a single sink node and only the sink node has the fusion function.

Table 1 summarizes the properties of four algorithms in terms of efficiency and robustness. Here efficiency denotes the number of messages for data delivery, including both data and feedbacks. The efficiency of path reinforcement algorithm with retransmission is worse than that of path learning algorithm since the former randomly chooses a neighbor, which may be found as a dead node before. Robustness denotes the data loss rates when failures happen. Both random walks and path reinforcement algorithm are vulnerable since they do not have retransmission mechanisms. They do not check if a neighbor has received the data or not.

In the experiments we will evaluate only three of them: random walks, path reinforcement algorithm (without retransmission) and path learning algorithm (only for the spanning tree).

5. ANALYSIS OF FUSION PROBABILITY

In this section we analyze the probability of relevant events being fused in a random network using random walks.

For simplicity, we do not consider link and node failures in the formal model. We assume there are some relevant events e_1, e_2, ..., e_F generated by multiple sensors at the same time. For any sensor node a, the probability that node a knows e_i at hop 0 is $P_a(e_i, 0)$, where $1 \leq i \leq F$.

The probability that sensor node a knows event e_i at hop h as $P_a(e_i, h) \rightarrow [0, 1]$. In a random network with homogeneous sensor nodes, we can model the probability that any sensor node a knows event e_i at hop 0 is

$$P_a(e_i, 0) = 1/|V| \tag{3}$$

where $a \in V$ and $|V|$ is the total number of sensor nodes. Given the relevant events e_1, e_2, ..., e_F, the probability that sensor node a has all these events at hop h is

$$P_a(\Delta, h) = P_a(e_1, h)P_a(e_2, h) \dots P_a(e_F, h) \tag{4}$$

where Δ is the fused result of events e_1, e_2, ..., and e_F.

The probability of all relevant events being fused by any member of the network V is,

$$P_V(\Delta, h) = 1 - ((1 - P_a(\Delta, h))(1 - P_b(\Delta, h)) \dots) \tag{5}$$

where $P_V(\Delta, h)$ is called the probability of fusion and $V = \{a, b, \dots, \}$. We assume the condition $P_a(\Delta, h) = P_b(\Delta, h) = \dots$ holds in a homogeneous logical network, and we can rewrite the probability that the team can do the fusion at hop h as,

$$P_V(\Delta, h) = 1 - (1 - P_a(\Delta, h))^{|V|} \tag{6}$$

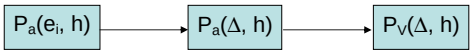


Figure 2: The relationships of three probabilities in the formal model

Figure 2 gives relationships among three probabilities in the formal model: $P_a(e_i, h)$, $P_a(\Delta, h)$, and $P_V(\Delta, h)$. From Equation 4 and Equation 6, we can find that the computation of the probability of fusion $P_V(\Delta, h)$ is trivial once we have the value of $P_a(e_i, h)$. Next we will describe how to compute the probability $P_a(e_i, h)$ for random walks.

Initially we have $P_a(e_i, 0) = 1/|V|$ for any sensor a. Its probability of knowing event e_i at hop 1, $P_a(e_i, 1)$, depends on the follows,

- Its probability of knowing event e_i at hop 0, $P_a(e_i, 0)$,
- The probability of being told e_i from its neighbors at hop $h = 1$.

The latter depends on two parts: (1) the probability of knowing e_i for one of its neighbors b at hop zero ($h = 0$), $P_b(e_i, 0)$; and (2) the probability of whether sensor a can get e_i from any neighbor b, denoted as $\delta(a, h)$.

The challenge is how to estimate $\delta(a, h)$ for random walks. Intuitively, random walks can be viewed as a random walker moving around in the network. If the walker's position is on node b, then it has $1/m$ probability to visit one of its neighbor. If not, sensor b cannot pass e_i to sensor a even when b knows e_i. Figure 3 shows an example of random walks for event e_i in the network, where the random walker visits node c, b, j, and g and its existing position is at g.

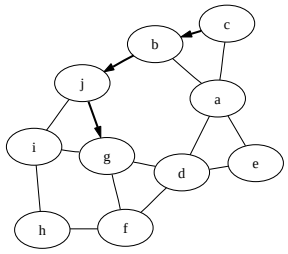


Figure 3: A example of random walks in the network, where the random walker visits nodes c, b, j, and g.

In this case, b cannot pass event e_i to a, although a is b's neighbor and b knows event e_i.

In other words, the event can be propagated from b to one of its neighbors a only and if only the walker's position is on b. For a random walk of e_i with $h - 1$ hops, b is one of the nodes that e_i visits, then we have $\delta(a, h) = 1/h$ for a $h - 1$ hops random walk.

Now let's estimate the probability being told e_i from its neighbors at hop h. Obviously, we have $\delta(a, 1) = 1$ when $h = 1$. Assume b has m neighbors, where m is the average degree of the network. The probability that b will forward e_i to a at hop one is $\delta(a, 1) * 1/m$. The probability of being told e_i by one of the neighbor at hop one is $P_b(e_i, 0) * \delta(a, 1) * 1/m$. If sensor node a has m neighbors, the probability a knows e_i from any of its neighbors is

$$P_b(e_i, 0) * \delta(a, 1) * 1/m * m = P_b(e_i, 0)$$

Similarly, the probability a knows e_i from its neighbors at hop h is $P_b(e_i, h - 1) * \delta(a, h)$, where $\delta(a, h) = 1/h$ refers to the probability of the random walker on node b.

The probability a knows e_i at h is

$$P_a(e_i, h) = P_a(e_i, h - 1) + \frac{(1 - P_a(e_i, h - 1))P_b(e_i, h - 1)}{h} \tag{7}$$

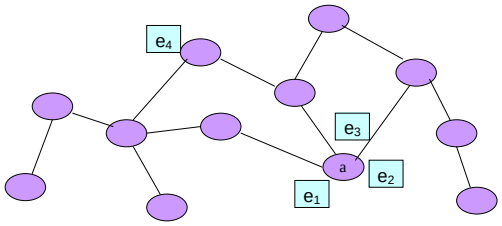


Figure 4: A subset of relevant events in the network

Up to now we consider to fuse all of the relevant events in the network. Fusing all relevant events is infeasible, since, sometimes, we may only the range of F for a given target and we do not know the exact number of F. It is also unnecessary when the confidence of fused data is already high. Therefore, we may only need to fuse a subset of the events if the number of relevant events F is big,, e.g., f out of F events and $f \leq F$. We will try to stop the propagation of the rest of relevant events in the network. As illustrated in Figure 4, sensor a will stop propagating event e_4 if it already fused other three relevant events e_1, e_2, and e_3.

For the rest of this paper, we simply choose $f = 3$ and $P_a(e,h)$ as the probability of fusing three events at hops h. If $F = f = 3$, we need to fuse all three of them and it can be described as $f/F = 3/3$. If $F > 3$ and $f = 3$, we may only fuse any three of them together.

Now we try to derive the relationship between $P_a(e,h)$ and $P_a(\Delta,h)$. The probability of fusion for 3/4 case is

$$P_a(e,h)+(1-P_a(e,h))P_a(e,h)+\ldots+(1-P_a(e,h))^3P_a(e,h) \quad (8)$$

When $F > f = 3$, the equation can be generalized as

$$P_a(e,h)(1+(1-P_a(e,h))+\ldots+(1-P_a(e,h))^{c_1-1}) \quad (9)$$

where $c_1 = C_F^f$. We can find that the value of $P_a(e,h)$ is still very small when we increase h to 10, e.g., $P_a(e,10) = 0.001$. Hence, we have

$$1+(1-P_a(e,h))+\ldots+(1-P_a(e,h))^{c_1-1} \approx C_F^f$$

The probability of fusing any three of them can be estimated as

$$P_a(\Delta',h) = c_1 P_a(e,h)$$

where $c_1 = C_F^f$.

The probability of fusion for $F > f = 3$ can be estimated as

$$P_V(\Delta',h) = 1-(1-P_a(\Delta',h))^{|V|}$$

Obviously, we have $P_V(\Delta',h) \geq P_V(\Delta,h)$ when we only need to a subset of relevant events.

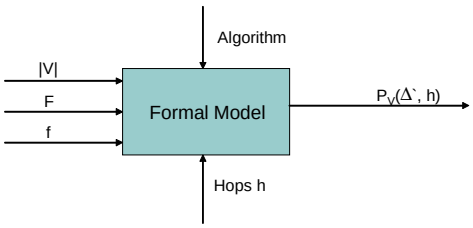


Figure 5: The inputs and outputs of the formal model

Figure 5 describes the inputs and outputs of our formal model. Given the size of networks $|V|$, two values F and f, we can easily compute the probability of fusion for random walks at hop h.

Figure 6 visualizes the outputs of the formal model — the probability of fusion $P_V(\Delta',h)$ — for a network with 100 nodes, where $f = 3$. While we may not use random walks in practice, the results here provide a baseline to test our proposed algorithms and estimate the low bound of TTL for a given network and the number of relevant events F.

6. EXPERIMENTAL RESULTS

In this section, we empirically study the data delivery algorithms. We use a random network of 100 nodes and each of them has, on average, four neighbors. Nodes are randomly chosen as the source of relevant events and they propagate the events according to different algorithms.

The experiments here serve as two purposes in this paper. First, we would like to validate our formal model for random walks using simulation results. For example, in the formal model, we assume each node has the same probability of knowing event e_i at hop h. This is not always true in practice. Second, we would like to study the robustness and efficiency of our algorithms.

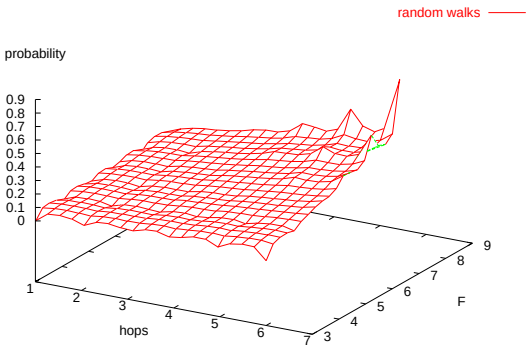


Figure 6: Probability of fusion for random walks for a sensor network with 100 nodes (f = 3).

6.1 Probability of Fusion

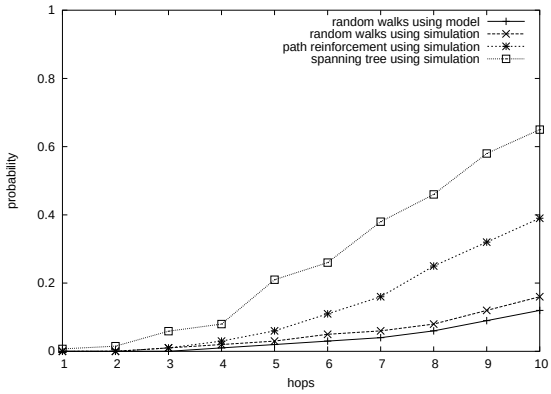


Figure 7: Probability of fusion for random walks, path reinforcement algorithm, and spanning tree (3/3 case).

We first study the probability of fusion for random walks, path reinforcement algorithm (without retransmission) and spanning tree when there is no failure. The probability of fusion from simulations is computed as $\frac{\sum s_i}{Num}$, where $s_i = 1$ if f events are successfully fused at iteration i. Otherwise, $s_i = 0$. The probability is averaged over 1000 iterations, e.g., $Num = 1000$.

Figure 7 and Figure 8 compare the probabilities of fusion for 3/3 and 3/5 cases using our formal model (only for random walks) and simulations. We can find that the results for random walks are consistent with those from the formal model. Also, random walks are quite easy to beat since they do not consider the relevance between events. For both 3/3 and 3/5 cases, path reinforcement algorithm performs better than random walks.

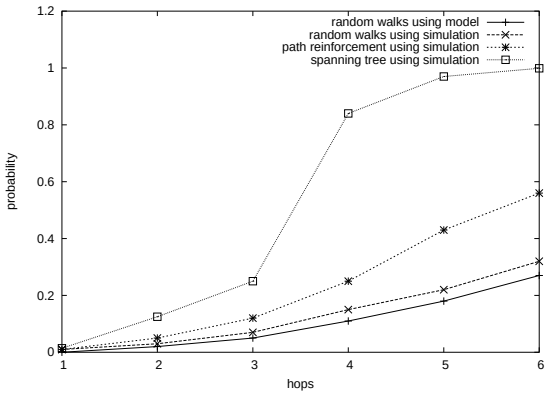


Figure 8: Probability of fusion for random walks, path reinforcement algorithm and spanning tree (3/5 case).

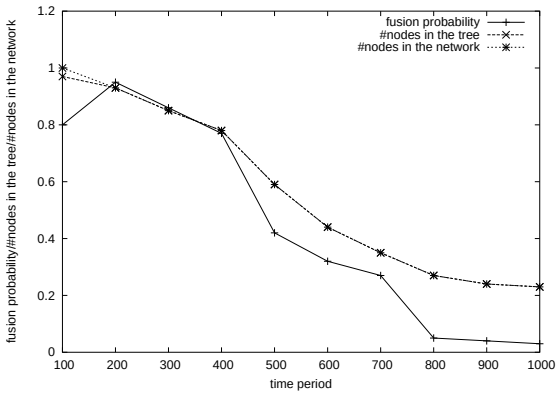


Figure 9: The fusion probability, percentages of active nodes in the tree and in the network (the line for #nodes in the tree is overlaid by the line for #nodes in the network).

We also study the effects of spanning tree for data delivery and fusion in the network. We use the first eight sets of relevant events to train the spanning tree and then we test the probability of fusion for the spanning tree. The results indicate that learning is quite effective for sensor data delivery. For the 3/5 case, the local learning scheme at least doubles the probability of fusion for path reinforcement algorithm when the hop of events is greater than three. The probability of fusion for 3/5 is guaranteed to be high if we allow the set of events to be propagated 6 hops in the network with 100 nodes.

6.2 Robustness

In this section and next section, we study if our localized algorithm is robust to failures in the network. We choose a random number between 0 and 0.01 as failure rate λ_a for any sensor a. We introduce a random number between 0 and 0.01 as the link failure rate between a node and any of its neighbors.

Figure 9 describes the fusion probability and percentages of active nodes in the tree and the network when $F = 5$. The dead nodes are detected after every 100 time periods. The figure tells us that trees evolve and merge and quickly one spanning tree covers most of the sensor nodes. Moreover, the tree always connects the nodes in the remained network. Another result is the probability of fusion decreases when the size of the network decreases. The reason is that the number of relevant events for fusion F becomes small, e.g., $F < 5$.

We consider the worst case of a mobile sensor network, where more than 50% nodes are destroyed after 500 time periods. In the hostile environments like this, e.g., a battlefield, we probably need to reduce the necessary number of events for fusion f . For example, if we reduce f from 3 to 2 ($F = 5$ on the network of 100 nodes), we can still reach the fusion probability of 70% after 500 time periods (Figure 10). Another solution is to incrementally deploy more sensor nodes to the sensor networks, as suggested by [6]. Figure 10 shows that, if we add 5 nodes to the network after every 100 time periods, we can still keep pretty high probability of fusion.

One concern of node failures is whether fused events can

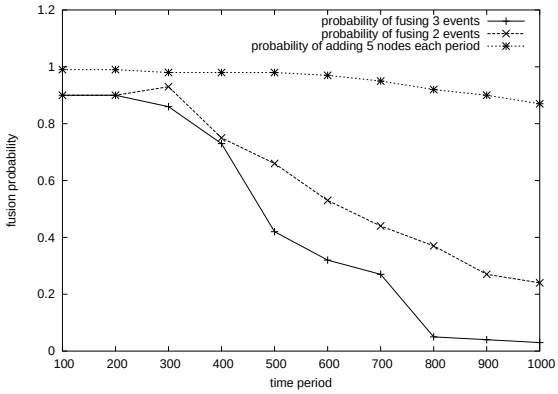


Figure 10: The fusion probability in the network when adding more nodes.

be lost when the fusion node happens to be destroyed. We study the fusion process and find this is not a problem when F is relatively big. For example, there are multiple copies of fused events at different nodes for 3/5 case when each event is propagated to 6 hops. The redundancy of fused events increases the survivability of the information and success rates of coordinating plans and actions for the network.

6.3 Efficiency

One question we did not answer in the formal model is if path learning algorithm is more efficient than path reinforcement algorithm. Since a node using path learning algorithm needs the feedback from its neighbors to learn their reliabilities. This will at least double the traffic in the network.

From Figure 11 we can find that if the events are allowed to propagate longer in the path reinforcement algorithm, both algorithms have very high probability of fusion. However, path reinforcement algorithm is vulnerable to node and link failures. Its probability of fusion becomes low when node failures happen. One reason is that the events are lost during information delivery in path reinforcement algorithm. In path learning algorithm we maintain the reliable

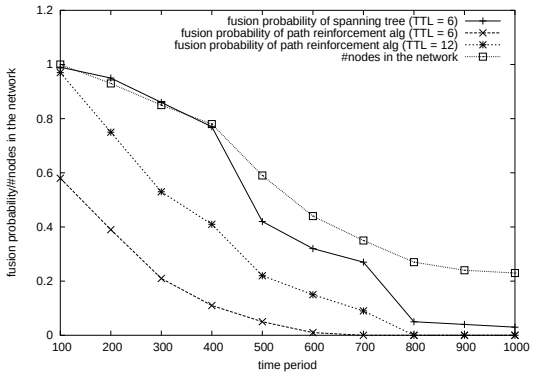


Figure 11: Probability of fusion and efficiency for path reinforcement algorithm and spanning tree.

data delivery of events through a retransmission mechanism, where an event is delivered to another neighbor if a node or link failure is found.

7. CONCLUSION

This paper studies localized data delivery algorithms for mobile ad hoc sensor networks with failures. The empirical results show that our path learning algorithm significantly increases the probability of fusion in a sensor network with node and link failures.

In future work we will develop methods for tuning system parameters, such as learning rate α in path learning algorithm, for different numbers of relevant events F, and node and link failure rates. Moreover, we have previously shown that small-world networks are more efficient than random networks for passing tokens in large teams [20, 18]. We plan to study the effects of different network topologies on probabilities of fusion, especially when both node and link failures happen in the network.

Acknowledgements

This research has been sponsored in part by AFRL/MNK Grant F08630-03-1-0005 and AFOSR Grant F49620-01-1-0542. We are indebted to the anonymous reviewers for their helpful comments.

8. REFERENCES

[1] Justin A. Boyan and Michael L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems*, pages 671–678, 1994.

[2] Chee-Yee Chong and Srikanta P. Kumar. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, 2003.

[3] David L. Hall, editor. *Mathematical Techniques in Multisensor Data Fusion*. Artech House Inc., 1992.

[4] David L. Hall and James Llinas. An introduction to multisensor data fusion. In *Proceedings of the IEEE*, pages 6–23, 1997.

[5] G. Hoblos, M. Staroswiecki, and A. Aitouche. Optimal design of fault tolerant sensor networks. In *Proceedings of the IEEE International Conference on Control Applications*, pages 467–472, 2000.

[6] Andrew Howard, Maja J. Mataric, and Gaurav S. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots Special Issue on Intelligent Embedded Systems*, 13(2):113–126, 2002.

[7] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MobiCom*, pages 56–67, 2000.

[8] Gal A. Kaminka and Milind Tambe. What is wrong with us? improving robustness through social diagnosis. In *AAAI*, 1998.

[9] Victor Lesser, Charles Ortiz, and Miland Tambe, editors. *Distributed Sensor Networks: A Multiagent Perspective*. Kluwer Academic Publishers, 2003.

[10] Ning Li, Jennifer C. Hou, and Lui Sha. Design and analysis of an mst-based topology control algorithm. In *InfoCom*, pages 1702–1712, 2003.

[11] Alexei Makarenko and Hugh Durrant-Whyte. Decentralized data fusion and control in active sensor networks. In *Proceedings of the Seventh International Conference on Information Fusion*, 2004.

[12] James Manyika and Hugh Durrant-Whyte. *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*. Ellis Horwood, 1994.

[13] Robin R. Murphy. National science foundation summer field institute for rescue robots for research and response. *AI Magazine*, 25(2):133–136, 2004.

[14] Ruixin Niu, Pramod K. Varshney, M. Moore, and Dale Klamer. Decision fusion in a wireless sensor network with a large number of sensors. In *Fusion*, 2004.

[15] H. Van Dyke Parunak, Sven A. Brueckner, and James J. Odell. Swarming coordination of multiple UAV's for collaborative sensing. In *Proceedings of the Second AIAA Unmanned Unlimited Systems Technologies and Operations Aerospace Land and Sea Conference and Workshop*, 2003.

[16] Matt Rosencrantz, Geoffrey Gordon, and Sebastian Thrun. Decentralized sensor fusion with distributed particle filters. In *UAI*, 2003.

[17] Allison Ryan, Marco Zennaro, Adam Howell, Raja Sengupta, and J. Karl Hedrick. An overview of emerging results in cooperative UAV control. In *Proceedings of 43rd IEEE Conference on Decision and Control*, 2004.

[18] Paul Scerri, Yang Xu, Elizabeth Liao, Justin Lai, and Katia Sycara. Scaling teamwork to very large teams. In *AAMAS*, pages 888–895, 2004.

[19] Devika Subramanian, Peter Druschel, and Johnny Chen. Ants and reinforcement learning: A case study in routing in dynamic networks. In *IJCAI*, 1997.

[20] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, June 1998.

[21] Feng Zhao and Leonidas Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann Publishers, 2004.

Efficient Agent-Based Cluster Ensembles

Adrian Agogino
UCSC, NASA Ames Research Center
Mailstop 269-3
Moffett Field, CA 94035, USA
adrian@email.arc.nasa.gov

Kagan Tumer
NASA Ames Research Center
Mailstop 269-4
Moffett Field, CA 94035, USA
ktumer@mail.arc.nasa.gov

ABSTRACT

Numerous domains ranging from distributed data acquisition to knowledge reuse need to solve the cluster ensemble problem of combining multiple clusterings into a single unified clustering. Unfortunately current non-agent-based cluster combining methods do not work in a distributed environment, are not robust to corrupted clusterings and require centralized access to all original clusterings. Overcoming these issues will allow cluster ensembles to be used in fundamentally distributed and failure-prone domains such as data acquisition from satellite constellations, in addition to domains demanding confidentiality such as combining clusterings of user profiles. This paper proposes an efficient, distributed, agent-based clustering ensemble method that addresses these issues. In this approach each agent is assigned a small subset of the data and votes on which final cluster its data points should belong to. The final clustering is then evaluated by a global utility, computed in a distributed way. This clustering is also evaluated using an agent-specific utility that is shown to be easier for the agents to maximize. Results show that agents using the agent-specific utility can achieve better performance than traditional non-agent based methods and are effective even when up to 50% of the agents fail.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Performance

Keywords

Clustering, Multiagent Systems, Reinforcement Learning

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

Clustering is a difficult problem in numerous fields including machine learning, multi-agent systems, data management and bioinformatics [13, 19, 10]. The central concept of clustering is that data points should be partitioned into separate clusters so that data points within a cluster are "close" to each other, while data points from different clusters are "far" from each other. A set of clusters with these properties is a clustering. The goal of the clustering problem is to find good clusterings. While the clustering problem is typically an NP-complete optimization problem, there are numerous heuristic clustering algorithms including k-means, expectation-maximization and Metis [12, 9, 2, 7]. Note that even though clustering is often used within multi-agent systems (e.g. to group similar agents), agents are not typically used to solve the clustering problem itself. This is as much due to the traditional view of a cluster as a passive label, rather than an active agent, as to the difficulty in decoupling the interactions among the agents representing the labels.

This paper focuses on using agents for an important subset of the clustering problem known as the *cluster ensemble* problem (Figure 1) [15]. In the cluster ensemble problem one needs to combine multiple clusterings, formed from different aspects of the same data set, into a single unified clustering. Cluster ensembles are particularly useful when all of the original data points are not available to create a clustering, but separate clusterings of the data still exist. This situation occurs:

1. When some of the data points come from proprietary sources, where the data owners are willing to reveal their clustering of the data, but not the data itself.
2. When the original data points are simply lost, or have been thrown away, but the much smaller summery cluster data is saved.
3. When all of the original data is available, but it may be too big to store in one site of computation. In this case, it may be desirable to make separate clusterings of different parts of the data and combine them later.
4. When we want to inject prior knowledge into learning systems or re-use information since new clusterings can be combined with older clusterings without needed to know all the information used in the previous clusterings or how they were derived.

Currently the best algorithms for cluster ensembles are graph-theoretic methods [14]. These methods treat the clus-

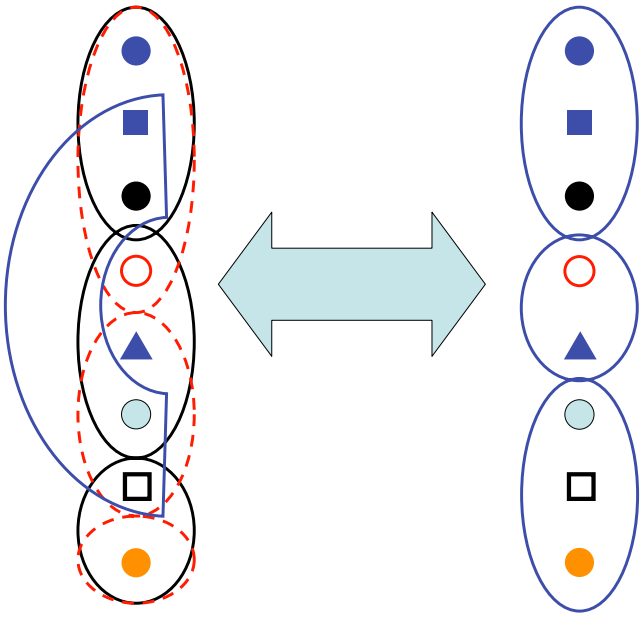


Figure 1: Cluster Ensembles. Shown on left are three different clusterings for eight data points. The goal of the cluster ensemble is to create a unified clustering (shown on right) that preserves the most information from the original clusterings.

ter ensemble problem as a global static problem, examining the entire data set and producing a single result. While these methods have shown to have good results on certain information theoretic measures, they tend to be inflexible [15]. The graph-based methods require the cluster combining to happen on a single node of computation. This requires all the data to be centrally available, creating a single point of failure. This centralization makes them inappropriate for domains that are inherently distributed, and have a potential for high component failure such as in space-based operations.

Instead of graph-based cluster ensemble methods, this paper proposes using learning agents to solve the cluster ensemble problem. In this approach each agent contributes to the final clustering and uses reinforcement learning to learn to maximize a utility function based on the clustering. Unlike graph-based methods, the multi-agent approach treats the cluster-ensembe problem as a dynamic system, where agents continually attempt to create a unified clustering that maximizes a utility. This dynamic agent-based approach is more flexible and robust in the following ways:

- Agents are robust against noisy data sources.
- Agents naturally adapt to loss of data sources.
- Computation is distributed.
- There is no single point of failure, so performance gracefully degrades with the number of failures.
- Computation can be stopped at any time when performance is adequate.
- Data sources do not have to be revealed.

These advantages allow multi-agent cluster ensembles to be used more effectively in more domains than their non-agent-based counterparts.

Despite these advantages, multi-agent systems have not been used in the cluster ensemble problem due to coupling between the cluster assignments. The utility the agents attempt to maximize is highly non-linear, limiting the independence assumptions that can be made. This non-linearity prevents many multi-agent techniques from being applicable. In addition if an agent attempts to maximize the utility directly, it faces a difficult signal-to-noise problem in that it does not know the effect of its actions on the utility, among the effects of the actions of all the other agents. To address this issue, this paper shows how agents can instead use agent-specific utilities that are much easier to maximize, yet are still aligned with the original utility.

In Section 2, this paper gives a formal description of the cluster ensemble problem as well as a brief overview of a graph-based and simple greedy solutions to the cluster ensemble problem. In Section 3 this paper presents an agent-based approach. Finally in Section 4 the paper shows results for the performance of agent-based cluster ensemble methods, and how they are robust against agent failure.

2. CLUSTER ENSEMBLE PROBLEM

In clustering problems a set of data points are grouped into clusters, so that data points in the same cluster are "closer" to each other than data points from different clusters. The closeness between data points is typically measured with a domain dependent distance metric. This metric is influenced by the type of data, the goal of the clustering and even the "scale" of the clustering depending on the amount of clusters desired [16, 15, 4]. Given a distance metric, a "soft-clustering" can be performed where data points belong to different clusters with varying probability, or hard clustering where each data point is part of only one cluster [6]. This paper will focus only on hard-clustering. This type of clustering problem is equivalent to creating a non-overlapping partition of the data. In this paper we refer to a particular partition as a *clustering*. In general there can be many different clustering for the same data, depending on the clustering algorithm used [11, 5, 7]. In addition there can be sets of related clusterings that are formed with overlapping subsets of the data [15]. The goal of the cluster ensemble problem is to create a single clustering that best characterizes a set of clustering, without using the original data points used to generate the clusterings [1].

To formalize the clustering ensemble problem, we need a way to compare two clusterings. In this paper, this is done through an information theoretic measure previously used in the clustering community called *normalized mutual information* (NMI) [15]. This measure is based on the sizes of the clusters within the clusterings (a formal justification for this is presented in [14]). To compute this measure we

[1]Note that sometimes the clustering problem includes computing an "optimal" number of clusters in addition to assigning data points to these clusters. However, finding the correct number of clusters is ill-defined, and changes depending on the context. All the algorithms in this paper assume that the final number of clusters is known ahead of time. If the number of clusters is not specified ahead of time, meta-algorithms can be used, where the number of clusters is varied until a good number is found.

first define a clustering X, where each cluster $x_i \in X$ is a set of data points. We now define the mutual information between clusterings X and Y as:

$$I(X,Y) = \sum_{x \in X} \sum_{y \in Y} \frac{|x \cap y|}{n} log_2 \left(\frac{n|x \cap y|}{|x||y|} \right) \quad (1)$$

where there are n data points and $|\cdot|$ and $\cap$ are the set cardinality and intersection operators respectively. The intersection operation $x \cap y$ returns the data points that are common to both clusters x and y. Next we define the entropy of a clustering, X, as:

$$H(X) = -\sum_{x \in X} \frac{|x|}{n} log_2 \left(\frac{|x|}{n} \right) \quad (2)$$

Now we can define the normalized mutual information (NMI) between two clusters, X and Y as:

$$\text{NMI}(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}} \quad (3)$$

which has the desirable property of being bounded by [0, 1] and having $\text{NMI}(X,X) = 1$.

The normalized mutual information between two clusters measures how much information they have in common. If two clusterings are similar they will have an NMI close to one. If two clusterings are completely different then they have an NMI of zero. In this case knowing one clustering would not help one predict the clusters within the second clustering. If we had a "true" clustering of the data we would want our ensemble clustering to have as high as possible NMI with respect to this true clustering. However, typically all we have to compare our ensemble clustering with is the original clusterings used to create the ensemble. In this case our goal is to create an ensemble clustering that shares as much information as possible with the clusterings it was created from. This can be measured by averaging the normalized NMI between the original clusterings and the ensemble. Formally we define the ANMI between a clustering X and a set of clusterings $\overline{Y}$ as:

$$\text{ANMI}(\overline{Y}, X) = \frac{1}{|\overline{Y}|} \sum_{Y \in \overline{Y}} \text{NMI}(X,Y) \ . \quad (4)$$

When an ensemble clustering X has more information in common with the original clusterings $\overline{Y}$, $\text{ANMI}(\overline{Y}, X)$ will have a higher value. Therefore the goal of the agents is to find the clustering, X^*, that maximizes the ANMI between X^* and the set of available clusterings $\overline{Y}$.

2.1 Graph-based Approaches

The graph-based approaches to the cluster ensemble problem involve representing the data points as nodes in a hypergraph, and the clusters as undirected hyperedges of the hypergraph. This paper will compare cluster ensemble performance with three of these algorithms, which are described in detail in [15]. Note that these methods are only heuristics and are not optimal, since similar to clustering methods, the graph methods tend to be NP-complete. However the methods described here have been shown to have high performance in practice. The first algorithm called CSPA (Cluster-based Similarity Partitioning Algorithm) provides moderate performance, but requires significant computation. CSPA works by creating a similarity measure between data points. Within a clustering, all data points in the same cluster have a similarity of 1, and data points from different clusters have a similarity of 0. If the similarities between data points are averaged over different clusterings, a new single clustering can be made with a similarity based clustering algorithm, such as METIS [7].

The second clustering algorithm is called HGPA (HyperGraph Partitioning Algorithm) and is simpler to compute than CSPA. This method creates the final combined clustering for the ensemble by using HMETIS [8] to perform hypergraph partitioning on the hypergraph that represents the clusterings. The objective of this method is to create clusters that break the least number of hyperedges. The final method, called MCLA (Meta-CLustering Algorithm), provides better performance than HPGA and retains its low computational complexity. This algorithm works by collapsing a group of related hyperedges into a single hyperedge. This can be seen as clustering clusters.

2.2 Simple Greedy Optimization

Simple greedy optimization approaches have been applied to the cluster ensemble problem in [15]. In this method, one starts with a single representative clustering, X, which is usually the clustering that has the highest ANMI with respect to all the other clusterings:

$$X = argmax_{X' \in \overline{Y}} \text{ANMI}(\overline{Y} \backslash X', X') \quad (5)$$

where $\overline{Y}$ is the set of clusterings, and $\backslash$ is the set difference operator. For each data point, a new clustering is created from the previous clustering by moving the data point to a new cluster at random. If this new clustering has a higher ANMI than the previous one, then it is preserved, otherwise it is thrown away. The algorithm is repeated for each data point. When it has gone through all the data points, the algorithm stops if all of the new clusterings where thrown away. Otherwise it repeats through all the data points. This algorithm can be seen as a form of serial simulated annealing with zero exploration or as greedy hill-climbing. Unlike the graph-based methods, this algorithm is dynamic as it continually tries to maximize utility. However, this algorithm has a number of difficulties. Since the exploration is zero, the system will only reach a local maximum. Also since for each loop, the ANMI has to be computed for every data point, the computationally burdensome when there are many data points.

3. AGENT-BASED APPROACH

Similar to the greedy optimization approach, the agent-based approach treats the cluster ensemble problem as a dynamic optimization problem, with the agents striving to create a final combined clustering with the highest utility. However, the agent-based approach requires significantly less computation and is naturally distributed. In this approach agents are assigned a set of data points and vote on which cluster in the final clustering the data points should belong to. Through reinforcement learning, the agents learn how to vote in a way that leads to the best final clustering.

3.1 Assigning Agents and Voting

In this paper, the cluster ensemble problem is approached by assigning one or more agents to each cluster in the original set of clusterings. If m agents are assigned to each

cluster, then the multi-agent sytem for r clusterings with k clusters each would have mrk agents. The action of an agent is to vote on the cluster in the combined clustering, to which the data points it is responsible for should belong. A data point will then belong to a final cluster that received the most votes from agents that were responsible for the cluster containing the data point. This process is shown in Figure 2. The global utility of the system is the ANMI of the combined clustering with respect to the original clusterings. Agents can then try to maximize the ANMI, by using their private utilities to help them choose the best actions. In this example, the top two points are in the same clustering in two of the three clusterings (agents 1 and 4). Similarly, the second and third data points are in the same clustering in two of three clusterings (agents 1 and 8). In the final clustering, all three data points belong to the same cluster, as this clustering minimizes the mutual information between the original clusterings.

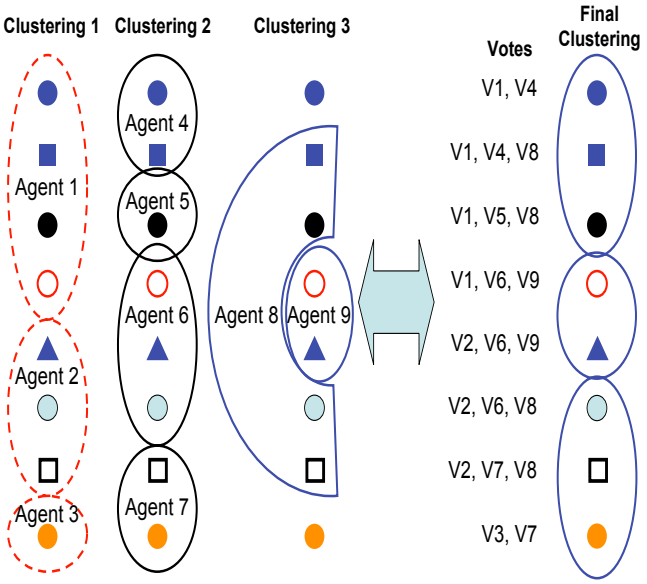


Figure 2: Agent-Based Cluster Ensembles. An agent is assigned to a cluster in each clustering. Agents then vote on the final cluster to which the data points within their current cluster belong.

Formally the votes by the agents for a combined clustering Z can be represented by a two dimensional array of sets indexed by agents and clusters. If agent i chooses cluster $z \in Z$, then $V_{i,z}$ equals the set of data points in agent i's assigned cluster. If agent i did not choose z then $V_{i,z}$ equals the empty set. The number of votes for data point p to be put in cluster z can be formulated as follows:

$$N_{p,z} = \sum_i I_{p \in V_{i,z}} \tag{6}$$

where I is an indicator function. The assignments of data points to a cluster in the combined clustering can now be expressed as:

$$z = \{p | N_{p,z} = max_{z'} N_{p,z'}\} \tag{7}$$

Tie votes are broken randomly.

3.2 Global Utility

The global utility that the agents ultimately try to maximize is the ANMI between Z and the initial set of clusterings $\overline{Y}$:

$$G(\overline{Y}, V) = \text{ANMI}(\overline{Y}, Z(V)) \; . \tag{8}$$

Note that the global utility can be computed in a distributed way when the votes V are broadcasts to all agents. Each agent can compute the final clustering $Z(V)$ and compute its own normalized mutual information NMI(Y, Z(V)). The NMIs can then be broadcasts and all the agents can compute ANMI and therefore G on their own. While this process requires a fair amount of communication, it is usually considerably less than broadcasting the entire data set, depending on the size of the clusters. While an agent can compute the global utility, it is difficult for an agent to maximize the global utility directly since this utility is a function of the actions of all of the other agents. If an agent takes an action and the global utility increases, the agent does not know if the increase is due to its action or the actions of other agents. As a consequence, when there are many agents, it will take a large number of learning steps for an agent to discern the effects of its actions from the actions of all the other agents.

Note that while the computation of G ultimately uses information derived from all of the data points, these data points are never assembled into a central data set. Instead the computation of G involves the agents broadcasting what they believe the final clustering should be, not what their current clusterings are. In this distributed framework the original clusterings do not have to be revealed. This decentralized computation allows the agent-based approach to be used in domains where the owners of individual data sources do not want their data set to be made fully available to the other data owners.

3.3 Agent-Specific Utility

Instead of maximizing the global utility directly, an agent can learn more quickly if it maximizes an agent-specific "private" utility that is more influenced by its own actions than the actions of all the other agents. We call this relative influence an agent has on its utility, the "learnability" of the utility [17]. Learnability for a private utility g_i for agent i can be informally defined as the ratio of agent i's influence on g_i to all other agents' influence on g_i. Generally if a utility is more learnable for an agent, the faster an agent will be able to maximize it. However, having all the agents being able to maximize their private utilities is not useful if this does lead to the maximization of the global utility. For the maximization of private utilities to lead to the maximization of the global utilities, each private utility should be "factored." A private utility is factored when any action an agent takes to increase its private utility also increases the global utility [17, 20].

A utility that has been shown to have high learnability while being factored is the difference utility, defined as:

$$D_i(z) = G(z) - G(z_{-i}); \tag{9}$$

where z are the actions of all the agents and z_{-i} are the actions of all the agents other than agent i. The second term of the difference utility is a counterfactual, computing the value of the system without agent i. Subtracting this counterfactual from the global utility therefore computes an agent's

contribution to the global utility. This utility is factored since the second term is not a function of the agent's actions, therefore it can only influence the utility by changing the value of the first term, the global utility. Furthermore, this utility usually has far better learnability than does $G(z)$ because the second term of D_i removes a lot of the effect of other agents (i.e., noise) from agent i's utility [17]. This utility has proven effective in many multi-agent system domains including network routing, rover control, job scheduling and congestion games [1, 18, 17].

Using the global utility defined in 8 we can define the difference utility for the cluster ensemble problem as:

$$\mathrm{D_i}(\overline{Y}, V) = \mathrm{ANMI}(\overline{Y}, Z(V)) - \mathrm{ANMI}(\overline{Y}, Z(V')) \quad (10)$$

where V' is the same as V, except that all of the elements of V' that are indexed by i are equal to the empty set. Since this utility is factored, agents maximizing $D_i(\overline{Y}, V)$ will tend to maximize $G(\overline{Y}, V)$, though they can learn more quickly since each agent has more influence over its own difference utility than over the global utility. Note that when an agent's vote does not influence the final clustering Z(V') the difference utility does not have to be computed since its value is zero. In large systems, this case will occur often. If an agent's vote is a deciding vote, then all the agents will have to compute and broadcasts their NMIs for the counterfactual $Z(V')$. In the worst case when all the votes are deciding votes, the amount of computation grows linearly with the number of agents. However, this system is still distributed and an agent can simply not average in NMIs from other agents that fail to respond. This distribution is critical to robustness as agents can still try to maximize their utilities even when some of the agents are not working properly.

4. RESULTS

The performance of the multi-agent approach to cluster ensembles relative to other methods was tested with three data sets. The first data set was artificial, while the other two came from real-world problems. These data sets showed different aspects of the cluster ensemble problem. In the first data set, the "true clustering" was known since it was artificially generated. The second data set involved clustering images of hand-written images, where the true number of clusters was known, but the correct clustering was subject to interpretation. The final data set involved groupings in the Yahoo! web site, where both the correct number of clusters and the correct clustering was subject to interpretation. The experiments showed that the multi-agent approach often achieved superior performance to the best existing centralized cluster ensemble algorithms, while being able to recover from a number of types of agent failures.

All agents learned with a simple single-time-step reinforcement learner. This learner was equivalent to an Q-learner with infinite reward discounting ($\gamma = 0$) so that only the immediate reward was processed. At every time step, each agent would choose one of k clusters based on its estimates of the utility for that choice. These estimates were stored in a utility table of size k. This process was done in an ϵ-greedy fashion (with $\epsilon = 0.005$) where the highest valued cluster would be chosen with probability $1 - \epsilon$ and a random cluster would be chosen with probability ϵ. After all the agents chose their cluster, each agent would compute their private utility and use it as the reward to update their reinforcement learner.

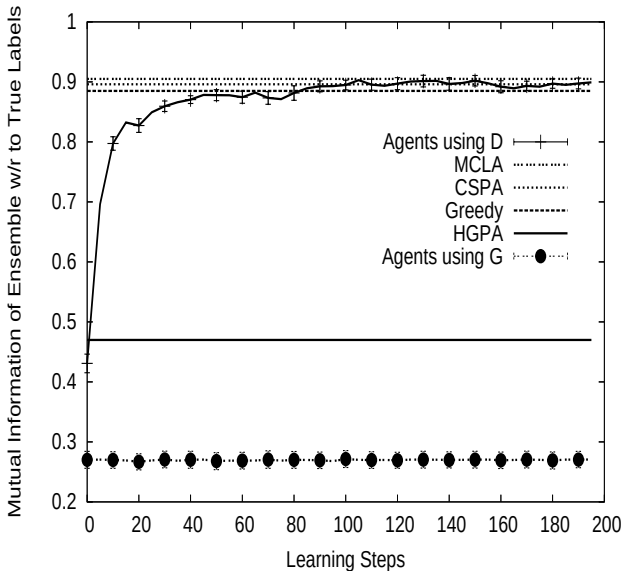


Figure 3: Results of Ensemble Methods with Artificial Data. Eight clusterings are created by adding 40% noise to the original cluster labels. The ensemble tries to recover the original labels from the eight clusterings. The decentralized algorithm using the difference utility performs as well as the best graph-based algorithm, which requires centralization.

4.1 Artificial Data

In the first experiment, four hundred data points where randomly placed into ten clusters to form an initial clustering. From this initial clustering, eight variants were created. Each variant was made by first duplicating the original clustering and then adding random noise by moving a random subset of the data points to new clusters (for each data point in the subset, the cluster it was moved to was chosen independently over a uniform distribution over all the clusters). The amount of random noise was specified by B, the fraction of all the data points contained in the subset. These noisy variant clusterings were used as the ensemble clusterings and experiments were performed with B ranging from 0.0 to 1.0. The performance of the algorithms were determined by the NMI between the clustering produced by the algorithm and the initial clustering. Since this is an artificial data set we have "ground truth" about what the true clusterings should be. A better cluster ensemble algorithm should produce a clustering as close to possible to the initial clustering in which the eight ensemble clusterings were derived. Figure 3 shows the results with $B = 0.4$ for agents using D_i or G as their utility along with results obtained from the graph-based methods and the simple greedy method.

The results show that the agent-based method using the difference utility performed as well as the best graph-based method, MCLA and considerable better than the lowest performing graph-based method, HGPA. In noisy domains HGPA had particular poor performance as compared to MCLA because MCLA could reduce noise by collapsing similar edges. In addition to performing as well as the best

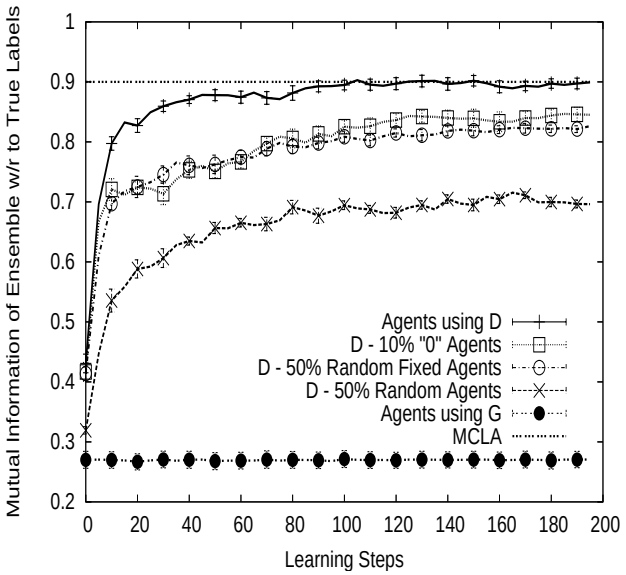


Figure 4: Results of Ensemble Methods with Artificial Data with Faulty Agents. Systems with agents using D generally perform well when some of the agents fail. Top curve shows performance when all agents are working properly. Second curve shows performance when 10% of the agents always choose cluster 0 as their action. Third curve shows performance when 50% of the agents always choose the same random cluster as their action. Fourth curve shows performance when 50% of the agents always choose a different random action at every time step.

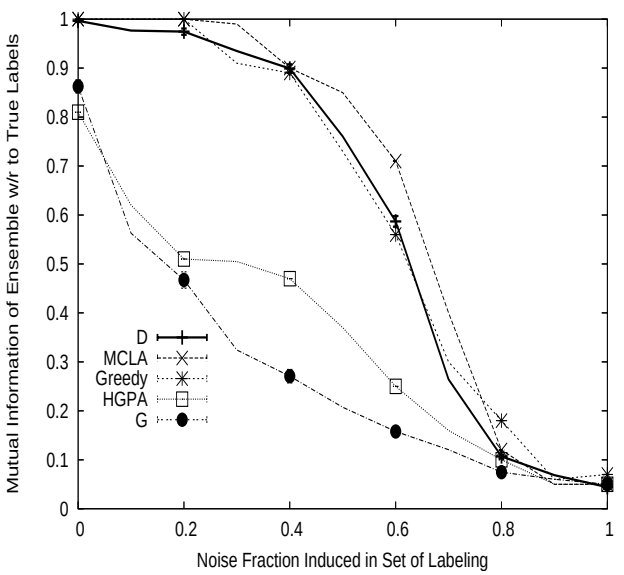


Figure 5: Results of Ensemble Methods on Artificial Data. Eight clusterings are created by adding variable amounts of noise to the original cluster labels. The ensemble tries to recover the original labels from the eight clusterings. Relative performance of methods is consistent across noise levels (CSPA performance is almost identical to greedy method, but not shown to reduce clutter).

graph-based methods, the agent-based method using the difference utility performed about the same as the simple greedy method. However this performance was achieved with much less computation. The greedy method was reported to take an hour on a 1Ghz PC [15]. The agent-based method took only 45 seconds on a similar computer. In contrast to agents using the difference utility, agents using the global utility performed very poorly. This poor performance can be attribute to the large number of agents in this problem (eighty). When an agent choose a cluster and the value of the global utility changed, the agent did not know whether the change was caused by its action or the action of one of the seventy nine other agents. This very low performance of the global utility is likely the reason why agent-based cluster ensembles have not been pursued until now.

In addition to testing the agents under normal conditions, we tested the case where some of the agents fail. In this test three agents were used for every cluster to provide redundancy. This redundancy was tested under three failure scenarios. In the first scenario 50% of the agents were faulty. These faulty agents (called "random agents" in the figures) chose a random cluster at every time step. In the second scenario 50% percent of the agents (called "fixed random agents") chose a random cluster at the beginning of a trial and kept making the same choice throughout the trail. In the final scenario 10% of the agents always chose the first cluster (these are called "0" agents). The results shown in Figure 4, reveal that agent-based cluster ensembles can perform well in a number of adverse conditions. Even when half of the agents were faulty, the system could still perform well. Note that the agent-based system in the first scenario performed worse than the one in the second scenario, even though the same number of agents were faulty. This can be explained by the adaptivity of the agents. In the second scenario the faulty agents always took the same wrong move, so the working agents could adapt to overcome the adversity. In the first scenario, the faulty agents simply added random noise to the system, which could not be easily adapted to. Despite the inherent failure tolerance of the multi-agent system, there was still a possibility of catastrophic failure if too many of the agents failed in the same way. While the system was able to recover when 10% of the agents choose the same wrong cluster, the performance drops to zero if 50% percent of the agents choose the same wrong cluster (not shown in the graphs). This failure is caused by the voting scheme, since when 50% percent of the agents vote for the same cluster, that cluster will always win. In this case, the final clustering will always have just one cluster.

The previous experiments were done with the noise parameter, B, used to generate the noisy clusterings from the initial clustering, set to 0.4. To show that the relative performance of all the methods was not dependent on this choice of noise parameter, we tested the performance of the methods on a wide range of values shown in Figure 5. The results show that for a wide range of noise parameters that the agent-based methods perform well, and the relative performances of the other methods is about the same. Averaged over all of the noise parameters, the NMI of MCLA, D, Greedy and HGPA are 6.4, 6.0, 6.1 and 3.5 respectively.

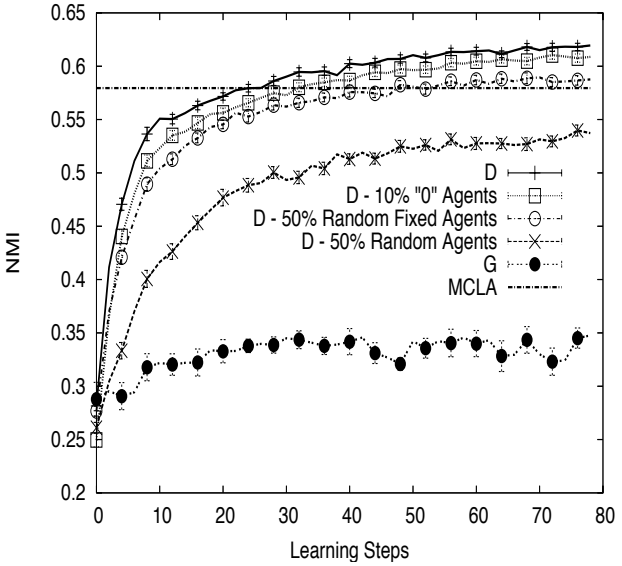


Figure 6: Results of Ensemble Methods for the Pendig Data Set. Agents using difference utility perform better than the best graph-based algorithm. Even when up to 50% of the agents fail, the system performs well.

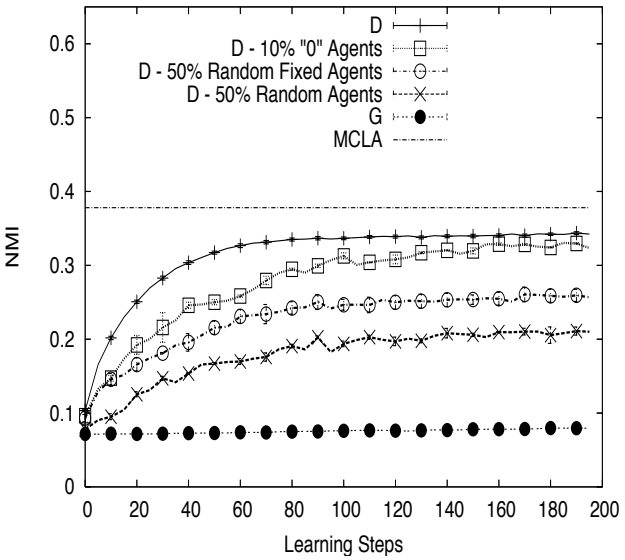


Figure 7: Results of Ensemble Methods for the Yahoo Data Set. Agents using difference utility slightly under-perform the best graph-based algorithm, but still perform adequately when up to 50% of the agents fail.

4.2 Pendig Data Set

The Pendig data set contained one thousand data points, with each data point having sixteen features. Each data point represented a processed handwritten digit. The data points were clustered into ten clusters, with each cluster representing a digit from '0' to '9.' While this data set is known to have precisely ten clusters, the true clustering is subjective since some of the images could be ambiguous. From the original Pendig dataset we produced ten different clusterings (by coincidence, each with ten clusters) by partitioning the data into ten data sets. Each point in the new data sets contained four features sampled from the original sixteen. Ten clusterings were then produced by clustering each of the new data sets with METIS. Using these clusterings for the ensemble, agents using the difference utility were able to perform better than the best centralized graph-based algorithm, MCLA (see Figure 6). Again the results show that agent-based cluster ensembles can perform well in a number of adverse conditions. Even when half of the agents were faulty, the system could still out-perform the graph-based methods. Only in the case when half the agents choose random actions at every time step, did the performance drop below that of the MCLA. Though even in this case the agents using the difference utility with 50% failure were able to perform better than agents using the global utility when there are no failures.

4.3 Yahoo Data Set

In this experiment a processed version of the Yahoo! data set was used. This data set has also been used in [16] and [3]. The data set contained 2340 data points, with each data point having 2903 features. Each data point represented a document and the features were a pruned set of word frequencies contained in the document. The data set was clustered into twenty clusters, based on which Yahoo! news category they were originally placed in. Note that both the correct number clusters and the true clusterings were subjective for this data sets. More or less news categories could have been used, and many documents could have been placed in different categories.

In the experiment we created twenty clusterings (coincidentally, each clustering having twenty clusters) by splitting the full data set into twenty separate data sets. Each new data set contained all of the 2340 data points, but each data point only had a 128 element subset of the original features. Each new data set was then clustered using METIS, forming twenty clusterings. These new clusterings were then used for the ensemble in the experiment. The output of the ensemble algorithm was then compared to the original Yahoo! clustering using NMI. Figure 7 shows the results between an agent-based method using D_i as its utility, an agent-based method using G as its utility and the best performing graph-based method, MCLA.

The results show that agents using the global utility were unable to produce an adequate clustering. This is not surprising since there were 400 agents in the system. In this case, the learnabity of the global utility was very poor since an agent cannot easily see the contribution of its action on the global utility, amongst the 399 actions of the other agents. In contrast the agents using D_i were able to do much better. While these agents were not able to do as well as the MCLA algorithm [2], they did exhibit a high level of fault tolerance. This reduced performance level of the

[2]Note that we could not generate the exact clusterings used in the papers [15, 14], due to the randomness in the Metis algorithm. Our results for MCLA and the agent methods were based on a set of clusterings that turned out to be slightly harder, therefore our results cannot be directly compared to the results in this paper.

agent-based solution compared to the graph-based solution here can be attributed to having to combine clusters that were not good representatives of the "true" labels. While the agent-based approached performed well at maximizing ANMI, the agents had no way of knowing if they were also maximizing NMI with respect to the true labels. Here the heuristics in the graph-based approach happened to produce a result that more closely matched the true labels. However, the agent-based solution was able to maintain high levels of performance with a high number of agent faults. When 10% of the agents were not working properly, the system still managed to achieve nearly the same performance as the fully operational system. Even when 50% of the agents were taking random actions, agents using the difference utility were still able to learn to compensate, and achieve much better performance than the system using the global utility.

5. DISCUSSION

Using the difference utility, the results show that agent-based cluster ensembles can perform comparably to the best existing cluster ensemble methods. In addition they are able to achieve this level of performance with relatively little computation as compared to simple greedy optimization methods. Also agent-based method also allow the cluster ensemble problem to be treated as a dynamic optimization problem, significantly increasing the robustness of the system. The agent-based cluster ensembles exhibit a high level of fault tolerance and are able to retain high performance even when 50% of the agents fail. This property allows them to be used in domains that have a high component failure rate, such as in space-based systems or domains with unreliable communication. In addition the agent-based cluster ensembles have the flexibility to be stopped early in domains where a fixed level of performance is needed. This ability can save computational costs, even when the difficulty of the ensemble problem is not known ahead of time.

The implementation presented in this paper assigned an agent to a single cluster and the agent vote involved all the data points in that cluster. However, this way of mapping agents to data points is not necessary to the framework. Depending on the performance requirements one agent could be assigned several clusters or multiple agent could be assigned to different parts of the same cluster. Having alternative agent to cluster mapping may be beneficial if the owners of the clustering sources do not want to reveal their exact clusterings, but still want to collaborate with other data owners to create unified clusterings.

Acknowledgments: We thank Joydeep Ghosh and Alexander Strehl for their help in this project.

6. REFERENCES

[1] A. Agogino and K. Tumer. Efficient evaluation functions for multi-rover systems. In *The Genetic and Evolutionary Computation Conference*, pages 1–12, Seatle, WA, June 2004.

[2] Christopher M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1996.

[3] D. Boley, M. Gini, R. Gross, S. Han, K. Hastings, G. Kary pis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 27(3):329–341, 1999.

[4] S. V. Chakravarthy and J. Ghosh. Scale-based clustering using the radial basis function network. *IEEE Trans. on Neural Networks*, pages 1250–61, September 1996.

[5] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.

[6] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd ed)*. John Wiley & Sons, New York, NY, 2000.

[7] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[8] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: Applications in vlsi domain. In *Proceedings of the Design and Automation Conference*, 1997.

[9] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 42(2):291–307, 1970.

[10] J. Kim and T. Warnow. Tutorial on phylogenetic tree estimation. In *"Intelligent Systems for Molecular Biology"*, Heidelberg, Germany, 1999.

[11] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[12] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[13] T. M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, Boston, MA, 1997.

[14] Alexander Strehl. *Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining*. PhD thesis, The University of Texas at Austin, May 2002.

[15] Alexander Strehl and Joydeep Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. In *Proceedings of AAAI 2002, Edmonton, Canada*, pages 93–98. AAAI, July 2002.

[16] Alexander Strehl, Joydeep Ghosh, and Raymond J. Mooney. Impact of similarity measures on web-page clustering. In *Proc. AAAI Workshop on AI for Web Search (AAAI 2000), Austin*, pages 58–64. AAAI/MIT Press, July 2000.

[17] K. Tumer and D. Wolpert, editors. *Collectives and the Design of Complex Systems*. Springer, New York, 2004.

[18] K. Tumer and D. H. Wolpert. Collective intelligence and Braess' paradox. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 104–109, Austin, TX, 2000.

[19] Ian Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2005.

[20] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.

ReAgents: Behavior-based Remote Agents and Their Performance

Eugene Hung
University of California, San Diego
Computer Systems Laboratory
San Diego, CA
eyhung@cs.ucsd.edu

Joseph Pasquale
University of California, San Diego
Computer Systems Laboratory
San Diego, CA
pasquale@cs.ucsd.edu

ABSTRACT

We present a performance analysis of an agent-based middleware system we have developed based on "reAgents," remotely executing agents that enhance the performance of client/server-based Internet applications. ReAgents simplify the use of mobile agent technology by transparently handling data migration and run-time network communications, and provide a general interface for programmers to more easily implement application-specific logic. This is made possible through the use of *behaviors*, i.e., common patterns of actions that exploit the ability to process and communicate remotely. We model the performance of each of the primary behaviors, and provide an experimental evaluation of reAgent performance.

Categories and Subject Descriptors

D.2.11 [**Software**]: Software Architectures [Remote Agents]

General Terms

Design, Performance, Experimentation, Reliability

Keywords

remote agents, dynamic deployment, design patterns

1. INTRODUCTION

ReAgents support the remote processing of client/server application requests and responses via dynamically deployed mobile code, which carries out work tailored to the particulars of the client device. A reAgent acts as an intermediary between client and server, but maintains the client/server model (Fig. 1). As a simple example, an application running on a client device with a small, limited display would benefit from a remote agent operating at or near the server that filters the rich server data into a spartan response.

Our system is designed with the following goals:

1. Provide the programmer a better way to handle the client's needs and limitations

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/2005 ...$5.00.

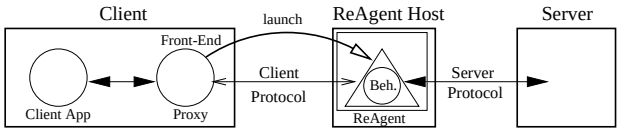


Figure 1: Client-ReAgent-Server Model

2. Be transparent to servers (i.e., do not require modifications to servers)

3. Be easy to program and use

To meet these goals, we propose a remote code mechanism that is more flexible than proxies but less complicated than fully-general mobile agents. We achieve this compromise by adopting a form of "one-shot" mobile agents, simply called a *reAgent* (for "remotely executing agent"). Unlike a general mobile agent, which can move to multiple machines during its computation and retain its state and identity, a reAgent moves exactly once: before it begins execution. The reAgent cannot move after it has begun execution, or even launch other reAgents (only the user may launch reAgents). These restrictions simplify the requirements of the underlying remote code execution mechanisms and reduce the potential for malicious behavior, while still deriving many of the benefits of remote execution.

A reAgent is launched to operate at a remote location because there is an advantage in doing so, such as superior computing or network resources. This location is known as the *reAgent host* (Fig. 1). The reAgent then acts as a customized dynamic proxy for that particular client, and returns the server response to it. This allows for extending the capabilities of the client in a customized fashion. By moving its own provided code to a better location, the client gains extra power to better deal with its limitations and needs. And, the client can represent itself to the server as a standard client via its reAgent, thereby keeping the server code unchanged.

A novel aspect of our approach is that reAgent code is strictly derived from a template library of *behaviors*. Behaviors are useful patterns of processing and communication that are the result of restricting and simplifying the form of movement of reAgents. Examples of general behaviors include filters, monitors, cachers, and collators. Once a general behavior is selected for a reAgent, it is then *specialized* for specific application needs via code parameters.

For the reAgent code to migrate to and run on the reAgent host, some middleware system that supports remote execution must be available. Our reAgent architecture does not specify its own remote execution mechanism. Instead, we leverage existing work in mobile code by translating the launch procedure into the appropriate code movement calls for each system. These movement calls

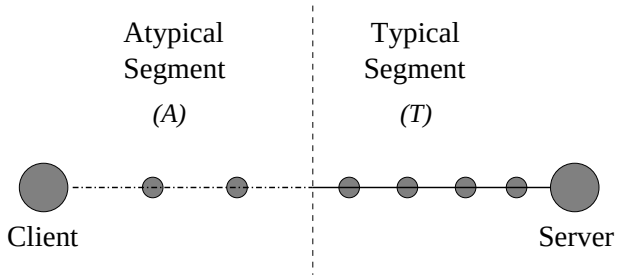


Figure 2: Operative network environment

are translated at run-time when the agent is launched, when the type of middleware on the target reAgent host is specified in a configuration file. The underlying mobile code middleware system is completely transparent to the programmer, who must only understand the simplified reAgent interface. For example, to experiment with reAgents we used a locally-developed and stand-alone Java-based mobile code system [16].

Our experience to date with reAgents has been in the context of Web-based applications [10], and mobile-computing applications [11]. In this paper, we go beyond these studies and analyze the micro-performance of each individual behavior using analytical models, and we present a set of experiments that demonstrate the macro-performance of ReAgents.

The rest of this paper is organized as follows. Section 2 presents a model of the network environment which allows us to reason about the advantages of reAgent location and execution. Section 3 analyzes the general behaviors that are the building blocks of ReAgents. Section 4 presents experimental results. Section 5 describes previous work in this area. Finally, Section 6 presents our conclusions.

2. SYSTEM ENVIRONMENT MODEL

In this section, we present a model of a system environment where an advanced remote code mechanism would be useful.

2.1 Definitions and Assumptions

We define the network environment as a traditional client/server environment, with the network conceptually divided into two segments, *the typical segment* T and the *atypical segment* A (Fig. 2).

T consists of physical network link(s) whose levels of performance and reliability are typically what is observed in the core of the Internet, in general. A, which represents the client device's connection to the Internet, contains links less predictable in their attributes and possibly significantly different from most links in T.

Each network segment is a conceptual unification of its individual composite links, and has gross characteristics of performance and reliability that can be abstracted from the finer-grained characteristics of its individual links. For example, on a link-level basis, the reliability of a segment is only as strong as its weakest physical link. When there is only one route between the client and server, the bandwidth of a segment cannot exceed the least bandwidth of its physical links. Finally, the latency of a segment is at least the sum of the latencies of its physical links.

Within this environment, we make the following assumptions:

- The server, when communicating with a client, assumes clients to be generally powerful devices that communicate with the server through links similar to those in the typical segment. If a client device deviates from this expectation, the server is not expected to handle it.
- The server expects the client to communicate with a predefined protocol P_S, which we refer to as the *server protocol*.
- The client knows the above server protocol P_S.
- The client knows its network limitations and how it deviates from the server's model of a standard, powerful, well-connected client.

2.2 Regions

A *region* is an encapsulation of part of the environment that incorporates the above assumptions into our model. All machines in a region (and the code that runs on those machines) possess knowledge of any deviations from the standard Internet environment. For simplicity, deviations are assumed to be static. In particular, a wireless network is only considered atypical if its operation is distinguishable from that of a wired network.

In addition, machines outside of a region lack any knowledge of any environmental deviations that operate within that region. Note that unlike a firewall, a region does not preclude the ability to communicate with machines in different regions — it merely expresses the boundary of environmental awareness.

Our model defines two regions: the client region and the server region. The client region consists of the client device and the atypical segment, to model the assumption that the client developer knows its deviations (and possibly, where necessary, how to measure them). Examples of such deviations include:

- Physical limitations of client device (small display, inadequate memory)
- Atypically low bandwidth in connection to Internet
- Atypically unreliable communications
- Atypically insecure communications
- Excessive latency between client and server

Meanwhile, the server region consists of the server machine(s) and the typical network segments that connect them to the atypical segment. This models the server's lack of knowledge about any client problems. (For purposes of simplicity and clarity, we limit the subsequent discussion to a server region containing a single server, but multiple servers are also supported by this model.)

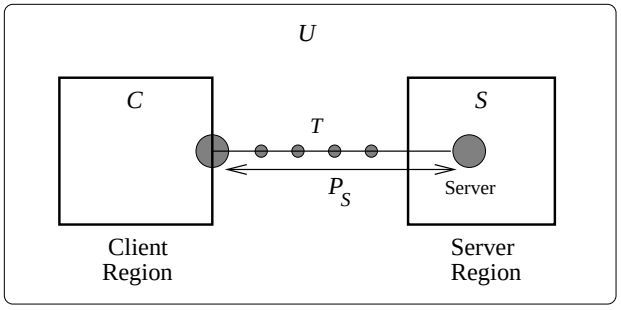


Figure 3: Server view of environment

Code in the server region operates as if the network environment were as depicted in Fig. 3. This view consists of the exterior of the client region, $U-C$ (which contains the server region S), and the physical network links between C and S that make up the client-server network connection (which is categorized as a typical network segment). Thus, server code is only required to assume that

there is a client, operating on a machine remote to the server, that connects and makes requests to the server using the server protocol P_S. It does not know about the atypical network segment at all.

In contrast (Fig. 4), code operating in the client region operates in an environment consisting of the exterior of the server region $U-S$ (which contains the client region C). The code developer is not required to understand either the implementation of the server architecture or the details of the typical segment, but only the server protocol P_S and the point of communication where the server can communicate with the client using P_S.

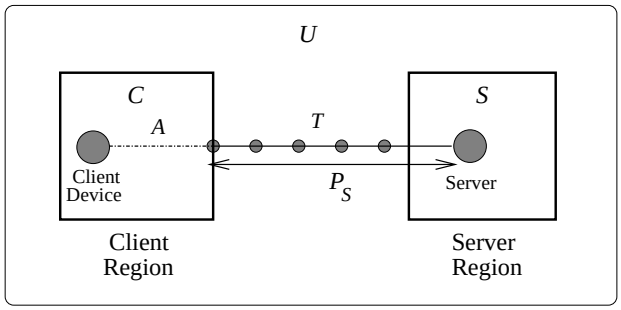


Figure 4: Client view of environment

2.3 Location of Customizing Logic

Given this model, and the concept of using remote code for customizing application performance, it is important to decide where the customizing logic should be placed. There are three possible areas where such customizing logic is based. They are:

- Within the client region
- Within the server region
- Within the network somewhere in between the client and server regions

Each of these three regions has been used in previous solutions for customizing performance. If the customizing logic is placed within the network, both client and server will have to deal with this change in their world view, making it hard to deploy. Also, the overhead of the mechanisms supporting the customizing logic is potentially incurred by all applications, regardless of whether they use them or not, thus lowering efficiency.

Meanwhile, if the customization logic is based at the server, then each change in a client causes a change in every server. This solution does not scale well: as the heterogeneity of client devices (represented by the addition of different client regions) increases, support required in the server universe also increases, as each server must add customization logic to handle each new client region.

Thus, we have chosen to base the location of the customizing logic, as a **reAgent**, in the client region. The reAgent is originated by the client and placed on a machine that resides in the typical segment, beyond the atypical segment. Once the reAgent is placed, we bound the client region to have the client device at one end and the customizing logic at the other end, so that the atypical segment is masked from the server. (Fig. 5)

From the model, we can see that the reAgent, being in the client region, is exposed to the problems of the client region, but that the server is not. Instead, the server treats the reAgent as the actual client, as they are both part of the opaque client region. Thus, the server need not change when new client problems arise, promoting deployability and scalability. Meanwhile, the client is still able

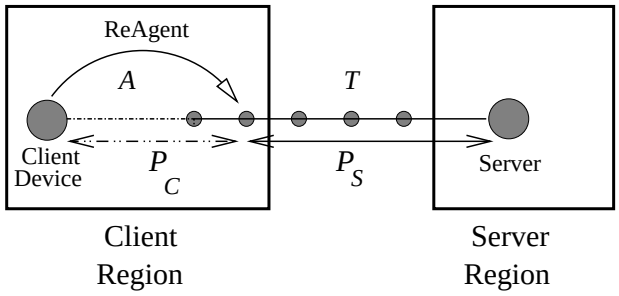


Figure 5: Bounding of Client Region

to communicate with the server by providing the reAgent with an understanding of the server protocol P_S. Moreover, with the addition of the reAgent as an intermediary between client and server, the client can now communicate with the reAgent using a custom *client protocol* P_C. Such a protocol could be used to ameliorate communication problems over the atypical segment, or provide the client with run-time control over the reAgent.

3. BEHAVIORS

We developed the definition of behaviors based on their usefulness in the design of reAgents. A behavior is "useful" if it exhibits benefits that are derived from a reAgent's ability to operate remotely. These benefits come from some combination of, but not limited to, the following: (1) use of remote computational resources; (2) avoiding or minimizing the effects of a problematic portion of the network (high delay, low bandwidth, low reliability, etc.); (3) ability to act autonomously on behalf of the client in a customized fashion.

The following sections catalog the useful behaviors we have identified. For each behavior, we present a description of the behavior, an outline of its base logic, an example of its use, and an analysis of its performance.

3.1 Filter

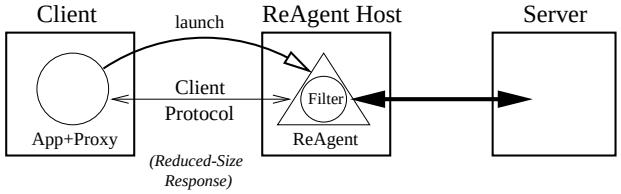


Figure 6: The Filter Behavior

Description. The Filter behavior (Fig. 6) is used whenever a server response needs to be reduced (in size) before reaching the client. The reAgent interposes itself between the client and server, filters the server responses into a smaller, possibly more suitable format for the client, and sends this filtered result back to the client. The CL (customizing logic) is the application-specific algorithm that defines *how* to reduce the data.

Base Logic. The reAgent passes the request through to the server, runs the filtering algorithm on the server response, and sends the new response back to the client.

Application. The Filter behavior is designed for scenarios where the server data is too large for the client. A common scenario involves browsers on clients with limited capabilities, such as small battery-powered wireless devices (e.g., PDAs). General features of such a device include limited network bandwidth as well as low-fidelity rendering of data, so filtering by removing extraneous or unusable data before sending it to the browser would reduce required bandwidth and reduce delay without significantly impacting the perceived quality of the data.

Analysis. In this analysis and those that follow, we assume that network latency (i.e., propagation time) and network message processing overhead are negligible (relative to the other factors we consider). All overheads that are attributable to the the reAgent itself, e.g., its launching time, are encapsulated as a single variable (λ).

One way a filter is effective is if it reduces end-to-end server-to-client delay. In straight client/server, the delay (D_{cs}) to return a file of size $\mathcal{S}$ is the transmission time given by file size divided by client-server bandwidth (B_{cs}):

$$D_{cs} = \frac{\mathcal{S}}{B_{cs}}$$

The delay as a result of using reAgents is equal to the the transmission time between the the server and reAgent ($\frac{\mathcal{S}}{B_{rs}}$) plus the processing time for the filter ($\mathcal{P}_{filter}$), plus the overhead of using the reAgent (λ), plus the transmission time between the reAgent and the client ($\frac{\alpha\mathcal{S}}{B_{cr}}$), where α is the percentage of original data left over from the filter:

$$D_{crs} \quad = \quad \frac{\mathcal{S}}{B_{rs}} + \mathcal{P}_{filter} + \lambda + \frac{\alpha\mathcal{S}}{B_{cr}}$$

To compare D_{cs} and D_{crs}, we calculate the *speedup*, which represents the percentage improvement of the reAgent approach. When the speedup is positive, reAgents are superior; when the speedup is negative, traditional client/server implementations are superior.

The speedup is derived from the following equation:

$$speedup = 1 - \frac{D_{crs}}{D_{cs}} \tag{1}$$

Plugging in the values of D_{crs} and D_{cs} into the speedup equation above, and setting B_{cr} equal to B_{cs} (assuming that the bandwidth of a set of links is equal to the bandwidth of the slowest individual link) results in a speedup of

$$speedup = \frac{(1-\alpha)\mathcal{S}/B_{cs} - (\mathcal{S}/B_{rs}) - \mathcal{P}_{filter} - \lambda}{D_{cs}}$$

If we express the ratio of the bandwidths as $\rho = \frac{B_{cs}}{B_{rs}}$, then $speedup > 0$ when

$$(1 - \alpha - \rho)\frac{\mathcal{S}}{B_{cs}} - \mathcal{P}_{filter} \quad > \quad \lambda$$

Thus, reAgents become more advantageous as

- α decreases (the filter reduces more data)
- ρ decreases (the bandwidth ratio of the client/reAgent to reAgent/server network segments becomes much smaller)
- $\mathcal{P}_{filter}$ decreases (the filtering algorithm processes more quickly)
- B_{cs} decreases (the overall bandwidth is small)
- $\mathcal{S}$ increases (there is more data to filter)
- λ decreases (overhead of using reAgent decreases)

So if client-reAgent bandwidth drops, or original data size gets larger, the filtering reAgent becomes more effective.

3.2 Monitor

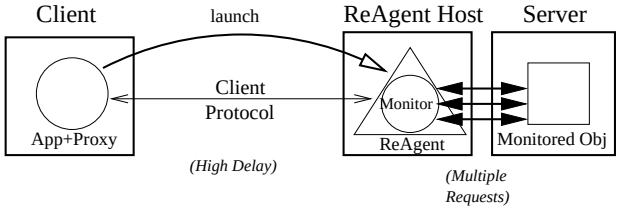


Figure 7: The Monitor Behavior

Description. The Monitor behavior (Fig. 7) is designed for use in applications that have a need to frequently examine the state of a remote object (on a far-away server) until a certain state is observed. The calculation of the next monitor attempt, plus the response evaluation function, forms the CL.

A reAgent that uses the monitor behavior is placed on a site close to the object that is being monitored, for the purpose of reducing the time of receiving a critical state change and sending the trigger action to the server. This is important for applications that require a real-time response to sudden changes in environment, such as a stock ticker or online bidding auction.

Base Logic. The behavior repeatedly calculates the next time to query the server, queries the server at that time, and then checks to see if a trigger state has been reached. Once the trigger state is reached, monitoring is terminated and the response that triggered the state change is returned.

Application. A simple example of an application for a Monitor involves intelligent auto-refresh of a Web browser. Many pages auto-refresh at fixed intervals. A Monitor can bypass the automatic refresh and refresh at its own customized rate. This can be advantageous when the Monitor is sensitive to network conditions and adjusts the rate depending on the amount of traffic. More importantly, the intelligent auto-refresh only updates the client when the server changes, and will not force a refresh when the data remains unchanged, saving bandwidth.

Analysis. To evaluate the monitor, we assume that the monitor will send a message to the server n times before the trigger condition is satisfied.

Under straight client/server, the total delay between sending a request to the server and processing its reply is

$$T_{cs} = n \times D_{cs}$$

where n = the number of queries before the trigger condition is satisfied.

With a reAgent, the total delay is

$$T_{crs} = D_{cr} + nD_{rs} + \lambda$$

as the delay between client and reAgent is only paid once. Equation 1 gives a reAgent speedup of

$$speedup = 1 - \frac{D_{cr} + nD_{rs} + \lambda}{nD_{cs}}$$

Solving for $speedup > 0$, we find that reAgents are better for monitoring when

$$(n-1)D_{cr} > \lambda$$

Thus, reAgents become more advantageous as

- n increases (more queries before trigger state is reached)
- D_{cr} increases (more delay between client and reAgent that is avoided)
- λ decreases (overhead of using reAgent decreases)

Therefore, many queries, or high delay between the client and the reAgent, points to using a monitor reAgent to gain a performance advantage.

3.3 Cacher

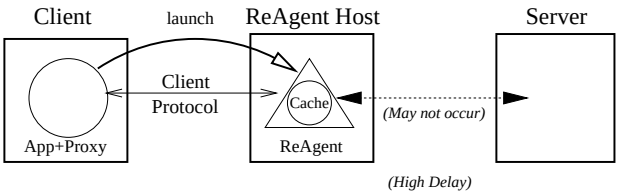


Figure 8: The Cacher Behavior

Description. The Cacher behavior (Fig. 8) is used for storing recently retrieved server data at a nearby location with the expectation that it will be accessed again, thus improving future performance. When previously retrieved data is requested again, the nearby stored copy is retrieved instead of the distant original. The cache replacement policy forms the CL. This behavior is especially useful for applications that have frequent yet identical requests to remote servers, such as occurs in Web browsing.

Base Logic. This behavior uses customized logic on the request input to decide whether or not to pass along the request to the server. If the request has not been made recently, the behavior generates a *key* that gets associated with the request, and then uses the request to contact the server. When the server responds, the behavior associates the data in the response to the key of the request and stores both items in a database, i.e., the cache, before outputting the response data. When a request is made that matches a key in the cache, the behavior will bypass sending the request to the server and immediately return the associated cache data.

The behavior is in charge of inserting, removing, and retrieving data contained within the cache. Insertion of data into the cache happens whenever the server sends the behavior a response. Cached data and its corresponding key are removed whenever the amount of storage allocated to the cache begins to run out, or by special order of the client. Data is retrieved from the cache when the client request key matches a key within the cache. While the behavior defines these general actions, particulars regarding cache policy (such as which cache entries to replace first when the cache is full) are supplied as part of the CL.

Application. Caching of frequently accessed Web pages is so beneficial to performance that most major Web browsers support some form of caching. With no intermediate hosts, the server data is stored on the client device. While storing the cached data on the client device is optimal for minimizing network delay, some client devices have such small amounts of memory that cache performance is seriously degraded by running locally. These resource-poor clients would benefit greatly from moving the cache from the client device to a nearby location with sufficient resources.

In such situations, a reAgent with a Cacher type of behavior can be created, along with a client-specified cache replacement policy and size, and launched to the nearby machine to effectively operate a cache for the browser.

Analysis. To evaluate the cacher in both scenarios, end-to-end delays are compared.

Under straight client/server, the delay between sending a request to the server and processing its reply is simply D_{cs}. Under a caching reAgent, the delay between sending a request to the server and receiving its reply depends on whether the item requested is in the cache. A delay between client and reAgent is always incurred. When the item is not found in the cache, a delay is also incurred between the reAgent and server. This can be expressed as

$$D_{crs} = D_{cr} + pD_{rs} + \lambda$$

where p is the probability of a cache miss.

The speedup of a reAgent Cacher over client/server is

$$speedup = 1 - \frac{D_{cr} + pD_{rs} + \lambda}{D_{cs}}$$

and $speedup > 0$ when

$$(1-p)D_{rs} > \lambda$$

So caching is better when

- D_{rs} increases (the delay from the network segment skipped by the cache is higher)
- p decreases (fewer cache misses)
- λ decreases (overhead of using reAgent decreases)

Thus, a reAgent cache outperforms client/server when the delay between reAgent and server is high enough, or the miss rate is low enough to overcome reAgent overhead.

3.4 Collator

Description. The Collator behavior (Fig. 9) transmits the same message to multiple servers from a remote location, and waits until a wait condition, specified by the client, is satisfied. Afterwards, the responses are sent to an application-specific function that produces a result for the client (collating).

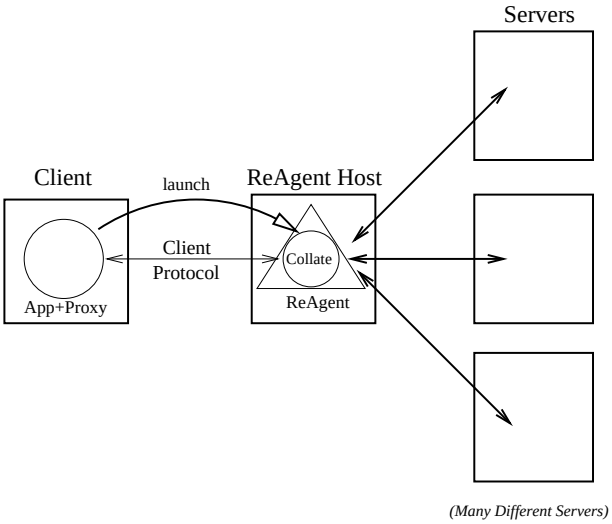


Figure 9: The Collator Behavior

Base Logic. The message is sent once to the reAgent, which then transmits it multiple times, once for each server. The reAgent then waits for responses from the servers in an application-specific fashion. For example, the reAgent may only wait for the first response from any server, or for some bounded number of responses, or even wait for responses from the servers within a timeout period. After waiting is completed, the server responses are collated in an application-specific way (as part of the CL), and the result is sent to the client.

Application. A typical Web application that exhibits this behavior is a comparison agent that queries different servers with the same question and returns the "best" result. While many services for finding the best price of an item already exist on the Web, they do not perform correctly if a server is not known or supported by the query service, or if the user is more concerned about some other attribute, such as delivery time or seller reputation, that the service does not support.

Analysis. For ease of analysis, the following assumes that the messages are sent serially, not in parallel.

Under straight client/server, the delay for completing requests to n servers is

$$D_{cs} = n \times d_{cs}$$

where d_{cs} = the average delay between client and server.

Under a collating reAgent, the time is

$$D_{crs} = nd_{rs} + \lambda$$

where d_{rs} = the average delay between reAgent and server.

Applying the speedup equation (1) gives a speedup of

$$speedup = 1 - \frac{nd_{rs} + \lambda}{nd_{cs}}$$

and a performance win for reAgents when

$$(n-1)d_{cr} > \lambda$$

where d_{cr} = the average delay between client and reAgent.

Thus, for any advantage to be gained, $n > 1$ (confirming the obvious). For each n beyond 1, a collating reAgent reduces delay by d_{cr}, which logically corresponds to traversing the network segment between the client and the reAgent, per additional server. If this amount is greater than the overhead from using the reAgent, the reAgent performs better than client/server.

4. EXPERIMENTS

Previous papers[10, 11] quantified the low overhead and utility of reAgents used for data compression and image filtering, based upon the Filter behaivor. Here, we describe an experiment that validates the theoretical advantages of the Monitor behavior.

A primitive stock server and client application were developed. The server outputs a ticker, showing the fluctuating price of a stock, and is programmed to accept only the first buy order it receives per run (i.e., only 1 share is available at this listed price).

The Monitor behavior was then used as the basis for a reAgent that would connect via network sockets to the stock server to watch stock prices and issue a buy order. The Monitor behavior was chosen because this application requires multiple communications between the reAgent and stock server. The reAgent monitors the ticker and sends a buy request when the price of a certain stock falls to a certain value (both chosen by the user as input arguments). The decision-making component is a custom algorithm that acts as the CL for the Monitor behavior.

In the following experiments, the performance metric used was the time it took for the server to receive a "buy" command after the stock hit the target price, essentially a measure of the total latency between the decision-maker and the server, plus the CPU time alloted to the decision-maker.

4.1 Response Time

The first experiment was to evaluate the relative response-time performance of three network application paradigms :

1. Traditional RPC
2. ReAgent-based computing
3. Service-based customization

To introduce a non-trivial latency, the client machine, *federation*, a Sun Ultra 1, was stationed at CMU in Pittsburgh, approximately 2500 miles away from the stock server machine, *ursus*, a Sun Ultra 10, in San Diego. The results, after 10000 iterations for each paradigm, are summarized in the following table:

Table 1: Comparison of Response Time for Various Paradigms

Paradigm	Response Time (*ms*)			95% Confidence Interval
	Min	Max	Mean	
Traditional	71	3474	79.6	$\pm$ 1.8 ms
reAgent	1	14	1.161	$\pm$ 0.008 ms
Service	0	11	0.051	$\pm$ 0.005 ms

The confidence intervals show that the averages are statistically significant. These results say, not surprisingly, that the best performance occurs if the server can be hard-wired with the action that the client requests. While this might be true for simple operations like limit-order stock purchases, in practice, not all servers will be able to anticipate every need of the client, such as when the user has an unusual stock-trading algorithm to implement.

Comparing the non-hardwired methods of traditional vs. reAgent, the traditional method suffers severe performance penalties as well as a highly variable response time. The high variance is a result of intermittent network congestion and timeouts, which the reAgent, running near the server, can ameliorate.

From this experiment, it can be concluded that a Monitor-based reAgent application achieves similar, although slightly inferior, performance when compared with a server-customized service. Of course, the cost of building applications with a similar level of performance with reAgents is much smaller than requiring all servers to hardwire their services for each user. Attempts to create flexible applications without reAgents result in highly variable and inferior performance due to the high-latency communications between the decision-making algorithm and the server.

4.2 Application Performance Comparison

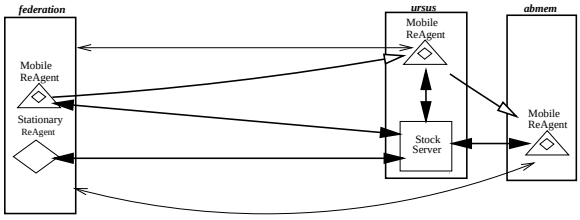


Figure 10: Stock Traders Setup

An application-based comparison was performed as a supplement to the previous experiment. The comparison centered on the Monitor-based stock trading reAgent described in the previous section. This reAgent was run in sets of two. In each set, one reAgent remains at the client (the stationary reAgent), while the other reAgent (the mobile reAgent) is potentially launched to a different host to improve application performance. Each Trader's goal is to wait until the stock hits a pre-determined low price, and then buy it before their opponent.

The experiment was conducted with three sets of reAgents. The first set has both reAgents begin at *federation* (the client), to demonstrate the worst case of no supporting reAgent servers. The next set has the mobile reAgent moves to *ursus*, the stock server itself, and results are compared. The final set has the mobile reAgent move to *abmem*, a machine on *ursus*'s LAN that acts as a reAgent host. The last experiment is important because it highlights the deployable nature of reAgents; one can neither assume that every server in the world will support reAgents, nor that servers will always have the capacity to host reAgents. However, it is reasonable to assume that an independent reAgent host will be nearby.

The experiment showed that the mobile reAgent beat the stationary reAgent every time when operating from a closer location. The full results can be seen in Table 2.

Table 2: Mobile vs. Stationary ReAgent

Mobile reAgent Host	Mobile reAgent Success Rate	95% Confidence Interval
client	50.8%	$\pm$ 1.4%
server	100.0%	$\pm$ 0.0%
reAgent host	100.0%	$\pm$ 0.0%

The 100% success rate in the cases where the reAgent migrated closer demonstrated that the mobile reAgent could parlay the theoretical performance benefits of reAgents into a superior, flexible application. In the worst-case scenario with no supporting reAgent servers, the reAgent still functioned on a relatively equal footing with a normal, traditional trader (represented by the stationary reAgent). However, the mobile reAgent was able to use its ability to move when closer reAgent hosts were available, allowing it to improve its performance along with its location. The reAgent allows this improvement to be easily customizable, merely by changing its destination (a line of text). As users have different requirements — for example, running directly on the server may be significantly more expensive, so a grad student's trading application may only give a destination of a cheaper reAgent server — reAgents allow users to scale the performance effectiveness to a level appropriate with their needs and abilities.

5. RELATED WORK

The idea of enhancing client applications with remote code is not new, but previous efforts have been divided on how and where to provide this functionality. The *active networks* approach [23] argues for putting customizing logic at the network level. While various active networks [26, 19, 1] are able to effectively support a wide variety of client devices, they do so at the expense of deployability [6].

Another customization solution lies in the use of proxies, which act as intermediaries between client and server. While traditional proxy applications concentrate on caching and firewalls [14], some have focused on actively customizing application behavior (dynamic proxies) [4, 24, 15].

Another approach is to attempt to have servers adapt to the specifics of each individual client. Server-provided CGI scripts/forms, and the group of technologies bundled into the Open Mobile Alliance (OMA) [17] are two examples of this type of approach, where the server programmer is responsible for anticipating common client problems and catering to them. To be less dependent on the server, researchers have developed customizers with more client-side support [28, 22, 3].

A significant area of past research in client-based customization has been based upon *mobile agents* [5]. Mobile agents are pieces of customizing logic that have a persistent identity, moving around the network to multiple sites [25]. The D'Agents project [8] and IBM Aglets Workbench [13] are prominent examples from industry and academia, respectively, of systems that support the execution of mobile agents. A fuller description of these and other important agent systems can be found in [9].

Our reAgent approach is a middle-ground solution, based on a one-shot remote code mechanism that is more flexible than proxies but less complicated than fully-general mobile agents. This model is similar to that of remote evaluation [21]. Several advantages arise from limiting movement to one hop. By avoiding some of the security issues introduced by code that can roam from site to site, infrastructural support is simplified. Also, with a stationary remote agent acting on its behalf, the client gains the benefits from remote execution without adjusting its client/server architecture: the reAgent acts as a server by taking requests and returning responses. Finally, technical problems associated with maintaining and updating program state during migration are avoided, without losing much functionality, a view supported by [12].

Finally, behavior templates are similar to the idea of "design patterns." In [27], the authors describe problems facing the development and deployment of mobile agent (and mobile code) applications. Consequently, there has been some work attempting to extend the idea of design patterns to mobile code [7]. Design patterns have previously been proposed for use in designing mobile agent applications [2, 18, 20].

6. CONCLUSION

In this paper, we described the behavior-based architecture and performance of reAgents. ReAgents are especially applicable to situations where clients are resource limited in their computing or network access capabilities. A reAgent effectively extends the client's reach into the network, and derives its power by performing remote operations that (1) minimize the effects of a problematic network path (2) gain an advantage by operating close to one or more servers, (3) take advantage of remote resources (e.g., computational, memory) that are relatively more powerful than those locally available at the client, and (4) act autonomously on behalf of the client in a customized fashion.

The use of reAgents allows us to derive benefits of mobile agent systems — specifically their support for dynamically-located remote execution, limited to one-shot movement — in what we believe to be an easy-to-program form. Key to this simplicity is the transparent handling of data migration and run-time network communications. This is a direct result of the reAgent programming model: reAgents are composed of one or more behaviors, i.e., abstractions for common remote patterns of action, each of which can be specialized with custom logic that allows reAgents to be tailored to their applications. We presented four such general behaviors — filtering, caching, monitoring, and collating — and using analytical models showed the conditions under which performance will improve when compared to the straight client/server model. We also presented empirical results from experiments that provide evidence of the performance benefits of reAgents.

7. REFERENCES

[1] D. S. Alexander, W. A. Arbaugh, et al. *The SwitchWare Active Network Architecture*, IEEE Network, May/June 1998.

[2] Y. Aridor and D. B. Lange, *Agent Design Patterns: Elements of Agent Application Design*, Proc. 2nd Int'l Conference on Autonomous Agents, May 1998

[3] R. Barrett and P. P. Maglio, *Intermediaries: New places for producing and manipulating web content*, Proc. 7th Int'l World Wide Web Conference, Brisbane, Australia, 1998.

[4] P. Cao, J. Zhang and K. Beach, *Active Cache: Caching Dynamic Contents (Objects) on the Web*, Middleware '98, Sep. 1998.

[5] D. Chess, B. Grosof, C. Harrison, D. Levine, C. Parris and G. Tsudik, *Itinerant Agents for Mobile Computing* IEEE Personal Communications, 2(5): 34–39, 1995.

[6] M. Fry and A. Ghosh, *Application Level Active Networking*, Computer Networks, 31(7): 655–667, 1999.

[7] E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Oct. 1994.

[8] R. Gray, G. Cybenko, et al., *D'Agents: Applications and Performance of a Mobile-Agent System*, Software - Practice and Experience, 32(6):543–573, May 2002.

[9] R. Gray, G. Cybenko, D. Kotz and D. Rus, *Mobile agents: Motivations and State of the Art*, Handbook of Agent Technology, AAAI/MIT Press, 2002.

[10] E. Hung and J. Pasquale, *Web Customization Using Behavior-Based Remote Executing Agents*, Proc. 13th Int'l World Wide Web Conference, May 2004.

[11] E. Hung and J. Pasquale, *Using Behavior Templates To Design Remotely Executing Agents for Wireless Clients*, Proc. 4th IEEE Workshop on Applications and Services in Wireless Networks, August 2004.

[12] D. Kotz, R. Gray and D. Rus, *Future Directions for Mobile-Agent Research*, IEEE Distributed Systems Online, 3 (8), Aug. 2002.

[13] D. Lange, M. Oshima, G. Karjoth and K. Kosaka, *Aglets: programming mobile agents in Java*, Proc. of Worldwide Computing and its Applications (WWCA'97), Lecture Notes in Computer Science, Vol. 1274, 1997.

[14] A. Luotonen and K. Altis, *World-Wide Web proxies*, Computer Networks and ISDN Systems, 27(2): 147–154, 1994.

[15] P. Mohapatra and H. Chen, *WebGraph: A Framework for Managing and Improving Performance of Dynamic Web Content*, IEEE Journal On Selected Areas in Communications, 20 (7), Sep. 2002.

[16] T. Newhouse and J. Pasquale, *Java Active Extensions Scalable Middleware for Performance-Isolated Remote Execution to Enhance Wireless Network Applications*, Elsevier Computer Communications Journal, Vol. 28 (14), Sept. '05.

[17] Open Mobile Alliance, *Affiliate Specifications*, http://www.openmobilealliance.org/tech/affiliates/index.html

[18] A. Silva and J. Delgado, *The Agent Pattern for Mobile Agent Systems*, European Conference on Pattern Languages of Programming and Computing, EuroPLoP, 1998

[19] S. da Silva, D. Florissi and Y. Yemini, *Composing active services in NetScript*, DARPA Active Networks Workshop, March 1998.

[20] A. Singh, R. Sankar and V. Jamval, *Design Patterns for Mobile Agent Applications*, Workshop on Ubiquitous Agents on Embedded, Wearable, and Mobile Devices, Italy, 2002.

[21] J. W. Stamos and D. K. Gifford, *Remote Evaluation*, ACM Transactions on Programming Languages and Systems, 12(4):537–565, March 1990.

[22] J. Steinberg and J. Pasquale, *A Web Middleware Architecture for Dynamic Customization of Content for Wireless Clients*, Proc. 11th Int'l World Wide Web Conference, Honolulu, Hawaii, USA, May 2002.

[23] D. Tennenhouse and D. Wetherall, *Towards an active network architecture*, Computer Communications Review, 26(2): 5–18, Apr. 1996.

[24] A. Vahdat, M. Dahlin, T. Anderson and A. Agarwal, *Active Names: Flexible Location and Transport of Wide-Area Resources*, Proc. USENIX Symposium on Internet Technologies and Systems (USITS), October 1999.

[25] Y. Villate, A. Illaramendi and E. Pitoura, *Mobile and External Storage Space Using Agents for Users of Mobile Devices*, Workshop on Ubiquitous Agents on Embedded, Wearable, and Mobile Devices, Bologna, Italy, 2002.

[26] D. Wetherall, J. Guttag and D. Tennenhouse, *ANTS: A toolkit for building and dynamically deploying network protocols*, IEEE OPENARCH '98, April 1998.

[27] M. Wooldridge and N. Jennings, *Pitfalls of Agent-Oriented Development*, Proc. 2nd Int'l Conference on Autonomous Agents, 1998.

[28] B. Zenel, *A proxy based filtering mechanism for the mobile environment*, PhD Thesis, Columbia University, 1998.

MLBP: MAS for Large-Scale Biometric Pattern Recognition

Ram Meshulam Shulamit Reches Aner Yarden Sarit Kraus
{meshulr1,reches,yardena,sarit}@cs.biu.ac.il
Department of Computer Science
Bar-Ilan University, 52900
Ramat-Gan, Israel

ABSTRACT

Security systems can observe and hear almost anyone everywhere. However, it is impossible to employ an adequate number of human experts to analyze the information explosion. In this paper we present an autonomous multi-agent framework which, as an input, obtains biometric information acquired at a set of locations. The framework aims in real-time to point out individuals who act according to a suspicious pattern across these locations. The system works in large-scale scenarios. We present a scenario to demonstrate the usefulness of the framework. The goal is to point out individuals who visited a sequence of airports using face recognition algorithms. Simulation results show a high overall accuracy of our system in real-time.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms,Theory

Keywords

privacy, safety and security in agent systems, cooperative distributed problem solving in agent systems

1. INTRODUCTION

In this paper we address the general domain of **L**arge-scale **B**iometric **P**attern recognition (LBP). LBP problems include situations where there is a stream of biometric information acquired at a set of locations e.g., a set of cameras positioned in several airports. The goal of LBP problems is to point out individuals who act according to a predefined suspicious pattern in real-time. The combination of multiple locations and streams of input, inaccuracy of biometric tests [1] and real-time constraints make the LBP problems hard to solve, and ideal candidate for MAS implementation [2]. A representative scenario is presented below.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1.1 Problem Description

In this paper we present a framework of a multi-agent solution for the LBP problems. We then present an implementation of this framework for a specific problem we call **The airport problem**. Assume a set of L airports denoted $ap_1, ap_2, ..., ap_L$. Our goal is to identify and detain at airport L every passenger who flew from airport 1 to airport 2.... to airport L. At each airport there are cameras which take one picture of each passenger.

There are two boolean comparison algorithms, namely ca_1 and ca_2. As an input each comparison algorithm receives two pictures. The comparison algorithm returns true if the pictures match, i.e. both pictures belong to the same person. Otherwise the algorithm returns false. We assume the algorithms are independent. Each comparison algorithm has a known *false acceptance rate* (FAR) which is the probability that the algorithm will receive two pictures which do not belong to the same person and it will return true. Each comparison algorithm has a known *false reject rate* (FRR) which is the probability that the algorithm will receive two pictures which belong to the same person and it will return false. We also know that the first algorithm has a lower FAR and the second algorithm has a lower FRR: $FAR_{ca_1} < FAR_{ca_2}$ and $FRR_{ca_2} < FRR_{ca_1}$. We define a *picture-set* of size k as a set of pictures taken from k airports. A key constraint of LBP is that the suspects do not need to cooperate with the system.

1.2 Difficulties and challenges

We would like to point out some of the attributes make LBP problems difficult. The examples refer to the airport problem, although all attributes exist in the LBP domain in general. Firstly, this is a large scale problem. Assume we would like to capture every person who visited two different airports ap_1 and ap_2. Also assume there are 1000 people in each airport. The agent must create $1,000 \times 1,000 = 1,000,000$ picture-sets of size two and use a comparison algorithm to compare each pair of pictures. Note that the cameras' location can ensure that only pictures of passengers who came on a certain flight are taken, so 1000 images per airport is not an unrealistic scenario.

Moreover, biometric comparison algorithms are imperfect. For example, current state-of-the-art face recognition algorithms have an error margin that is unacceptable in large-scale situations [1]. Suppose all the pictures in the previous example are of different people. Even an algorithm with an

Algorithm 1 MLBP framework: local module

Local-module(Path p, item i, path q)
1 Let r=**Phase1**(p, i, q)
2 If r='borderline' Then return **Phase2**(q, i, q')
3 return r

Phase1(Path p, item i, path q)
1 Let q =**AddItemToPath**(p, i, ca_1)
2 Let m=$\binom{L}{2} - LT$
3 If $q^- > m$ Then return 'prune'
4 If (($|q| < L$) AND ($q^- \leq m$)) Then
4.1 Return 'continue'
5 If ($q^- < m$) Then return 'trigger alarm'
6 return 'borderline'

Phase2(Path p)
1 Let q be an empty path
2 Let m=$\binom{L}{2} - HT$
3 For each item i in $p.I$
3.1 q=**AddItemToPath**(q, i, ca_2)
3.2 If $q^- > m$ Then return 'prune'
4 return 'trigger alarm'

FAR of 1% will, on average, mistakenly positively identify 10,000 out of the 1,000,000 picture-sets. Suppose there is a person whose picture was taken at ap_2. This picture is matched 1,000 times, each time against different pictures acquired at ap_1. The probability that this picture will appear in a set that is falsely accepted is very high. Using this algorithm will make all passengers of the second airport be (probably falsely) detained.

2. MAS SOLUTION FRAMEWORK

We present a solution to the LBP problem assuming we have two biometric comparison-algorithms. The solution uses multiple agents to incrementally build candidate suspects. At each step, the algorithm tries to prune some of the candidates. Each agent consists of a local-module and an interaction-module.

The main information structure which is transferred between the agents is *path* defined as $p = \{I, R\}$. $I = \{i_1, i_2, ...i_k\}$ is a set of k items (e.g., face images of passengers). R is a set of the comparison results between any two items in I. Given a path p, we denote p^- as the number of negative comparison results in $p.R$. Similarly, we define p^+ as the number of positive comparison results in $p.R$. We also denote the number of items in $p.I$ as $|p|$. There are $p^+ + p^- = \binom{k}{2}$ comparison results in a path of length k. A path p and an item i can be joined together to create a new path of size $|p| + 1$. The method *AddItemToPath* gets a path p and an item i and returns a new path q. $q.I = \{p.i \cup i\}$ and $q.R$ holds $p.R$ along with the comparison results of items in $p.I$ and i.

Suppose we have a perfect comparison algorithm, with no FAR and FRR. The system should trigger an alarm only when it detects a path of size L which has the maximum number of positive comparison results, i.e. $p^+ = \binom{L}{2}$ and $p^- = 0$. For example, If $L = 3$, the system should trigger an alarm only if $ca(1,2) = ca(1,3) = ca(2,3) = true$. In actuality, however, comparison algorithms have FRR and we must consider cases where all items match, but some of the comparisons are falsely rejected. For the first comparison algorithm ca_1 we define a threshold LT where $0 \leq LT \leq \binom{L}{2}$. The algorithm triggers an alarm when a path of size L is created and $p^+ > LT$. In other words, the algorithm allows a path to have $p^- < \binom{L}{2} - LT$ false comparisons and still treats it as a path with matching items which might lead to a target person. Similarly we define a threshold HT for ca_2.

Each agent is responsible for one biometric information source (e.g., a camera). An agent receives new information from two sources. The first source is from his biometric information unit (e.g., a passenger picture). The agent processes the acquired item and creates a path of size one. The second source of information is forwarded paths from other agents. Each agent holds a database of paths PDB in which it stores every new path it receives.

Algorithm 1 describes the local module of the MLBP framework. The local-module is embedded in each agent. As an input the module receives a path p and a biometric item i. It returns a string that classifies the joint path q which contains all items in p and i. The local module has three possible return values:

- 'prune' - There is no point in keeping the joint path.
- 'continue' - The joint path has a potential to become a target path.
- 'trigger alarm' The joint path is a target path and the system must notify about it.

The module may return 'continue' only if the size of the joint path is smaller than L. It may return 'trigger alert' only for full paths - paths of size L. Comparing a pair of pictures is time consuming. Thus, the algorithm uses ca_1 alone to classify the new path in most cases (lines 3-5 in function **Phase1**). The only scenario in which ca_2 is used is a borderline scenario. A borderline scenario occurs when a full path ($|q| = L$), has exactly LT positive results. The algorithm then uses ca_2 (function **Phase2**). First the algorithm creates a new path which contains all items in q, but this time it uses ca_2 to compare all pairs of items. The same classification rules are applied for q', but this time the threshold is HT rather than LT.

Though ca_2 (which is used in phase 2 only) has a higher false alarm ratio, it is of less concern since at this point the number of paths has already been significantly reduced. Conversely, the low FRR of this algorithm makes it very effective in verifying the identity of the true suspect. Phase 2 is significantly more time consuming than phase 1. In phase 1 $p.R$ already contains comparison results of all items in p (using ca_1). The new item i is then compared with items in p, which totals $L - 1$ comparisons (worst case). In phase 2, the algorithm uses $\binom{L}{2}$ comparisons (worst case) to build a path from scratch using ca_2.

Also note that the pruning made by the local-module is lossless. It is lossless in the sense that a system which uses the module will have the same accuracy with and without the pruning mechanism. This fact allowed us to ignore the pruning mechanism when we analyzed the algorithm performance in terms of accuracy. However, pruning is crucial in terms of run-time.

Algorithm 2 Airport Problem - Agent A_i

ProcessImage(Image *im*)
1 For each Path p in PDB
1.1 Let q be empty path
1.2 Let r=**Local-module**(p, *im*, q)
1.3 If r==’trigger alarm’ Then trigger alarm
1.4 If r==’continue’ Then forward q to A_{i+1}

L	N	LE	HE	$\overline{FRR}$	$\overline{FAR}$
4	500	1	2	0.0700	0.0011
4	500	2	0	0.0100	0.0682
5	500	2	3	0.0700	0.0022
6	500	2	3	0.1900	0.0000
4	1000	1	1	0.0967	0.0012
5	1000	2	3	0.0588	0.0040
6	1000	2	2	0.2225	0.0000
4	1500	1	0	0.1500	0.0001
4	1500	1	2	0.0700	0.0011

Table 1: Simulation results for selected scenarios

The local-module classifies a given path. The agents create paths, process them using the local-module and send them to the relevant agents. This flow of information is made according to the problem specifications. In the following subsection, we describe implementation of the interaction-module for the airport problem.

We have a camera installed in L airports: $ap_1, ap_2, ...ap_L$. At airport L we would like to detain every person who visited all airports *in a specific order*: $ap_1 \rightarrow ap_2 \rightarrow ...ap_L$. This allows us to use a simple sequential flow of paths as follows.

Agent A_1 (located at ap_1) acquires images and sends them to A_2. The interior agent, A_k, $1 < k \leq L$ maintains a database of paths sent by A_{k-1} denoted PDB. Each path in PDB is of length $k-1$. When a passenger’s image *im* is acquired at airport k, each path in PDB is sent to the local-module along with *im* (See algorithm 2). Paths that have enough positive comparison results are sent to A_{k+1} or trigger an alarm (if the path was created by the last agent).

We are interested in individuals who traveled through the airports in a specific order. This allows the agent to restrict the paths’ flow from agent A_i to agent A_{i+1}. Moreover, the costly procedure **Phase2** of the local-module is activated only by the last agent, which is the only one to process paths of size L.

3. EXPERIMENTAL RESULTS

We would like to know how the problem size (number of people N and number of information sources L) affects the accuracy of the algorithm. We also want to know how these parameters affect the running time of the algorithm. We would like to verify that the algorithm runs in real-time.

In order to answer these questions we implemented a simulation of the airport problem. For statistical significance we averaged results from 100 trials for each parameter studied. For each trial, we created a database which contains the flight plan for each passenger. We assumed a distribution function where most of the passengers only visit one or two airports out of the possible L airports. We modeled this behavior as an exponential decay function with the fewest people visiting all L airports.

We then ran experiments on various scenarios to test the overall behavior of the system. Table 3 shows various problem instances and the global FRR and FAR achieved by the MLBP algorithm. The first two columns describe the number of airports and the number of pictures taken at each airport. Columns three and four are the thresholds chosen for the simulation. The last two columns show the actual FRR and FAR. The results are very encouraging. For example, the results in line 5 shows that out of 1000 passengers, only one passenger will need special attention and the system will point out a target suspect with a reliability of 93%. We also tried to use ca_2 in the first phase and ca_1 in the second phase of the local-module (with $L = 1000$). The result was $\overline{FAR} = 1$! The high FAR of ca_2 caused each passenger in ap_L to have at least one path with enough false positive classifications to trigger an alarm.

We also were successful in significantly reducing the time complexity factor of this problem. We focused on the last agent since its time complexity is the most significant. The last agent is the only agent who uses the costly second pruning phase. This agent has the most time intensive decision to make, deciding in real time which passenger to single out. Other agents have the luxury of having more time before a final decision has to be rendered. Even in the worse case we studied, when $N = 1000$ and $L = 5$, the final agent was able to render a judgment under 12.5 seconds using a Pentium IV 3.0 Ghz computer with 1 GB of memory.

It is important to note that increasing the length of the path (L) does not necessarily increase the time complexity. The overall time needed by the last agent to make a decision is affected by two factors: 1) LT: the first threshold determines the number of paths reaching the last agent. A low threshold means a large number of paths that reach the last agent. 2) L: Borderline paths go through the second phase. The algorithm uses $\binom{L}{2}$ tests (in the worst case) on the second phase. Therefore, the longer the path the more time needed by the last agent for each path.

4. CONCLUSIONS

In this paper we addressed the domain of large-scale biometric pattern-recognition. The combination of multiple locations and streams of input, inaccuracy of biometric tests and real-time constraints make LBP problems hard to solve, and ideal candidates for MAS implementation.

We have presented the MLBP - a multi-agent solution to handle the LBP problem. The solution uses multiple agents to incrementally build solution candidates. Each agent consists of a local-module and an interaction-module. The local-module prunes candidates which do not reach a certain threshold. Borderline cases go through a secondary process of classification. *A web version which contains additional domains and a full theoretical analysis is available at www.cs.biu.ac.il /meshulr1/MLBP.pdf*

Acknowledgements : This work was supported by MAFAT and ISF #8008. In memory of Matat Rosenfeld-Adler.

5. REFERENCES

[1] A. K. Jain, A. Ross, and S. Prabhakar. Intro. to biometric recognition. *Transactions on Circuits and Systems for Video Technology*, 14(1), 2004.

[2] P. Weinstein, H. V. Parunak, P. Chiusano, and S. Brueckner. Agents swarming in semantic spaces to corroborate hypotheses. In *Proc. of AAMAS04*, 2004.

Using Queuing Theory to Predict Organizational Metrics *

Bryan Horling†
Multi-Agent Systems Lab
University of Massachusetts,
Amherst, MA
bhorling@cs.umass.edu

Victor Lesser
Multi-Agent Systems Lab
University of Massachusetts,
Amherst, MA
lesser@cs.umass.edu

ABSTRACT

Most existing organizational design processes focus on either the qualitative or domain-independent features of candidate designs. This paper demonstrates the significance of domain-specific features through an examination of an organizationally-driven information retrieval network. The behavior of a distributed search process and the consequences of hierarchical control are described. A model capable of predicting these characteristics is created in the ODML framework using techniques from queuing theory. This model can be used to guide the search for an appropriate design.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent Agents; I.6 [**Simulation and Modeling**]: Model Development; H.5.3 [**Group and Organization Interfaces**]: Organizational Design

General Terms

Design

Keywords

Organizations, Organizational Modeling, Queuing Theory, Hierarchical Systems, Quantitative Modeling

1. INTRODUCTION

Much of the existing work on organization design in multi-agent systems has focused on the qualitative aspects of those designs, or on a predefined set of numeric but largely domain-independent characteristics. In our recent work [2] we have developed a new representation called ODML that incorporates arbitrary quantitative information into the organizational model. Such a model can be used to make detailed predictions of how candidate designs will perform in different circumstances, which can be used as part of a larger search process through the space of design alternatives. Although such models can be more difficult to construct, we believe the benefits this more refined view can provide warrant research.

In particular, we believe that the ability to make concrete, numeric predictions about a range of domain-specific organizational characteristics can improve the design process by capturing the complexities that exist in realistic systems. In contrast to solutions that use instrumented simulations to make similarly detailed predictions, such explicit models also offer rapid but precise design evaluation, facilitating search of the design space. In this paper we demonstrate this potential by describing how an organizationally-driven information retrieval system has been modeled in the ODML framework. The end product of the modeling process is a single, succinct construct that both describes the space of alternatives and predicts the performance of individual designs.

The information retrieval model presented here is inspired by work by Zhang et al. [3], which uses a hierarchical organization to structure an information retrieval system. This allows queries to quickly propagate to data sources, and results routed back to a single agent in the network. At the top level of each hierarchy is a mediator responsible for providing a description of the data available in the information sources present in the hierarchy below it. Information sources may be an individual database, or an aggregator that manages other sources. Multiple mediators may exist, which service user queries by first using the signatures of other mediators to discover data sources, then routing the query to the appropriate mediators, and finally collecting the resulting data.

This organizational design provides several advantages. The use of hierarchies introduces parallelism into query processing and distributes the communication load. Conversely, if the data sources are widely distributed it may require an extensive search process to obtain a good result. There also exists a tension between the width and height of hierarchical structures, as narrow structures can have high latency and wide structures create bottlenecks. Determining what particular design to use can be a difficult problem, owing to the complex and interdependent nature of the organization's characteristics, interactions with the environment, and specified system requirements. We show in this paper how an ODML model facilitates this selection process by capturing these interactions, and using them to make concrete predictions about runtime behavior.

2. MODELING RESPONSE TIME

Similar to models created with other organizational languages, this model includes notions of roles, a task environment and perfor-

*This material is based upon work supported in part by the National Science Foundation Engineering Research Centers Program under NSF Award No. EEC-0313747. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

†Currently employed at Google, Inc.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

mance constraints. However, other relevant phenomena that must be captured in this domain are significantly different, and drive the shape of the organization in different directions. These include statistically predicting the results of the source discovery process, determining how the information contents of a hierarchy affect its expected load and approximating the effects of increased signature uncertainty caused by summarization. Although each can be modeled as a particular, distinct characteristic of the system, in reality they interact in complex ways. The tensions that arise in the resulting model embody the tradeoffs and decisions that must be made when designing the organization. This paper will focus on these expected response time of the system; the reader can refer to [2] for a description other characteristics this model possesses.

The IR system's response time is the duration of time between a query and the final response, a characteristic that is clearly important from an evaluation standpoint. Several design features affect this value. For example, each communication event incurs some message transit latency. The query processing by the databases, and the aggregation performed by both the aggregator and mediator will take some variable amount of time. Because queries are reflected to all subordinates, the latter two entities must also wait for slowest of their information sources before they can themselves respond. Finally, because multiple queries can exist simultaneously in the network, additional delays at individual agents can be incurred when a query must wait for existing processing to complete.

The rate at which an individual mediator will receive user queries it its *arrival_rate*. Responses will be returned back at this same rate, on average. More concretely, the existing model assumes that queries have a Poisson arrival distribution and mean rate *arrival_rate*. After the query makes its way down through any aggregators, the leaf databases will receive the query, also at rate *arrival_rate*. Each database has a *service_rate*, defining how quickly it can process queries. At any given time there may be previously received queries already being processed or waiting at the database. We assume FIFO processing, so the amount of time any new query must wait will depend in part on these existing queries.

Queuing theory can be used to analyze how long the wait will be, by modeling a database as a *M/M/1* queue with Poisson task arrival rate and service rate. The pdf $f_M(x,\lambda)$ and cdf $F_M(x,\lambda)$ of the *M/M/1* queue's waiting time distribution are given below, where $x \geq 0$ and $\lambda = service\_rate - arrival\_rate$.

$$\begin{aligned} f_M(x,\lambda) &= \lambda e^{-\lambda x} \\ F_M(x,\lambda) &= 1 - e^{-\lambda x}, \end{aligned}$$

The mediator waits for responses from all its information sources before progressing, so its expected service time will be dependent on the maximum service time of those below it. This value can be found by computing the maximum order statistic (the expected maximum value) of those source service times. The model makes a simplifying assumption that the source service time distributions are independent, but does not assume they are identical. The model generates the pdf $f_{(n)}$ and cdf $F_{(n)}$ distributions of this inid max order statistic using the following functions[1]:

$$f_{(n)}(x) = \Big[\prod_{i=1}^{n} F_i(x)\Big] \sum_{i=1}^{n} \Big(\frac{f_i(x)}{F_i(x)}\Big) \quad (1)$$

$$F_{(n)}(x) = \prod_{i=1}^{n} F_i(x), \quad (2)$$

where f_i and F_i represent the pdf and cdf of the ith sample, respectively (i.e., the service time distribution of the ith source).

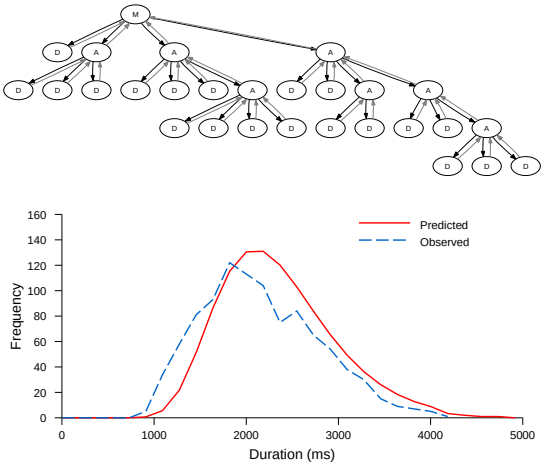


Figure 1: The predicted and observed response time distributions in organizations with twenty-eight agents. Node M is a mediator, A are aggregators, and D are databases.

The mediator itself is not simply a pass-through, but must process and aggregate the resulting data as well, introducing additional latency. Thus, the mediator can also be viewed and modeled as a *M/M/1* queue. The service rate of the mediator depends on the number of responses it receives, which depends on the number of information sources below it. The mediator's pdf and cdf can also be produced using Equations 1 and 1, with $service\_rate = response\_service\_rate/num\_sources$, and a Poisson rate of $\lambda = arrival\_rate - service\_rate$.

The model determines the total service time by combining these two activities, modeled as the sum of the times exhibited by these two random variables. The total service time pdf f_C and cdf F_C can then be determined by finding the convolution of the corresponding distribution functions, which has the general form:

$$f_C(x) = \sum_{i=0}^{d\_range/d\_step} f_s(i) f_l(x-i) d\_step \quad (3)$$

$$F_C(x) = \sum_{i=0}^{d\_range/d\_step} f_s(i) F_l(x-i) d\_step \quad (4)$$

For the mediator, f_s would be the aggregate information source pdf given in Equation 1, while f_l and F_l would be the pdf and cdf of the waiting time for the local *M/M/1* queuing process.

Note that Equations 1-4 are recursive, in that they rely upon both the pdf and cdf distributions of the sources below the mediator. The equations make no assumptions about the form of those distributions, so they can be used both when the information source is a single database or an arbitrarily complex aggregator hierarchy. This same assumption also allows Equations 1 and 2 to be used to compute the pdf and cdf distributions for the aggregator itself. The definition terminates in the exponential response distribution exhibited by the databases. By incorporating the cumulative overhead incurred by the message transit times of the query and result propagation process the model can now compute the expected *service_time* of the mediator, allowing prediction of the *response_time* distribution of the organization as a whole.

A representative prediction is shown in Figure 1. The scenario measures the response time performance of a different IR organi-

Organization	Query Rate 0.1	0.25	0.5	1	2	3	4	5	6	7
1. 	**725**	**723**	**719**	710	672	544	0	0	0	0
2. 	715	714	711	704	681	621	0	0	0	0
3. 	716	715	712	707	691	664	598	0	0	0
4. 	715	714	712	706	691	664	598	0	0	0
5. 	713	712	709	702	679	621	0	0	0	0
6. 	710	709	706	701	687	662	599	0	0	0
7. 	711	710	708	704	693	677	651	597	465	0
8. 	712	711	709	705	**695**	**680**	**657**	612	482	0
9. 	705	704	702	699	690	677	**657**	**619**	**517**	0
10. 	710	709	707	702	689	665	604	0	0	0
11. 	721	719	717	**711**	**695**	665	581	0	0	0
12. 	711	710	707	703	690	666	607	0	0	0
13. 	720	718	715	709	687	629	0	0	0	0
14. 	710	708	706	701	687	662	600	0	0	0
15. 	710	709	707	703	691	674	646	587	439	0
16. 	703	702	701	697	687	673	650	605	471	0
17. 	705	704	702	698	688	674	651	605	472	0
18. 	709	707	704	698	678	641	536	0	0	0

Figure 2: The utility predicted for the range of possible six-database organizations when the query rate is varied. Hollow circles are mediators and aggregators, with solid databases at the leaves. Optimal values for each rate are shown in bold.

zational design, by submitting 1000 queries to it in a Poisson fashion. The tested design is depicted above, along with the predicted and empirical response time data below. The solid line represents the distribution of response times predicted by the ODML model, while the dashed line indicates the observed frequency of response times in the simulation.

As can be seen in the performance graphs, the ODML model does a good job of predicting the response time distribution of the different organizational designs. Additional response time trials were performed for organizations with three [1M,2D], five [1M,4D], 10 [1M,2A,7D], 14 [1M,3A,10D], and 15 agents [1M,6A,8D], with similar results. The coefficient of determination R^2 was greater than 0.8 for all tested scenarios.

3. DETERMINING DESIGN UTILITY

There are many characteristics that impact design choice, including agent load, response quality and resource consumption. These are explicitly represented in the complete ODML model, and are used to derive two aggregate metrics, *response_time* and *response_recall*, which are combined to produce a single *utility* field that can then be used to compare and rank candidate designs during a search process. A complete description of how response recall is estimated in the model can be found in [2].

Figure 2 shows how utility is affected by the expected user query rate. This figure shows the full range of possible organizations that are possible in a six database environment with a maximum height of three and a minimum of two subordinates per node. The design parameters are such that response recall is identical among the alternatives, so utility is determined by the response time. When the query rate exceeds the service rate at one or more points, resulting in an infinite queue, the organization receives zero utility.

The single-level, single-mediator organization number 1 is predicted to be optimal when the query rate is 0.5 or less (i.e., less than one query every other second). This is intuitive, because the slow query rate avoids queuing delays, causing the response time to be dominated by the height of the organization. Note that other short hierarchies such as numbers 11 and 13 also have high utility.

As the query rate increases, first organization 11, then number 8 and finally number 9 become optimal, as the highly-connected mediator in organization 1 becomes a bottleneck. In contrast to most of the competing designs, organizations 11, 8 and 9 are all balanced (as is number 1). In an unbalanced organization, the segment with greater load tends to dominate the response time because the final result must wait for the slowest responder. These three designs avoid this by evenly spreading the load among participants.

The differences between the three revolve around where the final aggregation takes place, and how distributed the database search is. For example, 8 and 9 employ an additional mediator to aggregate the final results, while 11 does not. In 11 and 18, one of the mediators serves this role, which will slightly unbalance the load in the organization for any given query. Therefore, although the lack of another aggregating level saves on communication, the transient load imbalance ultimately leads to an increase in queuing delays.

These tests show one dimension of design tradeoffs that can be made in this environment. Organization 1 is fast but quite centralized. To avoid the increasing chance of bottlenecks, organizations 11, 8 and 9 trades off response time. Other organizations may trade off response quality, or other characteristics. Which design is most appropriate depends on the operational context. These needs may be incorporated into the model's definition of *utility* so that design alternatives are correctly ranked.

This range of viable alternatives, each possessing a set of interacting features, motivates the need for organizational models capable of representing the space of alternatives and predicting their quantitative character. More importantly, this work demonstrates how significant domain-specific characteristics can be affected by choices made in an organizational design. If such characteristics are not accounted for in the design process, or if they are approximated to the degree that important interactions are lost, then the ramifications of decisions that affect those characteristics may also be lost. We believe this will affect the quality of any resulting design, as those interactions are real and must be considered. This fact supports the detailed view of organizational behavior that is expressed in this paper, and the general ODML modeling framework we have designed to capture such details.

4. REFERENCES

[1] H. A. David. *Order Statistics, 2nd Ed.* Wiley, 1981.

[2] B. Horling. *Quantitative Organizational Modeling and Design for Multi-Agent Systems*. PhD thesis, University of Massachusetts at Amherst, February 2006.

[3] H. Zhang and V. Lesser. A Dynamically Formed Hierarchical Agent Organization for a Distributed Content Sharing System . In *Proceedings of the International Conference on Intelligent Agent Technology (IAT 2004)*, pages 169–175, Beijing, September 2004. IEEE Computer Society.

Agent Dependability as an Architectural Issue

Peter C. Lockemann
lockemann@ipd.uka.de

Jens Nimis
nimis@ipd.uka.de

Fakultät für Informatik, Universität Karlsruhe (TH), Postfach 6980, D-76128 Karlsruhe, GERMANY

ABSTRACT

The paper demonstrates how a layered reference architecture for software agents provides a framework for a dependability model that allows to associate the sources of failures and the ensuing error handling with a specific layer, thus integrating dependability directly into the design of agents.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Fault Tolerance; Reliability, Availability, and Serviceability; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems, intelligent agents*; D.2.11 [**Software Engineering**]: Software Architectures

General Terms

Design, Reliability

Keywords

Multiagent System, Layered Software Architecture, Dependability

1. INTRODUCTION

Ideally, agents never fail. Practically, though, disturbances may occur that are outside the control of the agent system. We call a multiagent system dependable if it or its constituent agents include internal mechanisms that guarantee that all services are rendered according to specification even in the presence of disturbances.

The main thesis of this short paper is that if the design of agents and multiagent systems follows the principle of levels of abstraction resulting in a layered architecture, the principle will also determine the design of dependability by associating a specific failure and its remedies with an individual layer. In this way one can develop a well-structured dependability model where each kind of error handling can be isolated and localized.

2. REFERENCE ARCHITECTURE

Let us first introduce a layered reference architecture as the framework within which to integrate dependability in a systematic fashion and thus prove our thesis. The architecture as shown in Fig. 1 was developed in [2] where it is explained in detail. The layers are based on the properties of intelligent software agents as listed by Wooldridge [7]. The lower two layers provide the basis for situatedness, autonomy, reactivity and proactivity, layer L3 for utility and goal-orientation. Layer L4 realizes agent behavior encompassing utility, reactivity and goal deliberation, while layer L5 adds sociability.

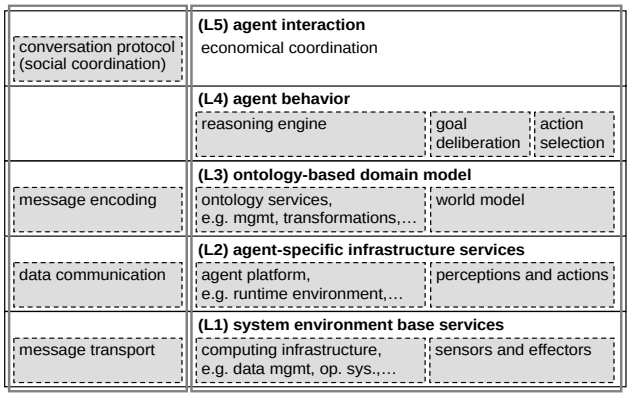


Figure 1: Reference architecture for a multi-agent system

Sociability includes agent interaction. Since a multiagent system is a distributed system where system control is distributed and agents are loosely coupled, information exchange above the physical layer must be incorporated within the individual agents. Modern message exchange software follows a layers pattern as well, referred to as a protocol stack. Fig. 1 includes the protocol stack. While interaction is primarily reflected on layer L5, layer L2 must account for the sequence of message exchanges between the agents, and layer L3 for the common understanding among the agents.

3. A DEPENDABILITY MODEL

3.1 Failures and dependability

We base our a dependability model on (some of) the definitions of Laprie [1]. Computer system **dependability** is the *quality of the delivered service* such that *reliance* can *justifiably* be placed on this service. The **service** delivered by a system is the system behavior *as it is perceived* by another special system(s) interacting with the considered system: its user(s). The service **specification** is an agreed description of the expected service. A system **failure** occurs when the delivered service deviates from the specified service.

The cause—in its phenomenological sense—of any failure is a (unintentional) **fault**. We should make sure that only very few faults occur in an agent. We speak of *fault avoidance* as the col-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

lection of all measures that reduce the number of faults. Upon occurrence, a fault creates a **latent error**. A latent error becomes an **effective error** when it is activated in the course of providing a service. Effective errors can be subject to treatment. If left untreated they are observed as failures.

3.2 Dependability model for a layer

A *dependability model* describes the service, the effective errors occurring within the system, which ones are dealt with and how (*error processing*), and what the ensuing failures are. Although we only consider faults endogenous to a multiagent system, from the perspective of the individual agent some may be due to exogenous failures, henceforth called external faults. The other faults are internal. An *internal* fault may be due to a programming fault or faulty parameter settings. *External* faults are either *infrastructure failures*, i.e., failures of the hardware and software the agent depends on to function (basically layers L1 and L2), or *peer failures*, i.e., agents on the same level of service that fail to deliver on a request due to connection loss, their own total failure, or inability to reach the desired objective due to unfavorable conditions (*unfavorable outcome*).

Error processing may at best mask the fault, i.e., the state reached after servicing the request is identical to the *regular state*. Next to best, the state is still the *old state*, i.e., the fault simply causes the request to be ignored. A bit worse, servicing the request may reach a state that while no longer meeting the specification may still allow the service requestor a suitable continuation of its work (*sustainable state*). At worst, the outcome may be a state that is plainly *incorrect* if not outright disastrous.

Our goal is to relate the dependability model to the layered reference architecture. Fig. 2 shows two layers where the lower layer provides a service requested by the upper layer. The figure summarizes which faults may occur, what kind of error processing may take care of them, which ones must be passed up to the next higher layer and which of these can receive special treatment there.

4. LAYERED DEPENDABILITY MODEL

4.1 Compressed model

Fig. 2 demonstrates that error processing cannot always be confined to a single layer. Instead it must often be spread to higher layers where a wider context is available, e.g., a sequence of statements rather than a single one, or a larger database. Consequently, a layered architecture should help to organize and localize the various dependability issues. We use the layered reference architecture of Fig. 1 for organizing the dependability of agents.

The result is shown in Fig. 3. It uses an abstraction of the dependability model of Fig. 2. A layer can be in one of two operation modes. During normal activity the component receives service requests from layers higher up, eventually accomplishes them using the services of the next lower layer components and returns normal responses. It may also draw on the services of other components on the same layer or receive service requests from such components.

If the lower layer fails to service the request, this can be interpreted on the current layer as the occurrence of a fault. The mode changes to abnormal activity where the fault is analyzed and processed along the model of Fig. 2. Propagated errors are signaled as a failure to the next higher layer. Otherwise the layer returns to normal mode and tries to resume normal operation.

Abnormal activity mode may also be entered due to abnormal events on the same layer. One such event is a malfunction within the same component—an internal fault. A typical mechanism for signaling such an event is (in Java terminology) by throwing exceptions. Exceptions can also be used to model a situation where a fault was not activated further down and thus remains unrecognized while crossing a layer in normal mode and moving upwards as a normal service response. Only on a layer higher up the fault is detected as such. Another kind of abnormal event is peer failures. If they are recognized as such they are being dealt with, otherwise they propagate upwards as part of a normal service response only to induce an exception higher up.

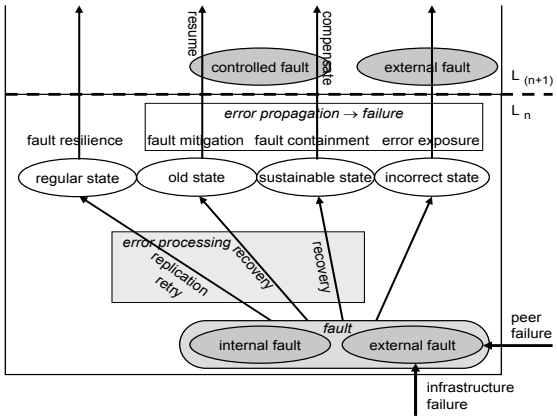


Figure 2: Dependability model: Faults, error processing and responsibilities

For the rest of this section we give a structured overview of the various faults that may occur on each layer, and short summaries of some of the approaches to cope with them (Fig. 3).

4.2 Layered model

All failures occurring on Layer L1 are infrastructure failures. The individual agent is affected by hardware, operating system, and data transmission failures. Operating systems have a built-in dependability so that only a fraction of faults such as excessive waiting or an unsuccessful disk operation are propagated upwards with a detailed description of the cause. Similarly, many of the faults typical for classical data communication are internally resolved by any of the numerous protocols so that only few failures are propagated upwards, such as network delays, connection loss, non-correctable transmission failures, unreachable destination. Some faults may cause a catastrophic failure, though. It cannot be recognized by the affected agent but will only be detected by collaborating agents ("node is down!").

Layer L2 covers specialized infrastructure services that are usually provided by agent frameworks or middleware. Typical interoperability services concern message handling and exchange, trading and naming, or agent mobility. For example, if an uncorrectable data transmission failure occurs on layer L1 resulting in a message loss, layer L2 may observe a time-out before the expected message has been received, and an exception method may invoke a retry by requesting a resend of the original message. A fault internal to layer L2 may be an invalid agent registration which is best be treated by notifying the upper layers of the failure to execute the needed agent service, where most probably the coordination layer (L5) would have to handle it by finding an alternative peer.

Layers L3 and L4 are tailored towards the domain semantics, with L3 dealing foremost with static aspects and layer L4 with the dynamics. Consequently, layer L3 mostly deals with peer failures in connection with ontologies, e.g., a mismatch in the meaning attributed to a certain term, or different coding into and decoding from the transport syntax.

Layer 4 deals with agent behavior and is in the best position to

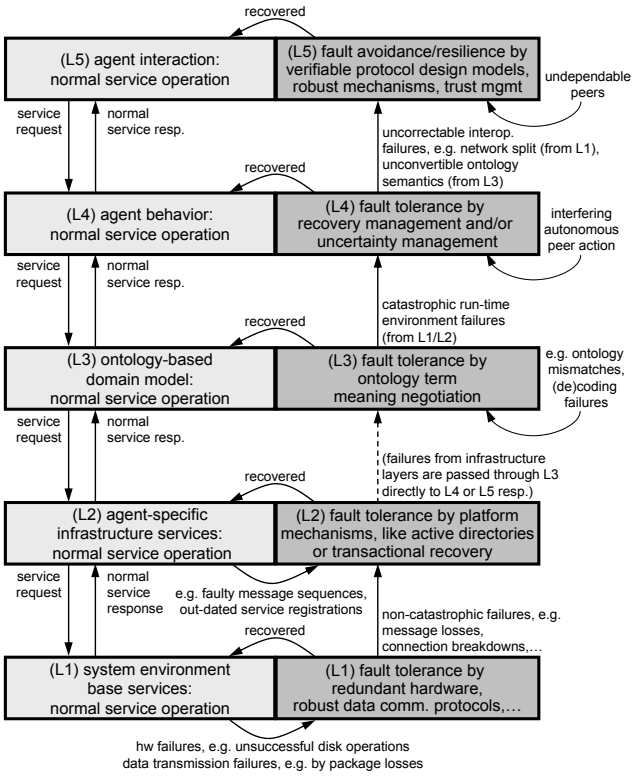


Figure 3: Dependability in the layered agent architecture

process the errors exposed from below. More precisely, L4 should be the layer of choice for all faults whose handling requires knowledge on the domain, the goals and the past interaction history. Consequently, most of the error processing will fall under the heading of fault resilience. This is true even for catastrophic infrastructure failures. Suppose the agent temporarily loses control when its platform goes down, but regains it when the platform is restarted. Recovery from catastrophic failure is usually by backward recovery on the platform layer. The resulting fault containment should now be interpreted in light of the agent's prior activities.

Layer L5 must ensure the dependable performance of the entire multiagent system. We refer to the coordinated interaction of agents as a conversation. Conversations follow a—usually predefined—conversation protocol. At runtime conversations are threatened by failures that are passed upwards from the lower layers. The infrastructure layers may pass up network splits and unreachable peers, the ontology layer semantic disagreements, the agent behavior layer unforeseen behavior of peer agents. Layer L5 may itself discover that peers respond in mutually incompatible ways. Techniques of fault resilience seem essential if the user is not to be burdened. For example, the agents may search for alternative services that are either very similar to the one originally needed, or they may attempt to agree on a new protocol with different peer roles or peers. A careful design should identify which errors should be left to the user and be passed as failures to what one could conceptually see as an additional "user" layer.

5. SPECIAL TECHNIQUES

Our group has extensively experimented with dependability issues in software agents. Below we give a few examples.

An abstraction of behavior that includes recovery is the transaction. The transactional abstraction of a conversation is the distributed transaction. Fault containment by recovery is then implemented via a 2-Phase-Commit (2PC) protocol, with a single coordinator as the initiator and any number of clients [6]. Hence, conversations spawn local transactions in the participating agents, that are coordinated by the initiating agent. (For technical details see [5]). The drawback is that 2PC robs the agents of some of their autonomy. Since 2PC ignores any transaction semantics, transactional conversations are a layer L2 technique.

Autonomy is preserved if only the individual agents are transactional but not the conversation as a whole. Hence, after recovery the agents are uncertain of the state of the conversation, they just carry beliefs. We modeled individual beliefs as fuzzy propositions over a discrete set of values and introduce a gradual consistency degree [3]. An agent can dynamically adapt its consistency degree over time. Fuzzified versions of belief revision operators allow to modify belief sets while observing a consistency prescribed degree. Fault mitigation by belief revision is a compensation technique for layers L3 and L4.

If one associates compensation with an entire conversation and, hence, an entire group of agents, the conversation must be understood to some detail. Consequently, such an approach is located on layers L4 and L5. Nested transactions for the agents fit the bill. They include diversified error processing such as local or global abort, backward recovery and retry, forward recovery and compensation. The synchronization that comes with a conversation is directly built into the nested transactions for the agents [4]. The drawback of this layer L4/L5 approach is the missing separation between agent behavior and conversation.

6. CONCLUSIONS

The paper demonstrates that a layered reference architecture for agents offers a framework for systematically allocating failure occurrences and the ensuing dependability mechanisms. Thus the architecture provides the means to include in the design of multiagent systems dependability right from the beginning. Our own work (funded by Deutsche Forschungsgemeinschaft) seems to suggest that the approach is indeed viable, but also indicates that dependability techniques for agents still raise intricate issues.

7. REFERENCES

[1] J. C. Laprie. Dependable computing and fault tolerance: concepts and terminology. In *15th IEEE Symposium on Fault Tolerant Computing Systems (FTCS-15)*, pages 2–11, 1985.

[2] P. C. Lockemann, J. Nimis, L. Braubach, A. Pokahr, and W. Lamersdorf. Architectural design. In S. Kirn, O. Herzog, P. C. Lockemann, and O. Spaniol, editors, *Multiagent Engineering Theory and Applications in Enterprises*, International Handbooks on Information Systems. Springer Verlag, 2006 (to appear).

[3] P. C. Lockemann and R. Witte. Agents and databases: Friends or foes? In *9th International Database Applications and Engineering Symposium*, pages 137–147, 2005.

[4] K. Nagi. *Transactional Agents: Towards a Robust Multi-Agent System*, volume 2249 of *LNCS*. Springer, 2001.

[5] J. Nimis and P. Lockemann. Robust multi-agent systems: The transactional conversation approach. In *1st International Workshop on Safety and Security in Multiagent Systems*, 2004.

[6] G. Weikum and G. Vossen. *Transactional information systems: theory, algorithms and the practice of concurrency control and recovery*. Morgan Kaufmann, 2002.

[7] M. J. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, 2002.

Quorum Sensing on Mobile Ad-Hoc Networks

Maxim Peysakhov, Christopher Dugan, Pragnesh Jay Modi, William Regli
Department of Computer Science
Drexel University
Philadelphia, PA 19104

ABSTRACT

Mobile ad-hoc networks (MANETs) are an increasingly important networking paradigm that will be the backbone of important defense and first response networks. Group decision-making is key to these environments, but is made difficult when MANETs are introduced due to network disruptions, bandwidth limitations, and host mobility patterns. Results gathered using standard group decision-making algorithms can become inaccurate, time-insensitive, or computationally undecidable. This paper focuses on a group decision-making approach using agent-based quorum sensing (ABQS) on MANETs. A mobile agent collects information (e.g., votes) from each host on a network until it can make an informed decision about global preference. This agent exploits the inherent tradeoff between efficient vote collection and result accuracy in order to provide better results, when considering survivability, hosts visited, hops made, and time spent, with only a very slight drop in correctness—benefits that greatly outweigh costs. Experimental evidence from live MANETs demonstrates the effectiveness of this solution.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed—*Intelligent agents*

General Terms

Reliability, performance

Keywords

Mobile agents, robustness and dependability of agent systems

1. INTRODUCTION

Mobile ad-hoc networks (MANETs) are composed of mobile computers that communicate over a wireless network without the support of an existing, fixed infrastructure. Instead, hosts can join or leave the network as they come in and out of range and each host acts as a router, allowing message traffic meant for another host to pass through it on a multi-hop route to its destination. MANETs present significant difficulties due to their high degree of decentralization, unstable physical-layer connectivity and resource constraint limitations. As a result, group decision making is a challenge that must be addressed. Security decisions protect the network from intruders, routing decisions affect the ability to communicate, and bandwidth decisions ensure network efficiency [2].

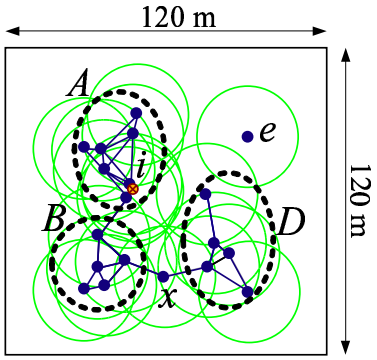


Figure 1: A simulated MANET topology.

An example group decision involving security on a MANET is highlighted in Figure 1 which depicts the *key revocation decision problem* [1]. Host *x* is believed to be compromised and no longer trustworthy so a decision must be made whether or not to revoke Host *x*'s key. This process is costly since it involves the creation and distribution of a new key plus it could possibly eliminate network connections between nodes (see cluster *D*); thus, the group should be certain that the host has really been compromised and that it can continue to operate functionally without Host *x* in the network. The obvious solution of voting is non-trivial due to the difficulties, described above, that MANETs present. Thus, elections can be incredibly inefficient or, at worst, undecidable.

In this paper, we propose an *agent-based quorum sensing* (ABQS) approach for vote collection in a MANET. In quorum sensing, vote collection is executed by a mobile agent or agents that traverses the hosts in the network. This approach promises to be more effective than conventional source/destination or broadcast message passing because the mobile agent is able to reason and adaptively choose its next destination based on certain percepts: connectivity, security, and distance to goal are examples. The agent uses a quorum sensing technique that employs a probability ratio calculation to compute a threshold. After the threshold is exceeded, the mobile agent decides that a reliable enough outcome has been determined. Furthermore, by exploiting the assumption that vote distributions are random, we can precisely calculate confidence intervals for any given threshold. This is important because it can allow principled tradeoffs between efficiency or cost of the vote collection procedure

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

and accuracy of the outcome decision.

2. AGENT-BASED QUORUM SENSING

There are two key components of an agent for quorum sensing:

1. An **agent migration procedure**, defining how the mobile agent navigates its environment to collect input (e.g., votes) from different hosts. In such a case, the environment is a MANET, hence agents need to sense and react to stimuli from the lower layers of the network stack (specifically the PHY, MAC and Link layers in the OSI Reference Model).
2. A **quorum evaluation procedure**, defining a statistical test to determine if the decision procedure can be terminated with high confidence of correct prediction of the outcome.

Agent Migration Procedure. Agent migration poses a unique problem on a MANETs. Most mobile agent approaches assume that communications are perfect or, at a minimum, reliable. In a MANET, migration of an agent from host *a* to host *b* may require traversal of multiple network hops that consume network resources (i.e., power, bandwidth, etc)—if it is even possible in the first place. Reliability is decreased as a consequence. Thus, this approach aims at reducing hop-count in order to increase reliability while trying to maintain a certain level of accuracy. Consider a network consisting of a set of hosts $H = \{h_1, h_2, \ldots h_{|H|}\}$. Without loss of generality, assume the quorum sensing agent starts off with a workflow that traverses these hosts in order (i.e., $h_1, h_2, \ldots h_{|H|}$). A greedy heuristic can be defined with a simple inductive argument. For the base case, assume agent a is at host h_1:

- Let $\bar{H}$ be the hosts traversed so far, hence $\bar{H} = \emptyset$ and agent a needs to visit the hosts in the set H/h_1; *succ*$(h_1) = H/h_1$.
- Let *hop-count*(h_1, h_i) be the number of network hops from host h_1 to host h_i. This measure can be sensed using sensors that can access the state information maintained by the network routing protocol in the MANET (i.e., OLSR or AODV).
- Select the next host h_i such that $\forall h_j \in H/\{h_1, h_i\}$, *hop-count*$(h_1, h_i) \leq$ *hop-count*(h_1, h_j); $\bar{H} = \bar{H} \cup \{h_i\}$.
- Migrate to h_i.

For the inductive case, assume agent a is at a host h_k:

- If the *quorum threshold* has been reached, terminate migration and return to the originating host with the result; else
  - *succ*$(h_k) = H/\bar{H}$;
  - Select the host h_i such that $\forall h_j \in H/\{h_k, h_i\}$, *hop-count*$(h_k, h_i) \leq$ *hop-count*(h_k, h_j); $\bar{H} = \bar{H} \cup \{h_i\}$.
  - Migrate to h_i.

Quorum Evaluation Procedure. In ABQS, we deploy a mobile agent to sequentially collect votes—not unlike Wald's classic sequential probability ratio test [7]. The ABQS vote collection agent contains a quorum sensor based on those found in social insects, specifically the *Leptothorax albipennis* family of ants. These ants estimate the quality of new nesting sites by observing a *quorum number*, a rough calculation of the number and suitability of nest mates [5]. Whenever this *quorum number* goes outside of an ants indecision threshold, the nesting site can be decided upon.

Our goal is to determine a threshold which results in a statistically reliable performance for the quorum sensing agent in a vote estimation task. We assume that the samples are independent. If this is not the case, we force the agent to collect votes in a random manner thereby removing the dependency. The probability density function (PDF) f is then computed for the fair binomial distribution for the corresponding number of tests as each vote is collected:

$$f(x = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Let n be the number of votes collected, k be the number of positive votes and $p = 0.5$. The value of the function f is the probability that a sample with exactly k positive votes was generated by a binomial distribution with mean of 0.5. The narrowest confidence interval C determined around the mean of 0.5 such that the combined probabilities of all its members $F(C)$ are at least 90%:

$$F(C) = \sum_{k \in C} f(x = k) \geq 0.9$$

Note that the 0.9 can be changed in order to take advantage of the tradeoff between accuracy and time mentioned earlier. Elections are too close to call and the process of collecting samples continues when the collected sample belongs to the interval C. If, however, the collected sample is outside of the interval C, then this would indicate a strong bias toward one of the two options and the election can be decided with certainty $\frac{1+F(C)}{2}$. More confidence in the predicted result can be achieved by broadening the interval C.

3. EMPIRICAL VALIDATION

We analyze the performance of ABQS briefly in simulation and focus on embodied experiments. Votes can be collected in two ways: sequential and parallel. Sequential vote collection sends out a single agent which tours the network and collects votes. ABQS has an obvious implementation in such a case because the agent determines when the election is decidable. Parallel vote collection sends one agent to each host and returns with the result. The Host collecting the votes implements ABQS as votes are received and stops collecting ballots once the threshold is surpassed.

Simulated Experiments. A MANET simulator [6] is employed to test these hypotheses for large numbers of hosts under a variety of networking scenarios. The simulator enables the creation and mobility of agents which can perform various tasks—voting, in this case. The transmission time for each agent over each link is inversely proportional to the square of the physical distance between the hosts if they are in radio range of each other. Links are random and continuously changing. Agents cannot travel between hosts that have not established direct radio contact.

The network simulator was initialized with 30 mobile hosts. Each host was given a random vote with $[60 : 40]$ bias towards the "yes" decision. Agents were given 500 iterations of the simulator to collect the votes and report their decision. The experiment was repeated 500 times during a single run of the simulator. We also performed 10 independent runs of the simulator to reduce the dependency on the random number generator used in the experiments. As a result, 5000 data points were collected for each agent. Out of these experiments, agent $\mathcal{A}$, a sequential collecting ABQS agent, was able to produce a result 83% of the times while agent $\mathcal{T}$, a sequential collecting agent which requires all votes, produced the result only in 77% of all cases. Considering only the experiments where election results were obtained or estimated, we computed measures for correctness and performance improvement. Table 1 gives the comparison of both agents based on their average performances. Agent $\mathcal{A}$ is 6% more survivable and about 19% more efficient (in terms of visited hosts, network hops and time spent) than traditional agent $\mathcal{T}$ with only 0.5% sacrifices in correctness.

Embodied Experiments. We present experiments using Drexel University's live MANET testbed[4, 3]. It consists of dozens of mobile computing hosts—PDAs and tablets—on an 802.11b wireless network with ad-hoc routing. In particular, five (5) Compaq TC-1000 tablet PCs were configured with the OLSR MANET routing protocol operating over CISCO Aironet 802.11b network cards. These hosts were placed in an actual outdoor urban environment.

Agent	No ABQS	ABQS
Correctness (%)	100.0	99.5
Survivability (%)	77.0	83.0
Hosts Visited (%)	100.0	81.1
Hops Made (hops)	131.1	107.5
Time Spent (sec)	407.6	335.3

Table 1: Performance comparison for simulated experiments.

Method	Sequential		Parallel	
	No ABQS	ABQS	No ABQS	ABQS
Correctness (%)	100.0	100.0	100.0	100.0
Survivability (%)	90.0	**100.0**	93.0	**100.0**
Hops Made (hops)	29.4	**12.6**	61.2	22.5
Hops Made *(st.dev.)*	*3.7*	*4.4*	*5.7*	*11.7*
Time Spent (sec)	14.4	7.6	5.9	**3.5**
Time Spent *(st.dev.)*	10.3	7.5	5.7	1.6

Table 2: Performance comparison for embodied experiments.

Note that this is an uncontrolled environment that includes multi-path interference, other networks as well as disruptions from foliage, automobiles and physical structures.

Hosts 1 through 3 are static, positioned linearly and approximately 40 meters apart. Hosts 4 and 5 are mobile and randomly change position. Each physical device runs five virtual hosts. A slight modification to the routing protocol makes hosts on the same physical device appear two hops away. This forces agents to treat virtual hosts just like they would physical hosts—simulating a larger network with fewer physical nodes. Host 4 begins all election procedures and we compare sequential and parallel vote collection with and without ABQS.

The votes of individual group members were pre-determined and randomly distributed prior to the vote collection in such a way that there was an exact 74% bias in each election towards a "yes" vote. Rogue hosts are uncommon in practice so a 74% bias conservatively approximates real-life situations. The threshold for ABQS was set to 95% accuracy in all of the performed experiments. All agents are compared on completion time, bandwidth utilization, whether or not it completed, and accuracy. The experiments were repeated 30 times. Average values and standard deviations of measured parameters are presented in Table 2.

In each of the experiments, all of the elections were tabulated correctly. The size of our network (25 hosts) and the number of experiments did not appear to be large enough to allow any mistakes in ABQS to manifest themselves. Both parallel and sequential vote collectors utilizing ABQS were always able to call the elections before the deadline. Vote collection without ABQS had a failure rate of 10% and 7% for sequential and parallel vote collection.

As expected, a sequential process of vote collection results in less network traffic and longer times than a parallel process. However, applying ABQS allows 57% less traffic for sequential vote collection. We recognize that the number of hops traversed prior to the making a decision for parallel ABQS does not represent accurately the savings in the bandwidth usage, since reaching a decision does not automatically remove all the vote collection agents that are still in transit. Although time costs have much higher standard deviations, we still saw a speed up of 47% for sequential and by 40% for parallel—levels that are statistically significant.

The numbers from comparison Table 2 and the histograms in Figure 2 show that both sequential and parallel methods of vote collection can gain in reliability as well as significantly increase their performance by utilizing the ABQS technique.

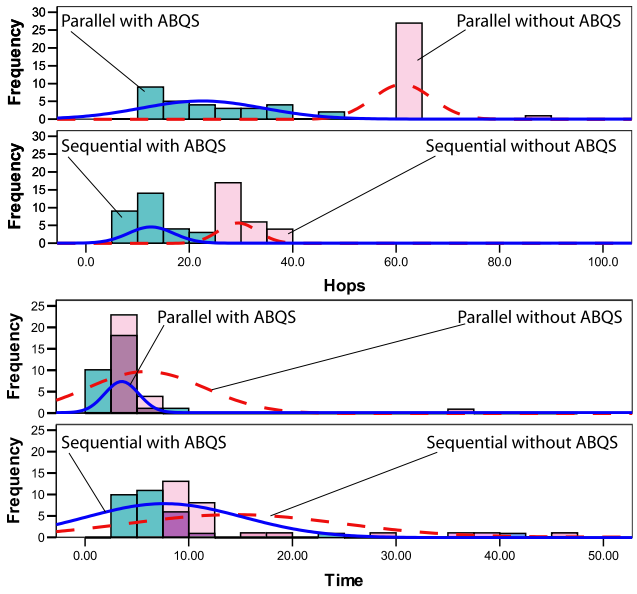


Figure 2: Histogram comparison of the number of hops and the time elapsed for all vote collection types.

4. CONCLUSIONS AND FUTURE WORK

This paper introduced a technique for *agent based quorum sensing* and provided a viable solution to distributed decision-making on a MANET. Greater efficiency was obtained in terms of significant time and bandwidth improvements at the cost of only a slight drop in decision accuracy. These results are not limited to a binary plurality vote. Preliminary results show that these techniques can be applied to other methods of vote collection: Borda count, approval vote, and plurality with multiple outcomes. The authors hope to explore that matter further, along with analyzing the exact role that vote collection order plays in efficiency.

5. REFERENCES

[1] D. Artz, M. Peysakhov, and W. Regli. Network meta-reasoning for information assurance in mobile agent systems. In Proc. *IJCAI*, pp. 1455–57, 2003.

[2] F. Bai, N. Sadagopan, and A. Helmy. Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks. In Proc. *IEEE Comp. and Communications Soc.*, pp. 825–835, 2003.

[3] V. Cicirello, *et. al.* Designing dependable agent systems for mobile wireless networks. *IEEE Intell. Sys.*, 19:39–45, 2004.

[4] J. Kopena, *et. al.* Service-based computing on manets: Enabling dynamic interoperability of first responders. *IEEE Intelligent Sys.*, 20(5):17–25, 2005.

[5] S. Pratt, E. Mallon, D. Sumpter, and N. Franks. Quorum sensing, recruitment, and collective decision-making during colony emigration by the ant leptothorax albipennis. *Behavioral Ecology and Sociobiology*, 52:117–127, 2002.

[6] E. Sultanik, M. Peysakhov, and W. Regli. Multi-agent-based simultation iv. In Proc. *MABS*, 2005.

[7] A. Wald. *Sequential Analysis*. John Wiley & Son, 1947.

Performance Evaluation of Open-Source Multiagent Platforms*

Luis Mulet
lmulet@dsic.upv.es

Jose M. Such
jsuch@dsic.upv.es

Juan M. Alberola
jalberola@dsic.upv.es

Department of Information Systems and Computation
Polytechnic University of Valencia
Cami de Vera s/n. 46022, Valencia (Spain)

ABSTRACT

Nowadays, most multiagent platforms are internally designed as middleware and are usually implemented in Java and run on top of an operating system. This kind of design maximizes portability and reduces the development cost; however, it may lead to low performance and scalability. In this context, our research has the long-term goal of integrating into the operating system some key services which are currently supported by middleware platforms. The first step in achieving this goal is to study some well-known, open-source platforms in order to understand to what extent the internal design of a platform influences its performance.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Measurement, Performance, Design, Experimentation.

Keywords

performance evaluation of agent systems, agent and multiagent architectures

1. INTRODUCTION

The area of Multiagent systems has produced both methodologies and actual frameworks to make the implementation of agent-based systems possible. The support software of such frameworks are often known as multiagent *platforms*. In essence, these platforms provide both a model for developing multiagent systems and an environment for running distributed agent-based applications. Among all the existent platforms, only those whose source code was available were feasible for us, since part of the study required the inspection of specific parts of the code. Three of these platforms were selected: JADE[3], MadKit[4] and AgentScape[1]. They are representative of different design principles and different programming models (from FIPA[2] compliance to multi-language support).

*This work has been partially funded by grant TIN2005-03395 from the Spanish Government

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. EXPERIMENT DESIGN

Services tested in this paper were either common to the three platforms or offered the same function. Moreover, these services are defined by the FIPA standard. The messaging service is the most important service due to the fact that multiagent systems are based on the communication between the agents. The service directory service is also important because an agent platform requires services offered by agents be accessible to other agents.

2.1 Messaging Service

We measured the RTT (round-trip time) of each sent message between a sender agent and a receiver agent, i.e. the time elapsed between when the message is sent to the receiver and when it comes back to the sender. The number of couples was increased in order to observe the platforms scalability. For each agent couple, we calculated an average RTT from a collection of 1000 sent messages. The result of the experiment was the average of the RTT averages from all the couples in the experiment. The experiments were also repeated changing the kernel (container in JADE terminology) and the host where the agents are placed.

The effect of message size on messaging service performance was also tested. For this purpose, we performed an experiment modifying the message size (from 1 byte up to 100000 bytes). The number of agent couples were fixed to 50 and 100.

Finally, in order to observe the platform behaviour when a massive message sending occurs, we designed a test in which many agents tried to communicate with an unique receiver. The number of sender agents was increased from 1 to 250.

2.2 Service Directory Service

We tested the service directory service response with two kinds of experiments. First, registration and deregistration time were measured by increasing the number of agent services already registered. The structures of the platforms were changed from one host to two hosts in order to determinate the influence of agent location. We also estimated the

platform response when a service search was requested. As explained above, the number of agent services already registered was increased. The agent location was also changed. We allocated the searcher and registered agents in both the same or different hosts.

3. EXPERIMENT RESULTS

In order to perform the experiments, we used two PC's Intel Pentium 4 @1.5 GHz, 256 MB of RAM memory and Sun JDK 1.5 version.

3.1 Messaging Service

Figure 1 shows the behaviours of the three platforms when the sender and receiver are placed in the same kernel located in one host. AgentScape offers the poorest RTT values. These results may be due to the fact that there is yet no mechanism to make a message blocking reception in this platform. AgentScape proposes a polling mechanism for message reception. According to this, the platform suspends the agent between each iteration. JADE and MadKit have a similar response in low load. In contrast, when load increases, JADE RTT grows with respect to MadKit. This may be due to the fact that MadKit uses a direct copy-write mechanism, while JADE uses a more complex process which is also utilized in distributed communication.

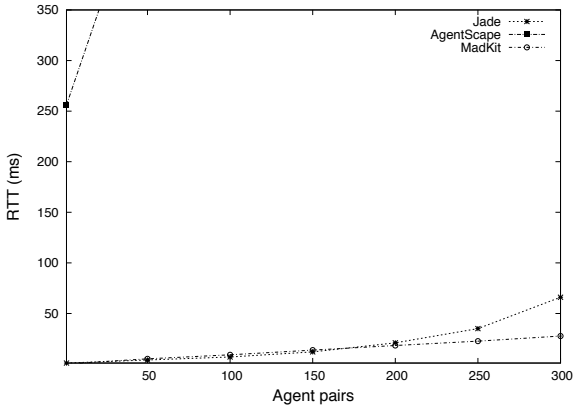


Figure 1: Message Sending, 1 host, 1 Kernel

When the agents are located in different kernels (Figure 2), JADE presents shorter RTT delays in the message sending process than the other two platforms. This is due to how containers are implemented and the fact that they communicate among them using Java-RMI. Madkit distributed messaging introduces an overload due to the set of *Communicator Agents* that are responsible for communicating different kernels through sockets. As a result, several local messages are needed to carry out a distributed communication. AgentScape uses XML-RPC technology to communicate different kernels and requires a thread for each kernel.

We can see the message size experiment in Figure 3, in this case AgentScape indicates the worst RTT values, being less adaptive to size changes than JADE and MadKit. Besides, JADE presents the best response to a multiple sending. It manages the message queues more efficiently than the others when a lot of messages are received. In contrast, AgentScape and Madkit are more influenced by this massive reception.

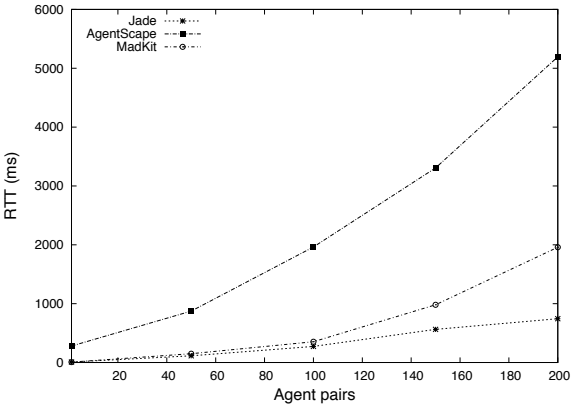


Figure 2: Message Sending, 2 hosts

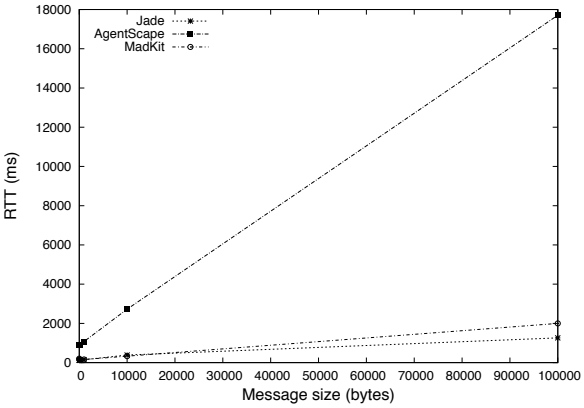


Figure 3: Message Sending 50 pairs

JADE is more scalable than the other platforms. Due to the lack of space, figures showing results for multiple sending are not included.

3.2 Service Directory Service

Figure 4 depict the registration time in 1 host. In these experiments, the differences are reduced. The number of services already registered does not have any influence on a new agent service register time. The MadKit registering service is the fastest one because it is distributed and duplicated among all the kernels. When an agent wants to register a service, it is done automatically by the local kernel. Afterwards, if there is more than one kernel, information replication is needed in the other kernels. AgentScape and JADE do not have this service distributed. In AgentScape, when an agent wants to register a new service, it must contact the *lookupService* of the platform. This platform has a better register time when the agents are in a host that is different from the lookupService. This is due to the fact that lookupService can be placed in a host without any other components of the platform. Hence, there is less overload in this host. JADE offers worse times than MadKit because agents that want to register their services must establish a communication act with the DF agent. This fact increases the time due to ACL message exchanges.

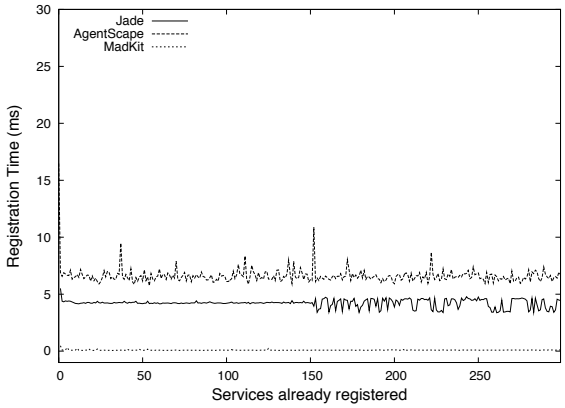


Figure 4: Registration in 1 host

As shown in Figure 5, AgentScape presents a linear increase in search time. This may be due to the fact that a method has not yet been developed in the lifecycle of the AgentScape platform to suspend the agent execution. For this reason, there are more active threads, so the platform load is higher. In contrast, MadKit and Jade offer a constant behaviour. Since JADE implements service directory service processes using communication acts with the DF agent, its result times are not as good as those in MadKit. This is a result of how MadKit implements this service. While JADE implements it as an agent (DF agent), MadKit integrates it in the platform software. In other words, this service is centralized in JADE, whereas it is distributed among the platform kernels in MadKit. Due to the lack of space, the figures of the other search experiments have been not shown, but the conclusion is similar.

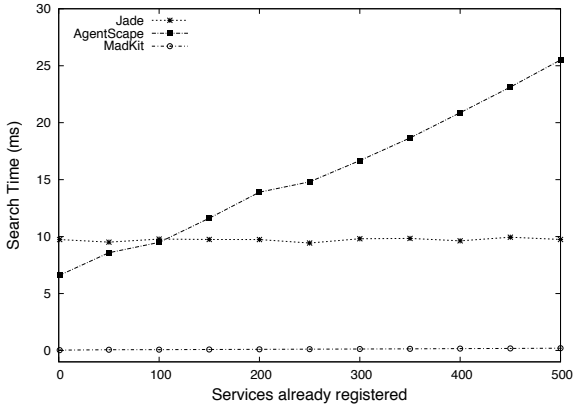


Figure 5: Search in different hosts

4. CONCLUSIONS

The main focus of the experiments was to try to link performance with internal design, that is, to identify the key design decisions that lead to better performance. In addition to the quantitative results about performance presented in the paper, two general conclusions can be drawn.

First, some platforms implement basic services (such as messaging) by means of agents, while others implement these services in their kernels. Offering services by means of agents may increase the platform modularity; however, it also degrades performance. Every time an agent needs the service, it has to communicate with the agent offering the service through an ACL act. This increases both the platform load and the service's response time. This conclusion is clearly proved in the two experiments shown in the paper. JADE implements the messaging service directly in the kernel, whereas MadKit requires an *AgentCommunicator* when agents in different hosts want to communicate. The difference in performance in this case is shown, for example, in Figure 2, in which JADE clearly outperforms MadKit. The service directory service in JADE is provided by means of an agent (the DF), while MadKit offers it directly. Again, Figure 4 shows how the platform that provides a service by means of an agent (like JADE) has a penalty in performance.

Second, centralizing services in a single host in the platform also degrades performance since this host can become a bottleneck in the case of very popular services. The experiments presented in the paper also support this conclusion. For example, Figure 5 shows that the performance for the service directory service in both JADE and AgentScape is much worse than in MadKit since, in these cases, the service is centralized. Therefore, distributing the services among the various hosts in the platform may benefit the response time of these services, however, the coordination among the hosts is an issue to be taken into account.

5. ADDITIONAL AUTHORS

Additional authors: Vicente Botti (`vbotti@dsic.upv.es`), Agustin Espinosa (`aespinos@dsic.upv.es`), Ana Garcia (`agarcia@dsic.upv.es`) and Andres Terrasa (`aterrasa@dsic.upv.es`).

6. REFERENCES

[1] Agentscape. `http://www.iids.org/research/aos/`.
[2] Fipa. `http://www.fipa.org`.
[3] Jade. `http://jade.tilab.com`.
[4] Madkit. `http://www.madkit.org`.
[5] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. Jade a white paper. *EXP*, 3:6–19, 2003.
[6] F.M.T. Brazier, D.G.A. Mobach, B.J. Overeinder, S. van Splunter, M. van Steen, and N.J.E. Wijngaards. Agentscape: Middleware, resource management, and services. In *Proceedings of the 3rd International SANE Conference*, pages 403–404, 2002.
[7] K. Burbeck, D. Garpe, and S. Nadjm-Tehrani. Scale-up and performance studies of three agent platforms. In *IPCCC 2004*.
[8] D. Camacho, R. Aler, C. Castro, and J. M. Molina. Performance evaluation of zeus, jade, and skeletonagent frameworks. In *Systems, Man and Cybernetics, 2002 IEEE International Conference on*.
[9] E. Cortese, F.Quarta, and G. Vitaglione. Scalability and performance of jade message transport system. *EXP*, 3:52–65, 2003.
[10] O. Gutknecht and J. Ferber. The madkit agent platform architecture. *Lecture Notes In Computer Science*, 1887:48–55, 2000.
[11] P. Vrba. Java-based agent platform evaluation. In *Proceedings of the HoloMAS 2003*, pages 47–58, 2003.

Structural Evaluation of Agent Organizations

Davide Grossi Frank Dignum
Virginia Dignum Mehdi Dastani
Utrecht University
PO Box 80.089, 3508TB
Utrecht, The Netherlands
{davide,dignum,virginia,mehdi}@cs.uu.nl

Lambèr Royakkers
Eindhoven University of Technology
PO Box 513, 5600 MB
Eindhoven, The Netherlands
l.m.m.royakkers@tm.tue.nl

ABSTRACT

A multi-agent system can be analyzed and specified as an organization consisting of roles and their relations. The performance of an organization depends on many factors among which the topology of its organizational structure, i.e., the set of relations holding between its roles. This short paper provides the sketch of quantitative methods for addressing the issue of the analysis, evaluation, and comparison of organizational structures. To this aim, quantitative concepts from graph theory are applied which deliver numerical analyzes of organizational structures. Here, an illustrative number of these concepts is sketched and their connection to properties that are commonly indicated as critical for organizations (robustness, flexibility and efficiency) is shown.

Categories and Subject Descriptors

F.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems—*Multiagent Organizations*; F.2.11 [**Distributed Artificial Intelligence**]: Coherence and Coordination

General Terms

Theory.

Keywords

Social structure, multi-agent organizations, graph theory.

1. INTRODUCTION

The notion of organization plays an important role in multi-agent systems, which is also reflected in many agent-oriented software methodologies (cf. GAIA, TROPOS). The performance of different organizations depends on organizations' characteristics such as robustness, flexibility, and efficiency. For example, hierarchies are known not to perform well in rapidly changing environments because of their poor flexibility. The paper is based on the intuition that a connection can be drawn between some of these characteristics and graph-theoretical properties of the structure of organizations. For example, flexibility depends on how strongly the roles in the organization are connected with one another. We claim that an investigation of this connection is important for the development of appropriate methods for comparing and evaluating different types of organizations and their performances.

We believe that, in order to tackle the evaluation problem, "the space of organizational options must be mapped, and their relative benefits and costs understood" [5], and to provide such a "map" a rigorous analysis of organizational structure plays a crucial role. The perspective chosen consists thus in addressing the evaluation issue from a structural perspective, that is to say, analyzing the organizational structure of MAS and providing a way to rigorously describe the pros and cons of them which lie in their structures.

More concretely, it is known for instance that *flexibility* is an advantage of network types of organizations and a disadvantage of hierarchical ones in rapidly changing environments. The notion of flexibility, though complex and multi-faceted, can be definitely given an interpretation on the basis of merely structural aspects. Intuitively, the more abundant are the connections between the roles in the organization, the more flexible is the organization. The point is to relate the notion of flexibility to precise properties of the organizational structure. Given an organization, can it be said to be flexible? And to what degree? Is it more flexible than another one as far as structure is concerned? How can a designer foster flexibility in a MAS just working on its structure? These types of questions constitute, in a nutshell, the target of the present work.

2. ORGANIZATIONAL STRUCTURE

In [3] a view on organizational structure has been proposed, inspired by foundational work on the theory of organizations (see for instance [7]), which is based on the claim that organizations do not exhibit only one structural dimension, but rather a multiplicity of interrelated dimensions, the dimensions of *power*, *coordination* and *control*.

According to this view, an organizational structure consists actually of (at least) three different structures. A natural way of modeling this notion of organizational structure is via directed graphs, which we represent here as systems of relations.

DEFINITION 1. **(Organizational structure)**
An organizational structure OS is a tuple:

$$\langle Roles, R_{Pow}, R_{Coord}, R_{Contr} \rangle$$

where $Roles$ is the finite set of roles, and $R_{Pow}, R_{Coord}, R_{Contr}$ are three irreflexive binary relations on $Roles$ characterizing the Power, respectively, the Coordination and the Control structures.

For every R_k s.t. $k \in \{Pow, Coord, Contr\}$, we denote with $Roles_k$ the subset of $Roles$ such that, if $(x, y) \in R_k$ then $x, y \in$

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

$Roles_k$. In other words, sets $Roles_k$ denote the set of roles involved in the structural dimension k. Each digraph $\langle Roles_k, R_k\rangle$ in OS will be also referred to as the *structural dimension* k of OS.

Besides the representation of organizational structure as systems of three different relations, the aforementioned [3] proposes also a formal analysis of the effects that each of these relations has on the activities of the agents playing roles in the organization. Briefly, the *power structure* defines the task delegation patterns possible within the organization. The *coordination structure* concerns the flow of knowledge within the organization, and the *control structure* has finally to do with the task recovery functions of the organization. In other words, the existence of a power link between role a and role b implies that every delegation of tasks from agent a (agent enacting role a) to agent b (agent enacting role b) ends up in the creation of an obligation directed to agent b. If a and b are connected via a coordination link, then every information act from a to b ends up in creating the corresponding knowledge in agent b. Finally, a control link between a and b implies that agent a has to monitor the activities of agent b, possibly taking over the tasks of agent b which have not been accomplished.

3. MEASURING STRUCTURE

This section presents three examples of equations measuring specific graph-theoretical aspects of organizational structures: connectedness, economy and univocity.

Connectedness of an OS concerns how strongly roles are linked with one another within one of the structural dimensions k, or in other words, how much is the given structure split in fragments.

$$\text{Connect}_k(OS) \;=\; 1 - \frac{|\text{DISCON}_k|}{|Roles_k| * (|Roles_k| - 1)} \qquad (1)$$

where $|R_k| > 0$ and DISCON_k is the set of ordered pairs (x, y) of $Roles_k$ s.t. there is neither a R_k-semipath[1] from x to y nor from y to x, i.e., the set of disconnected ordered pairs of the structural dimension $\langle Roles_k, R_k\rangle$. The condition $|R_k| > 0$ states that the structural dimension k does indeed exist. If the structure does not exist it cannot be measured. In practice, formula 1 measures how much that dimension is 'not disconnected'. With respect to connectedness, an important notion is that of cutpoint or, in an organizational reading, *liason role* [4], i.e., a role whose removal decreases the connectedness of the structure.

The *economy* of a given OS expresses a kind of balance between keeping the structure connected and of minimizing the number of links:

$$\text{Economy}_k(OS) = 1 - \frac{|R_k| - (|Roles_k| - 1)}{|Roles_k| * (|Roles_k| - 1) - (|Roles_k| - 1)} \qquad (2)$$

where $|R_k| > 0$. The equation is based on the intuition according to which the most 'economical' digraph of n points consists of $n-1$ links, i.e., the minimum number of links which is still sufficient to keep the digraph connected. Indeed, the nominator of the fraction, consists of the number of links in the structural dimension k which are in excess or in defect w.r.t. the optimum of $n - 1$ links. The denominator denotes instead the absolute number of links in excess in k. If $|R_k| = n - 1$ then the value of $\text{Economy}_k(OS)$ is optimal, i.e., equal to 1. The equation measures, therefore, how much k is 'not expensive' in terms of links, given that "links are not without cost in a social systems" [6]. Notice that $\text{Economy}_k(OS) = 1$ does not imply $\text{Connect}_k(OS) = 1$, it does only imply that there are enough links in R_k for it to be possibly connected. If the existence of symmetric links in R_k is assumed, then $n - 1$ links are clearly not enough any more for guaranteeing connectedness. On the other hand, notice also that $\text{Economy}_k(OS)$ can assume a value greater than 1. That indicates a sort of '*over-efficiency*' of k. In this case, it is easy to see that, if $\text{Economy}_k(OS) > 1$ then $\text{Connect}_k(OS) < 1$. In other words, if the economy measure of OS is lower than the optimal value 1, then OS has more links than the ones necessary for OS to be connected. If the economy measure is instead higher than the optimal value 1, than there are in OS too few links for it to be connected.

The property of *univocity* expresses the tendency of an OS to display the absence of redundant links ending up in the same role. In other words, how many of those connections are such that no role has more than one incident link of the same structural dimension.

$$\text{Univocity}_k(OS) \;=\; \frac{|\text{IN}_k|}{|Roles_k|} \qquad (3)$$

where $|R_k| > 0$ and IN_k denotes the set of roles x in $Roles_k$ which either have indegree equal to 1 in k or which are a source[2] of k or of some subgraphs of k. Intuitively, equation 3 measures how much a dimension k is univocal or "non ambiguous". The most univocal structures are assumed to be either the ones in which every point, except the source, has one and only one incident link (like in trees), or the ones in which exactly all points have only one incident link (like in cycles). Univocity has to do with the level of conflict and redundancies of a given structure.

4. LINKING STRUCTURAL MEASURES TO ORGANIZATIONAL PROPERTIES

In this section we show how graph-theoretical properties of organizationals structures can be used as a ground for establishing to what extent an organization exhibits key properties such as robustness, flexibility and efficiency.

Robustness. "Robustness is simply a measure of how stable the yield is in the face of anticipated risks. [...] *Adding robustness thus adds complexity*" [9].

A key "system characteristic" which should be preserved by any organization is the continuity of the task delegation chain. Failure of an agent enacting a role in this structure halts the delegation process only if the role it enacts is a *liason role*, i.e. a role the removal of which would result in a disconnected structure. So, the more alternative paths exist in the power graph the more robust is the organization: in a nutshell, low value for Univocity_{Pow}[3]. Moreover, the coordination structure determines how well information can disseminate over the organization. For robustness it is important that information about failures can spread to the roles that can take responsible action. Also this structure can serve as a back up for a failure of the power structure. So, one can easily claim that the more connected (Connect_{Coord}) and less economical (Economy_{Coord}) the coordination structure is, the more robust the organization is.

To sum up, relevant structural measures for robustness are:

Economy_{Contr}	0	Univocity_{Pow}	0	Connect_{Coord}	1

The 1 and 0 symbol indicate the value which is considered to maximize robustness with respect to that measure. For instance, the

[1] A R_k-semipath (of length n) is a sequence $\langle x_1, ..., x_{n+1}\rangle$ of distinct elements of $Roles$ s.t. $\forall x_i\ 1 \leq i \leq n$, $(x_i, x_{i+1}) \in R_k$ or $(x_{i+1}, x_i) \in R_k$.

[2] A *source* in $Roles$ is an element s s.t. $\forall d \in Roles$ with $d \neq s$ there exists a R_k-path from s to d.

[3] See also [2] for similar investigations on this issue.

maximum enhancement of robustness obtainable via modification of the connectedness measure is yielded by value 1.

Flexibility. "Flexible organizations are a looser co-operative association than classic hierarchical organizations. [...] Flexible organizations are continually in flux and are able to adapt in a flexible way to changing circumstances" [8].

As suggested by the above quote, an articulated power structure hinders flexibility constraining the distribution of tasks to predisposed patterns. In contrast, network organizations and teams, where no power structure exists, are commonly indicated as the paradigmatic example of flexible organizations. This suggests, for enhancing flexibility at the structural level, a low degrees of Connect_{Pow}. In absence of a power structure a coordination structure becomes essential because the knowledge concerning which agent might be capable to handle the new task needs to flow within the whole organization. The more roles are connected in this structure the more likely the right agent can be found to perform a new task. Connectedness together with a low level of economy (Connect_{Coord} and Economy_{Coord}) are thus directly linked also to the enhancement of the flexibility of an organization.

To recapitulate, these are the relevant measures for flexibility:

Connect_{Pow}	0	Connect_{Coord}	1	Economy_{Coord}	0

Again, 1 and 0 indicate the measures' values which are considered to maximize flexibility.

Efficiency. According to [1], efficiency mostly refers to the amount of resources used by the organization to perform its tasks. All links in the different organizational structures incur a certain cost to be maintained. So, the less links there are, the more efficient is the organization. Notice, on the other hand, that the maximum efficiency would require a power structure with one unique source from which the delegation of subtasks propagates. So, a fragmented power structure is intuitively *over-efficient* since links would miss through which the delegation activity can take place. Because of this a hierarchical, tree-like, power structure covering all the roles seems to be the most efficient. In fact, the existence of a power structure means efficient distribution of tasks, and a tree is the most efficient structure to cover all roles. Such a structure exhibits value 1 for Economy_{Pow}. As to coordination and control, the economy measure (Economy_{Coord} and Economy_{Contr}) should also be required to be optimal in order to minimize the amount of links. These are thus the measures we consider to be related to efficiency:

Economy_{Pow}	1	Economy_{Contr}	1	Economy_{Coord}	1

Again, 1 and 0 indicate the measures' values which are considered to maximize efficiency.

Needless to say, more different types of measures, which we can not expose here for space reasons, would provide a deeper analysis of the structural properties we touched upon.

5. TUNING STRUCTURAL MEASURES TO ORGANIZATIONAL PROPERTIES

At this stage the obvious question is whether organizations can be designed which maximize the adherence to all three properties at the same time. From a structural point of view and as intuition suggests, it is easy to show that this is not possible. Consider, for instance, the coordination structure. In fact, efficiency increases when Economy_{Coord} approximates 1. Maximum robustness and flexibility both require Economy_{Coord} equal to 0, while maximum efficiency requires Economy_{Coord} equal to 1:

	Robust	Flexible	Efficient
$\text{Economy}_{Coord}(OS)$	0	0	1

Intuitively, both robustness and flexibility increase the number of structural links and thus the costs of the organizational overhead, while efficiency reduces these overhead costs. Similar problems exist, for instance, for the power structure. The robustness criterion requires as many redundancies as possible, and therefore low levels of univocity, while flexibility demands the structure to be as small as possible and therefore with very low degrees of completeness. A number of similar incompatibilities can be detected and mathematically investigated.

Since it is not possible to maximize the adherence to all properties at the same time, the point becomes then to find suitable compromise solutions. An extensive analysis of the interdependencies between the various structural measures could provide useful insights on this type of issues.

6. CONCLUSIONS

The work addressed the issue of the influence of organizational structures on the performance of organizations, aiming at providing a rigorous method for analyzing, comparing and evaluating different types of structures. We proceeded as follows. First, making use of graph theory, we provided a number of meaningful measures for quantifying the adherence of organizational structures to specific structural features. Second, these measures have been used to ground a numerical analysis of the key organizational properties of robustness, efficiency and flexibility. Third, it has been shown that such an analysis pose the ground for an exact investigation of the extent to which those properties can conflict with each other, providing interesting information for a more aware design of organizational structures.

7. REFERENCES

[1] A. Etzioni. *Modern Organizations*. Prentice Hall, 1964.

[2] M. F. Friedell. Organizations as semilattices. *American Sociological Review*, 32:46–54, 1967.

[3] D. Grossi, F. Dignum, M. Dastani, and L. Royakkers. Foundations of organizational structures in multiagent systems. In *Proceedings of AAMAS'05, Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 690–697. ACM Press, July 2005.

[4] F. Harary. Graph theoretic methods in the management sciences. *Managemenet Science*, 5(4):387–403, 1959.

[5] B. Horling and V. Lesser. A Survey of Multi-Agent Organizational Paradigms. Computer Science Technical Report 04-45, University of Massachusetts, May 2004.

[6] D. Krackhardt. Graph theoretical dimensions of informal organizations. In C. M. Carley and M. J. Prietula, editors, *Computational Organization Theory*, pages 89–110. Lawrence Erlbaum Associates, 1994.

[7] O. Morgenstern. Prolegomena to a theory of organizations. Manuscript, 1951.

[8] M. Schoemaker. Identity in flexible organizations: Experiences. In *Dutch Organizations Creativity and Innovation Management*, volume 12, December 2003.

[9] W. Stimson. *The Robust Organization: Transforming Your Company Using Adaptive Design*. Irwin Professional Publishing, 1996.

Comparing Market and Token-Based Coordination

Yang Xu
School of Info Sciences
University of Pittsburgh
Pittsburgh, PA 15260, USA
yxu@sis.pitt.edu

Paul Scerri and Katia Sycara
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{pscerri, katia}@cs.cmu.edu

Michael Lewis
School of Info Sciences
University of Pittsburgh
Pittsburgh, PA 15260, USA
ml@sis.pitt.edu

ABSTRACT

Many coordination algorithms claim to be *general*, implying that they can be used to coordinate agents in a variety of domains. However, little work has been done to quantitatively compare distinctly different approaches to coordination across a range of domains. In this paper, we present a detailed comparison of two published coordination algorithms, performed in an abstract coordination simulation environment that allows extensive, quantitative experimentation. The simulator is to used to compare two distinct approaches to coordination, token-based coordination and market based coordination. The results largely show the generality of different approaches, but performance and performance tradeoffs varies greatly across domains.

1. INTRODUCTION

Autonomous coordination is a complex process because several distributed algorithms are required to interact to produce agile, cohesive and efficient coordinated behavior. Because of the importance of autonomous coordination, many approaches have been developed, including approaches based on markets [1], tokens [5], etc. Typically, each of these approaches is designed to work in a specific domain, but the authors, usually with good reason, claim they will work in a wide range of domains. However, such claims are rarely quantitatively verified and, more importantly, competing approaches are rarely systematically compared. Thus, when a developer needs to select an approach to use in a particular domain, they are confronted with many claims but little concrete data with which to make a decision.

In this paper, we present an initial attempt at systematically and scientifically comparing distinct approaches to coordination. To make such a comparison feasible, we have developed an abstract simulation environment that is sufficient to capture a variety of real world concerns, but sufficiently abstract to be highly configurable and very fast. We used the coordination simulator to investigate the relative strengths of three distinct approaches to coordination: auction-based coordination [2]; token-based coordination [5]; and a hybrid of the two.

Based on an analysis of previous literature [1, 5], several hypotheses can be formed about the relative performance of the algorithms. Auctions are focused on maximizing overall utility taking into account the *bids* of all team members [1]. Token-algorithms are focused on scalability, hence they minimize communication, sometimes at the expense of overall utility. Thus, the clearest hypothesis is that auctions will communicate more than token algorithms, but result in better allocations of tasks and resources. More subtly, the performance advantage of an auction should be most pronounced when small changes in allocations lead to big differences in performance, i.e., typically highly constrained cases, while the token algorithms should maximize their communication advantage when the probabilistic models they rely on are most advantageous, i.e., weakly constrained cases. The empirical results support these hypotheses. Initial experiments suggested that auctions find superior allocations because they compare many options, while tokens use little communication by quickly focusing on the agents most likely able to perform tasks or having most use for resources. A hybrid algorithm that uses tokens to solicit auction bids from those agents most likely to submit winning bids then uses an auction to select from between the small number of bids should perform well. However, this hybrid algorithm should only perform well under restricted circumstances. We implemented the hybrid algorithm and compared its performance to the other algorithms.

2. COORDINATION ALGORITHMS

In this paper, we only consider the coordination tasks that can be performed by both tokens and auctions and is limited to task and resource allocation. The coordination problem in this paper is defined as: Agents, $A = \{a_1, \ldots, a_k\}$, are cooperating on a joint goal G. G is broken into discrete subtasks $\alpha_1, \ldots, \alpha_n$. Each subtask α_i when is applicable, must be performed by a single agent a to receive reward. The amount of rewards is defined as $Reward(Cap(a, \alpha), Holds(a)) \rightarrow \mathcal{R}$ where it depends on the capacity ($Cap(a, \alpha)$) that agent a can perform α and the exclusive resources that are available for a to perform α, i.e., a bucket of water is required for distinguishing a fire. The function $Assigned(a, \alpha) = 1$ if agent a is assigned to task α, otherwise it is equal to 0. Only one agent may be assigned a task at any time, i.e., $\sum_{a \in A} Assigned(a, \alpha) \leq 1$. The coordination is to maximize the reward to the team, while minimizing the *costs of coordination*. The overall reward is:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

$$\sum_{i=0}^{n} \sum_{a \in A} Assigned(a, \alpha_i) Reward(cap(a, \alpha_i), Holds(a))$$

We are specifically concerned with only the volume of Messages as costs of coordination. It is the number of times agents communicated, either between themselves or with the auctioneer.

The first of the algorithms we used for comparison is market-based approach. Our implementation of this approach was based on TraderBots [2] with adaptations where are necessary to make a comparison possible. In our market-based approach, one agent acts as auctioneer and both tasks and resources are treated as merchandise. Agents bid for either single items or combinatorial sets of items in order to maximize their own utilities. The auctioneer maximizes its utility by "selling" their "merchandise". In this approach, Sandholm's winner determination algorithm [3] is used to determine the allocation for tasks and resources by the auctioneer. Because of the centralized position of the auctioneer, it develops a complete knowledge of how agents will use a task or resource if allocated. Thus, the auctioneer can perform assignments that maximize the team utility. To be fair to all the bidders, the auction should last for a fixed period of time. Where early determination is infeasible; Agents are allowed to bid for resources after tasks have been allocated. Moreover, to prevent deadlock in resource allocation, agents are only allowed to bid for resources for their *first* pending task.

In Token-based algorithms [5], tokens encapsulate both resources and tasks, allowing actors to locally decide what to do with the token. For example, a *task token* allows an actor to decide whether to perform the task or pass it off for another actor. An intelligent routing algorithm is built in the token-based approach to help agents build local decision theoretic models to determine when and where to pass tokens. By utilizing the relevance between tokens, intelligent routing algorithm is able to efficiently deploy tokens to make higher utility with less communication [5]. In this paper, tasks are allocated by the LA-DCOP token algorithm [4], agent is allow to reject previous accepted task but accept another task that it can get more reward.

The hybrid algorithm works in the following way. The auctioneer algorithm runs exactly as before, except that instead of broadcasting announcements for auctions an *auction token* is created. Each auction token is allowed to exist from the starting of the auction to the end of the auction being closed. The auctioneer has a probabilistic model of the team state, just as all agents do in the token-based approach. The auction token is then intelligently routed to the agents most likely to be able to submit the best bids. The token stops moving after the auction it presents is closing or has visited a fixed number of teammates. Note that although the intelligent routing algorithm should work to route tokens for higher bids, it should not work better than the token-based approach because of incapability of making use of the relevance between tasks and resources which have been encapsulated into auction tokens. The auctioneer determines the winner of the auction and allocates tasks and resources the same as in the basic auction case. We expect that the hybrid approach should keep high rewards as auction-based approach but reduce communication over the basic auction, by targeting only those agents likely to make good bids and reduce computation by limiting the number of bids the auctioneer must deal with.

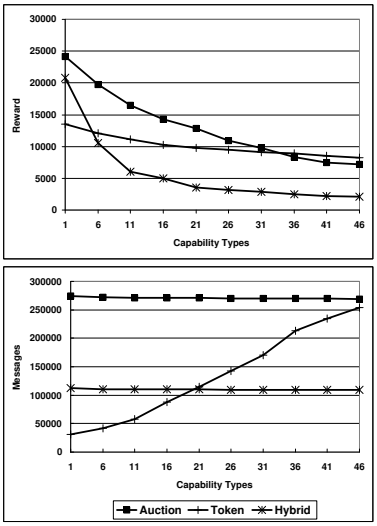


Figure 1: The reward for heterogeneous teams dramatically decrease in auction and hybrid approaches but maintains constant in token-based approach where more messages are required.

3. EXPERIMENTS

We implemented the three approaches in an abstract simulator called CoordSim. If not otherwise stated, the CoordSim are configured as follows. There are 100 agents to perform 50 tasks with 50 resources. Each task requires only one resources which could be interchangeable with four others. In the default setup, there is only one type of capability required and all agents have none-zero value for this capability, i.e., all agents are at least somewhat capable of all tasks. Auctions are held open for 40 time steps and the task tokens, resource tokens are allowed to move unless accepted. The initial threshold on a task token is 100, meaning that the task will not be accepted by an agents until it can get a reward more than 100 by performing this task. Simulation runs for 2000 steps and the results are based on 100 runs.

3.1 Heterogeneous Team

In the first experiment, we examined team performance by varying team composition and the capabilities required to perform tasks. For example, in an emergency response experiment some agents might only be able to fight fires while others could only provide medical treatment. In this experiment, we varied the number of capabilities from 1 to 46 where fewer agents in the team were available to perform particular tasks and in the most heterogeneous condition, only two agents on average are capable to performing a task.

The experimental results in Figure 1 show that for heterogeneous teams, auction and hybrid approaches earn less reward as the team becomes more heterogeneous because there are fewer agents able to compete for the more specialized tasks. The advantages of teamwide maximization of utility by the auctioneer decrease as there are progressively fewer feasible alternative bids. In contrast, reward

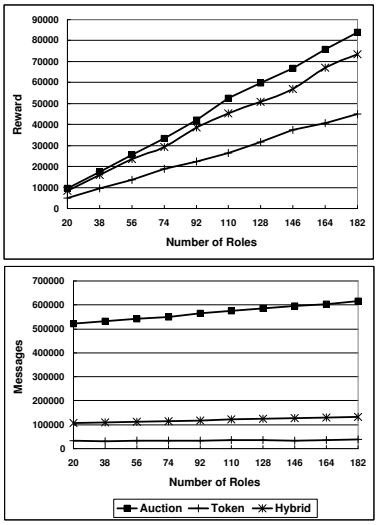


Figure 2: Reward and Messages increase dramatically with the number of tasks.

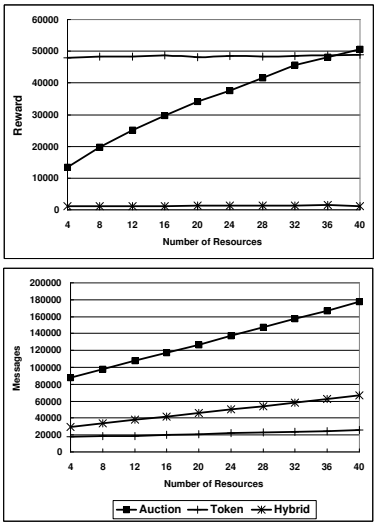


Figure 3: Reward decreases with scarcity in auction approach and is very low in hybrid approach, but with token-based approach is uniformly high

for the token-based approach remain almost flat with increasing specialization. We propose two reasons. One is that token-based approach greedily finds reasonable solution rather than searching for the optimal. As the other reason, by passing a higher number of tokens around the network and making use the relevance between them, intelligent routing algorithm gets better knowledge to route tokens. This is manifested that although the average distance to route a token increases with heterogeneity as reflected in an increase in message, token-based approach maintains the same level of reward.

3.2 Time Critical Tasks

In the second experiment, we investigated team performance when many tasks needed to be performed within a short period of time. To increase their reward, teams were required to perform tasks and allocate resources as rapidly as possible. In this study we varied the number of tasks the teams were required to finish from 20 to 182. After 2000 time steps, the accumulated reward and message count were recorded as shown in Figure 2.

All three approaches performed more tasks in order to get higher reward. As expected, the auction approach attained higher reward than the hybrid or token-based approaches. Considering both reward and messages, however, the hybrid approach performs well by almost matching the reward obtained by the auction at just a quarter of the communication cost. The reason whythe hybrid approach achieves such good performance with so little communication overhead is that the intelligent routing algorithm limits communication to a small number of agents while high bidders must always be informed in auctions.

3.3 Competitive Resources

The third experiment used 200 tasks each requiring an average of four resources with no interchange possibilities. As available resources are increased from 4 to 40, competition for them declines and they become less likely to be a bottleneck.

The experiment was stopped after 1000 time steps. Figure 3 shows that the reward for the auction based approach increased rapidly with increases in resources. Both the token-based and hybrid approaches remained flat with token-based approach earning the highest reward at all levels of scarcity while the hybrid approach yielded very limited reward. We hypothesized that because resource contention in this experiment was high the centralized control of the auction and hybrid approaches would often force agents to either bid for all four resources together or miss the task while the distributed token-based approach weakened this constraint.

Acknowledgements

This research has been sponsored in part by AFRL/MNK Grant F08630-03-1-0005 and AFOSR Grant F49620-01-1-0542.

4. REFERENCES

[1] B. Dias, R. Zlot, N. Kalra and A. Stentz. Market-based multirobot coordination: A survey and analysis. In *Technical report, CMU-RI-TR-05-13*, 2005.

[2] B. Dias and A. Stentz. Traderbots: A market-based approach for resource, role, and task allocation in multirobot coordination. In *Technical report, CMU-RI -TR-03-19*, 2003.

[3] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. In *Artificial Intelligence, 135*, 2002.

[4] P. Scerri, A. Farinelli, S. Okamoto and M. Tambe. Allocating tasks in extreme teams. In *Fourth Int. Conf. on Autonumous Agents and Multiagent Systems*, 2005.

[5] Y. Xu, P. Scerri, B. Yu, S. Okamoto, K. Sycara and M. Lewis. An integrated token-based algorithm for scalable coordination. In *Fourth Int. Conf. on Autonumous Agents and Multiagent Systems*, 2005.

SAgent: A Security Framework for JADE*

Vandana Gunupudi
Dept of Computer Science and Engineering
University of North Texas
gunupudi@cs.unt.edu

Stephen R. Tate
Dept of Computer Science and Engineering
University of North Texas
srt@cs.unt.edu

ABSTRACT

This paper presents SAgent, a general-purpose mobile agent security framework that is designed to protect the computations of mobile agent applications in potentially hostile environments. SAgent works with the JADE (Java Agent DEvelopment) platform [6], a FIPA-compliant multi-agent environment. SAgent supports modular and mostly orthogonal development of agent protection techniques and secure agent applications, so protocols and applications can be developed independently of each other. To accomplish this, a clean conceptual framework is presented which encapsulates in several general class interfaces the common security functionality required by secure agent applications. Furthermore, implementations are provided for two secure multi-agent protocols, and we give experimental results showing the feasibility of these protections. While a few other research projects have examined protocols and techniques for protecting agents, these have been theoretical explorations. SAgent's goal is to bring these theoretical techniques into practice so that they can be experimented with and used, in the framework of a design generic enough to support both software-based and hardware-based protections. The abstractions are clean, giving a well-de ned way for a new security provider to implement and experiment with new techniques for protecting mobile agents.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

Privacy, Safety and Security in agent systems; Agent-mediated electronic commerce and trading agents; Frameworks, infrastructures and environments for agent systems

1. INTRODUCTION

SAgent is a general-purpose agent security framework for the JADE platform that is designed to protect the computations of mobile agent applications in potentially hostile environments. The JADE (Java Agent DEvelopment) platform [6] is a popular mobile agent platform that enables the development and deployment of multi-agent applications using Java, and conforms to FIPA standards for software agents. SAgent supports the development of secure mobile agent applications, where data is protected from rogue or compromised hosts. While other researchers have developed frameworks that provide solutions for various agent security problems, such as authentication of agents, agent replication, and voting [3], SAgent is concerned speci cally with protecting the data that travels with a mobile agent. We believe that SAgent is unique in bringing into practice a software framework for protecting agents from malicious hosts with rigorous security requirements. Much of the previous work deals with securing communication or infrastructure, or providing agent authentication. Strong methods for protecting agent data have been proposed by researchers, including hardware-based techniques [9] and techniques based on clever use of cryptography and distributed protocols [2, 7], but SAgent is the rst attempt at unifying these techniques under a common framework, and as far as we're aware provides the rst experimental explorations of the proposed agent protections.

In SAgent, methods of protecting agent data are cleanly separated from applications using those protections, so both application and protection technique can be independently considered. A number of general security interfaces are de ned, which encapsulate the common security functionality required by secure agent applications. Security providers can provide implementations using these interfaces, and SAgent is distributed with two different implementations of software-based protection techniques, one by Algesheimer *et al.* [2] and one by Tate and Xu [7]. While the support for these software-based techniques is clear, the SAgent framework is general enough to support hardware-based protections just as cleanly. While the current lack of hardware to provide such protection makes this impossible to experiment with now, SAgent was designed so that if the current proposals for trusted platforms [8] become widespread and useful, then simple support in the SAgent framework would be possible.

SAgent was designed keeping in mind a basic tenet of software engineering, that de nition of proper abstractions is essential for software reusability. The key abstraction behind SAgent mirrors that of public/private key pairs in public key cryptography in which a private key is known only to the owner of the key, while the corresponding public key is available to everyone. Similarly, a mobile agent application has *public* functionality and information that it needs to perform computations at remote hosts and *private* information and functionality that is kept by the originator of the agent and not exposed to other entities. Note that while we refer to the information that travels as public, this does not mean it is under-

*This research is supported in part by NSF award 0208640.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00

standable to an outside viewer encrypted data (ciphertext) can be public even if the corresponding plaintext remains unintelligible. The private information is necessary only for the nal interpretation of the results and does not need to be available in any way during the agent's travels. To that end, our design separates the public and private functionality of the agent into distinct pieces.

There are two different views of these pieces in SAgent one from the point of view of a programmer that develops protection techniques for SAgent and another from the point of view of a developer who writes applications for SAgent. The architecture of SAgent is designed so that the security provider and the application-developer can remain unaware of each other and develop protocols and applications independently of one another. Data transfer and usage in SAgent revolves around a generic, intermediate data format that is not secure but also not application-dependent. The application developer is responsible for providing translation routines that convert application-speci c data into the intermediate format, and the security provider provides routines which convert this intermediate format into a secure but protocol-speci c representation. Data can then be operated on in this secure format by going through methods in the generic public interface, which resolve to the appropriate protocol-speci c methods. The contributions of this work include

- A clean and useful abstraction of data protection in mobile agents;
- A working platform that realizes these abstractions and provides an effective framework for both security providers and agent application developers; and
- Implementation of two software-only security solutions that are usable and practical for protecting small amounts of agent data.

As this is a short paper, many details are omitted from this presentation and the full version of this paper is available as a technical report [4]. The SAgent software is freely available under the LGPL license [1].

2. SECURITY MODEL

Agent applications are decomposed into protected and unprotected computations, with only the sensitive data and computation being run in the protected environment. For instance, a shopping agent may browse through a store using unprotected computations, but could use SAgent to protect the portions of the agent which make purchasing decisions. The protected portion could include very sensitive information, such as payment authorization values, credit card numbers, or electronic cash tokens. The protected agent computation is modeled as a 3-input, 2-output function, as shown in Figure 1. The three inputs are the current agent state, the input from the host, and an input provided by any unprotected computations performed by the agent on that host. The two outputs are an updated agent state and an output that is provided to the current host. Speci c security providers guarantee slightly different security properties, but in general the agent state is protected so that it is unintelligible to an outside observer (either an attacker or a malicious host), and the host's input is protected so that it can only be used in a manner consistent with the agent functionality, which can be determined or negotiated as a contract between the originator and the host.

3. CORE INTERFACES OF SAGENT

SAgent uses sound principles of object-oriented design and achieves abstraction by speci cation of several general *interfaces*, which are extended by security providers to supply protocol-speci c implementations. The design is similar to some well-designed cryptography libraries, such as Sun's Java Cryptography Architecture, in which generic encryption and decryption interfaces are de ned, and these are bound to speci c algorithms and implementations when the cryptography objects are allocated. The core of our design, therefore, includes secure private and public interfaces, along with a secure data interface, which we describe in this section.

3.1 SecureFnPrivate Interface

The `SecureFnPrivate` interface encapsulates the private functionality in our mobile agent model, de ning the methods for interacting with the agent that are restricted to the agent originator. The originator creates an agent for a particular application using a constructor given by a particular security provider, with the result being a `SecureFnPrivate` object. Through this interface, the originator sets the initial state of the agent, and retrieves the public part (a `SecureFnPublic` object) which is then sent out in the mobile agents for this application. Each mobile agent then visits various remote hosts, utilizing its public functionality to perform computations on behalf of the originator. Note that the agent can perform both unprotected and protected computations at remote hosts, and good design would suggest that only the sensitive parts of the agent application be performed within the SAgent framework. The `SecureFnPrivate` interface also de nes the functionality that allows the originator to decode the state of the agent after the agent returns home. After successful decryption of the agent state, the state of the agent is converted from a secure data format to a generic object, understood by the application. Thus, it can be seen that the `SecureFnPrivate` interface provides exactly those methods that allow the originator to perform operations that no remote host should be allowed to do.

3.2 SecureFnPublic Interface

The `SecureFnPublic` interface encapsulates the functionality required by the agent to perform protected computations at remote hosts. The originator initializes the agent with the public functionality it requires to perform computations at a host. Upon successful contact with the host (through the host agent), the agent supplies its public part to the host and asks the host for its input. Speci - cally, this step involves reading in the host's input in a generic form and converting it into an object of type `SecureFnData`, where the speci cs of this transformation depend on the particular protocol which implements this interface. Only the encoded input is returned from the host to the agent, which is what enforces that the host data is used only as speci ed by the particular agent functionality. Similarly, if the agent has unprotected computation results that need to be included in the protected computation, then it encodes its input using the *encodeAgentInput* method. Once the agent obtains the host and agent inputs in a secure format, it calls the *evaluate* method to perform the protected computation. This method updates the agent state and returns a `SecureFnData` object representing the output to be provided to this host, which can be converted to plaintext using the *decodeHostOutput* method. This evaluation obviously depends on the speci cs of the agent application, but the evaluation is protected in a uniform way by the particular security provider. The key to all of the mobile agent computations is that computations are only performed on `SecureFnData` objects, and the privacy and/or integrity of these operations is guaranteed by the security provider and the security protocol. Any protected information stored in the agent state can be operated on through the *evaluate* method, but can not be decoded or decrypted except

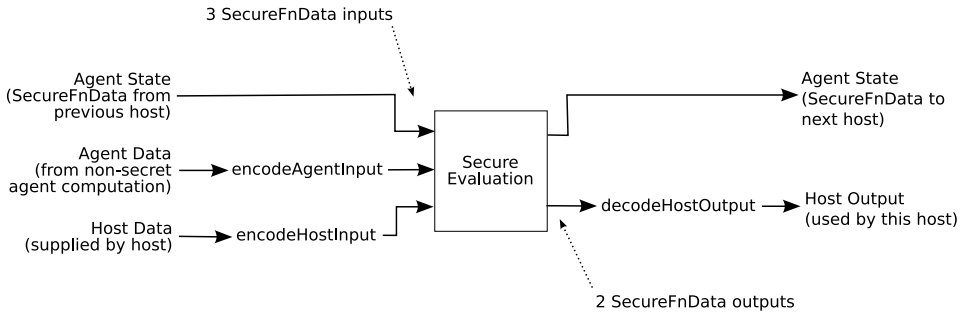


Figure 1: Agent State Update in SAgent

through the `SecureFnPrivate` object, which only the originator possesses.

3.3 SecureFnData Interface

SAgent is designed to protect the security of the mobile agent data. When data are operated upon within SAgent, the data are in a secure format. Therefore, the key idea with SAgent is that only *protected* data objects should be allowed within the protected execution. The `SecureFnData` type is simply a generic, opaque type to represent protected data, and does not provide any methods for operating on this data all operations are performed using either `SecureFnPublic` or `SecureFnPrivate` methods, including conversion to/from insecure data representations from/to `SecureFnData` objects. Different applications have different data requirements, so SAgent de nes an application-independent intermediate format that can be used within SAgent. This is just a binary representation of the data, and makes it simple for application-developers and security providers to develop their products independently of one another. The security provider just deals with bits which he encrypts, decrypts, and operates on as binary data, and is unaware of the actual form or meaning of the data in any application that may use his protocol. Once protected as a `SecureFnData` object, the actual encrypted representation of the data remains opaque to the user of the protocol. The application-developer, on the other hand, provides conversion routines between generic binary data and application-speci c data.

4. EXPERIMENTS AND EFFICIENCY

In this section we present experimental results from running the applications described earlier under both protocols provided with SAgent. Results of an extensive set of experiments on the protocols themselves, showing the how complexity grows with input size and how parameters such as cryptographic algorithms and key sizes affect ef ciency are available in [5]. The running times from our experiments are shown in the following table, where times are cumulative times given in seconds. The experiments were performed on a cluster of seven 2GHz Pentium IV machines with one host designated as the originator and the rest designated as remote hosts.

Application	Protocol		
	Insecure	ACCK	TX
Maxbid	0.11	0.53	12.24
BuyAgent	0.12	1.95	69.2

Note that in all cases except the more involved BuyAgent application running under the TX protocol, the times are just a few seconds per host visited, which is quite a low overhead for the rigorous security guarantees that these protocols provide.

5. CONCLUSIONS

We have developed a comprehensive security framework called SAgent for the JADE multiagent platform. Our framework allows the development of secure protocols and mobile agent applications, and can be used to provide both con dentiality and integrity protections for agent data. Use of generic, application-independent interfaces and data formats within SAgent allows programmers to create applications and protocols with ease and transparency. We also provide the rst software implementations of the ACCK [2] and the TX [7] mobile agent privacy protocols, with experimental results showing the feasibility of these software-only protections within our framework. We leave as future work the inclusion of support for hardware-based protections in SAgent, but believe that such an implementation would be straight-forward in a system that provides veri able Java environments through hardware-based attestation.

6. REFERENCES

[1] SAgent software available at:. http://www.cops.csci.unt.edu/sagent/.

[2] J. Algesheimer, C. Cachin, J. Camenisch, and G. Karjoth. Cryptographic security for mobile code. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 2 11, 2001.

[3] C. Bryce. A security framework for a mobile agent system. In *Proc. of the 6th European Symposium on Research in Computer Security*, volume 1895 of *Lecture Notes In Computer Science*, pages 273 290, 2000.

[4] V. Gunupudi and S. Tate. SAgent: A security framework for JADE. Under review available as CoPS Lab Technical Report 2005-01.

[5] V. Gunupudi, S. Tate, and K. Xu. Experimental evaluation of security protocols in SAgent. Under review available as CoPS Lab Technical Report 2006-01.

[6] JADE, documentation and resources. available at: http://jade.tilab.com/.

[7] S. R. Tate and K. Xu. Mobile agent security through multi-agent cryptographic protocols. In *Proc. of the 4th International Conference on Internet Computing (IC 2003)*, pages 462 468, 2003.

[8] Trusted Computing Group. Web site. http://www.trustedcomputinggroup.org.

[9] B. Yee. A sanctuary for mobile agents. *Secure Internet Programming, Lecture Notes in Computer Science*, 1603:261 273, 1999.

Coordinating Simple and Unreliable Agents

Kagan Tumer
NASA Ames Research Center
Mailstop 269-4
Moffett Field, CA 94035, USA
ktumer@mail.arc.nasa.gov

ABSTRACT

Coordinating a large number of simple and unreliable agents to achieve a global objective poses a unique design challenge: How to achieve robust system level behavior in spite of the computational/reliability limitations of the agents. We investigate agent coordination in the difficult optimization problem of selecting the subset of sensing devices with distortions that provides the smallest average distortion [1]. We approach problem by assigning an agent to each device and having that agent use a simple reinforcement learning algorithm to determine whether to be part of the aggregate device. Our results show that the right agent reward structure provides significant improvements over both traditional search methods and traditional multi-agent methods. Furthermore, even in extreme cases of agent failures (i.e., half the agents fail during the simulation) this approach still outperforms a failure-free and centralized search algorithm.

1. INTRODUCTION

Coordinating simple and unreliable agents to achieve a global objective is different than traditional coordination problem in that: (i) agents have limited sensing and decision making capabilities; (ii) agents are not be able to model the other agents in the system; and (iii) agents are likely to fail (up to 50% of them) during the simulation. We propose addressing this problem by having Reinforcement-Learning (RL) agent [3] aim to maximize their own utilities and let the system reward structure overcome the limited observation and robustness problems. The design problem we face then, is to determine how best to set the utility functions of the agents in a way that will lead to good values of the global utility, without involving difficult to scale external mechanisms. Note that though the agents have simple decisions to make, this is still fundamentally a multi-agent problem: Each agent autonomously makes a decision at each time step based on its estimate of the reward it will receive; and the system is fully distributed as each agent has full autonomy over its actions.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

For the joint action of agents working in such a system to provide good values of the global utility, we must both ensure that the agents do not work at cross-purposes, and that each one has a learning problem that is solvable with limited resources. Typically these two requirements are in conflict with one another. For example, providing each agent with the global utility will ensure that they will not work at cross purposes. However, each agent then faces a difficult signal to noise problem as the reward of each agent depends on the actions of all the other agents. At the other extreme, providing each agent with a simple, local utility function will provide a clear signal, but may not necessarily lead the system to high values of global utility. The challenge is is to find the best trade-off between these two requirements.

2. COMBINATION OF FAULTY DEVICES

The faulty device combination problem arises when many simple and noisy observational devices (e.g., nano or micro devices, low power sensing devices) attempt to accurately determine some value related to the phenomenon they're observing [1]. Each device provides a single number that is slightly off, similar to sampling a Gaussian centered on the value of the real number. The problem is to choose the subset of a fixed collection of the devices so that the average (over the members of the subset) distortion is as close to zero as possible. Formally, the problem is to minimize

$$\epsilon \equiv \frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} , \quad (1)$$

where $n_j \in \{0, 1\}$ is whether device j is or is not selected, and there are N devices in the collection, having associated distortions $\{a_j\}$. This is a hard optimization problem that is similar to known NP-complete problems such as subset sum or partitioning [1], but has two twists: the presence of the denominator and that $a_j \in R \, \forall j$. We denote the global utility function by $G = -\epsilon$ (so that to be consistent with the concept of "utility", the goal is to "maximize" G). The system is composed of N agents, each responsible for setting one of the n_j. The problem we face is in determining the agent utilities so that, as the agents learn to maximize their own utilities, they also maximize G.

2.1 Expected Difference Utility

The first agent utility we discuss is the **Estimated Difference Utility**, EDU, given by [5, 6]:

$$EDU_i \equiv G(z) - E_{z_i}[G(z)|z_{-i}] \quad (2)$$

where z is the full system state, z_i is the components of z that depend on agent i's actions, z_{-i} are the components of z that do not depend on the actions of agent i, and $E_{z_i}[G(z)|z_{-i}]$ is the expected value of G over the possible actions of agent i. Using Eq. 1, EDU leads to:

$$\begin{aligned} EDU_i(z) = & - \frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} \\ & + \left(p(n_i = 1) \frac{|\sum_{j \neq i}^{N} n_j a_j + a_i|}{\sum_{k \neq i}^{N} n_k + 1} \right. \\ & + \left. p(n_i = 0) \frac{|\sum_{j \neq i}^{N} n_j a_j|}{\sum_{k \neq i}^{N} n_k} \right) \end{aligned} \quad (3)$$

where $p(n_i = 1)$ and $p(n_i = 0)$ give the probabilities that agent i set its n_i to 1 or 0 respectively. In what follows, we will assume that those two actions are equally likely (i.e., for all agents i, $p(n_i = 1) = p(n_i = 0) = 0.5$).

Depending on which action agent i chose (0 or 1), EDU(z) reduces to:

$$0.5 \left(\frac{|\sum_{j=1}^{N} n_j a_j - a_i|}{\sum_{k=1}^{N} n_k - 1} - \frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} \right) \text{ if } n_i = 1 , \quad (4)$$

or:

$$0.5 \left(\frac{|\sum_{j=1}^{N} n_j a_j + a_i|}{\sum_{k=1}^{N} n_k + 1} - \frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} \right) \text{ if } n_i = 0 . \quad (5)$$

Note that in this formulation, EDU provides a very clear signal. Because EDU is aligned with G [5](i.e., actions that improve EDU also improve G) if EDU is positive, the action taken by agent i was beneficial to G, and if EDU is negative, the action was detrimental to G. Thus an agent trying to maximize EDU will efficiently maximize G, without explicitly trying to do so. Furthermore, note that the computation of EDU requires very little information. Any system capable of broadcasting G can be minimally modified to accommodate EDU. For each agent to compute its EDU, the system needs to broadcast the two numbers needed to compute G: the number of devices that were turned on (i.e., the denominator in Equation 1) and the associated subset distortion as a real number (i.e., the numerator in Equation 1 before the absolute value operation is performed. Based on those two numbers, the agent can compute its EDU.

2.2 Wonderful Life Utility

The second utility we consider is the **Wonderful Life Utility** [5, 6], given by:

$$WLU_i \equiv G(z) - G(z_{-i}). \quad (6)$$

The major difference between EDU and WLU is in how they handle z_{-i}. EDU provides an estimate of agent i's impact by sampling all possible actions of agent i whereas WLU simply removes agent i from the system. For this application, the WLU becomes:

$$WLU_i(z) = -\frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} + \frac{|\sum_{j \neq i}^{N} n_j a_j|}{\sum_{k \neq i}^{N} n_k} \quad (7)$$

Note however, that unlike with EDU, the action chosen by agent i has a large impact on the WLU. If agent i chooses action 0, the two terms in Equation 7 are identical, resulting in a WLU of zero. Depending on which action agent i chose (0 or 1), WLU reduces to 0 if $n_i = 0$ or:

$$\frac{|\sum_{j=1}^{N} n_j a_j - a_i|}{\sum_{k=1}^{N} n_k - 1} - \frac{|\sum_{j=1}^{N} n_j a_j|}{\sum_{k=1}^{N} n_k} \text{ if } n_i = 1 . \quad (8)$$

In this formulation, unlike EDU, WLU provides a clear signal only if agent i had chosen action 1. In that case, a positive WLU means that the action was beneficial to G, and a negative WLU means that the action was detrimental for G. However, if agent i had chosen action 0, it receives a reward of 0 regardless of whether that action was good or bad for G. This means that on average half the actions an agent takes will be random as far as G is concerned.

3. EXPERIMENTAL RESULTS

Each agent used a simple reinforcement learning algorithm, where each action is associated with the value of the utilities it has yielded. Agents choose actions to maximize those utility values. However, to spur exploration, rather than selecting the action with the highest expected utility, the agents used a Boltzman distribution over those utility values to select an action. In order to allow the agents to gather data, the agents took random actions for the first 20 time steps. The results are based on 20 different $\{a_i\}$ configurations, where each $\{a_i\}$ is selected from a Gaussian distribution with zero mean and unit variance. For each configuration, the experiments were run 50 times (i.e., each point on the figures is the average of $20 \times 50 = 1000$ runs).

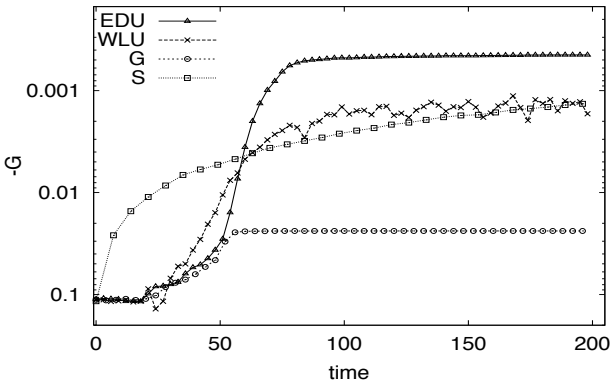


Figure 1: System Performance, N=100.

3.1 Agent Utility Performance

Figure 1 shows the simulation results. The graphs labeled G, EDU and WLU show the performance of agents using G, EDU and WLU as their reward, respectively. S shows the performance of local search where new n_i's are generated at each step by perturbing the current state and selected if the solution is better than the current best solution (in the experiments reported here, 25% of the actions were randomly changed at each time step, though somewhat surprisingly, the results are not particularly sensitive to this parameter). This local search algorithm provides the performance of an algorithm with centralized control. Because the runs are only 200 time steps long, algorithms such as simulated annealing do not outperform local search: there is simply no time for an annealing schedule. Agents use G directly have

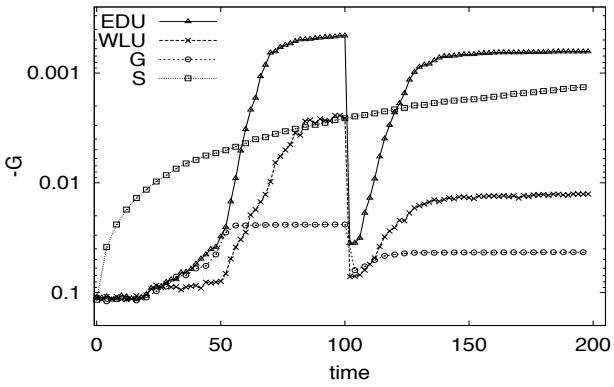


Figure 2: System performance for 100 agents, 20% of which fail at time t=100.

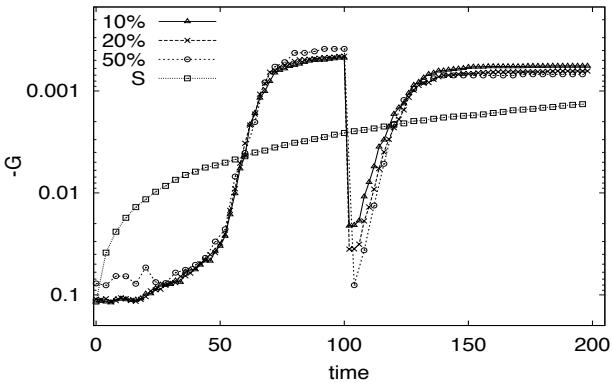


Figure 3: Effect of agent failures on EDU for 100 agents (S has no agent failures).

a difficult time learning. The noise in the system is simply too large for such agents to learn how to select their actions. WLU performs at the level of the centralized algorithm. Because agents only receive useful feedback when they take one of the two actions, the performance is between that of G and EDU. EDU outperforms both the other RL algorithms and the centralized search algorithm. This is due to EDU receiving a clear signal for each of the two actions.

3.2 Handling Agent Failures

In order to evaluate the robustness of the proposed utility functions for multiagent coordination, we tested the performance of the system when a subset of the agents failed during the simulation. At $t = 100$, a certain percentage of agents failed (e.g., were turned off) simulating hazardous condition with agents malfunctioning. The relevance of this experiment was in determining whether the proposed utility functions require all or a large portion of the agents to perform well to be effective, or whether they can handle sudden changes to their environment.

Figure 2 shows the performance of 100 agents where 20% of the agents fail at time step $t = 100$. The results of the centralized search algorithm with no failures ("S" from Section 3.1), is also included for comparison. In these experiments, the agent learning algorithms were not adjusted to account for the change in the environment. In agents that continued to function, the learning proceeded as though nothing had happened. As a consequence, not only did the agents need to overcome the sudden change in their task but they had to do so with parameters tuned to the previous environment. Despite these limitations, EDU recovers rapidly from having 20% of the agents fail. Note this is a powerful results: a distributed algorithm with only 80% functioning agents, each tuned to a different environment outperforms a 100% functioning centralized algorithm.

Figure 3 shows the performance of EDU when the percentage of agent failures increases from 10 to 50%. For comparison purposes, the search results (From Section 3.1) are also included. After the initial drop in performance when the agents stop responding, the system using EDU recovered rapidly and outperformed the fully functioning and centralized search algorithm. These results demonstrate both the adaptability of EDU and its robustness to failures of individual agents, even in extreme cases where half the agents fail during the run.

4. DISCUSSION

The combination of faulty devices is a simple abstraction of a problem that will loom large in the near future: How to coordinate a large number of agents – many of which may have limited access to information and perform unreliably – to achieve a prespecified global objective. This problem is fundamentally different from traditional multi-agent problems in at least three ways: (i) the agents have limited sensing and decision making capabilities; (ii) the agent do not model the actions of other agents; and (iii) the agents are unreliable and failure-prone.

The work summarized in this article is based on ensuring coordination among simple and unreliable agents by providing them with the proper reward structure. This approach is well-suited for addressing coordination in large scale cooperative multi-agent systems where the agents do not have preset and possibly conflicting objectives, or when the agents do not need to hide their objectives. The results shows the promise of this method by providing improvements over both a centralized algorithm and a multi-agent system using a team game approach. Furthermore, when as many as half the agents fail during simulations, the proposed method still outperforms a fully functioning centralized search algorithm.

Acknowledgments: The author would like to thank David Wolpert for invaluable discussions and for bringing the faulty devices problem to his attention, and Adrian Agogino for his many comments.

5. REFERENCES

[1] D. Challet and N. F. Johnson. Optimal combinations of imperfect objects. *Phys. Rev. Let.*, 89:028701, 2002.

[2] T. Hogg and B. A. Huberman. Controlling smart matter. *Smart Materials and Structures*, 7:R1–R14, 1998.

[3] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[4] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[5] K. Tumer and D. Wolpert. A survey of collectives. In *Collectives and the Design of Complex Systems*, pages 1,42. Springer, 2004.

[6] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.

The Emergence of Global Properties from Local Interactions

Static Properties and One-Dimensional Patterns

Daniel Yamins
Division of Engineering and Applied Sciences
Harvard University Cambridge, MA 02138 USA
yamins@fas.harvard.edu

ABSTRACT

Groups of simple objects, interacting under local rules, can produce global properties of great complexity and rich functionality. In this paper, we address of whether a given global property can emerge from purely local interactions. We provide a necessary condition for the case of static properties, and provide complementary sufficient conditions for patterns on the special case of one-dimensional lattices. This work shows that two general considerations, local checkability and symmetry, are the main determinants of "emergeability," at least in these special cases. This work also shows that relatively unified constructions provide solutions to large classes of patterns.

A General Model

Groups of simple objects, interacting under local rules, can produce global properties of great complexity and rich functionality ([1, 2, 3, 4]). In this paper, we address of whether a given global property can emerge from purely local interactions. First, we describe a model of stateless, stigmergy-less agents. The three fundamental objects in the model are configurations, local dynamics, and global properties. Configurations represent agents and the spaces in which they are situated as labeled graphs.

Definition 1 ***[Configurations]*** *(Fig. 1)*
A configuration *is a graph $X = (N_X, E_X)$ of the form*

$$X = (N_X, E_X) = (N_a \cup N_g, E_a \cup E_g),$$

in which: N_a is a set of finite set of nodes labeled by the symbol a; N_g is a set of nodes labeled by the symbol g; E_g is a set of edges (directed or undirected) between nodes in N_g; and E_a is a set containing, for each $x \in N_a$, one directed edge of the form (x, y), where y is a node in N_g that is unique for each x.

Denote by $\mathcal{C}_G$ the set of configurations for which $(N_g, E_g) = G$. For a set $\mathcal{G}$ of graphs G (called underlying spatial graphs*), let $\mathcal{C}_{\mathcal{G}} = \bigcup_{G \in \mathcal{G}} \mathcal{C}_G$.*

For example, the "directed line", denoted $\overrightarrow{\mathbb{Z}}$, is the one dimensional integer lattice with global orientation given by taking $N_g = \{(g, n) | n \in \mathbb{Z}\}$ and $E_g = \{(i, i+1) \mid i \in \mathbb{Z}\}$. The graph $\overline{\mathbb{Z}}$, the "undirected line", has the same N_g, but $E_g = \{\{n, n+1\} \mid n \in \mathbb{Z}\}$. The graph G could also be higher dimensional lattices $\overrightarrow{\mathbb{Z}}^n$, $\overline{\mathbb{Z}}^n$ defined in analogy to their one-dimensional versions, in addition to rings, spheres, and graphs with less regular structure.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

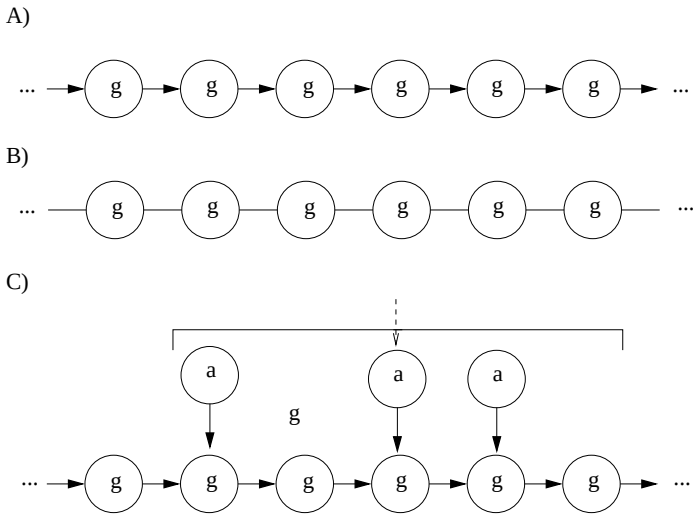


Figure 1: A) The infinite integer lattice with global orientation ("the directed line"). B) The infinite integer lattice without orientation. C) A configuration of three agents on the directed line. The bracket above the middle agent indicates the ball of radius 2 around it.

For a configuration $X = (N_g \cup N_a, E)$, let the *agents of* X be the set $Ag(X) = N_a$, and $|Ag(X)| = |N_a|$ be the number of agents in X. Given $x \in Ag(X)$, let the unique node number m for which $(x, (g, m)) \in E_a$ be denoted by $pos(x, X)$, the *position* of x in X. For $x, y \in N_g$, let $dist(x, y, X)$ be the graph distance in the underlying spatial graph G of X; this extends to a distance between $x, y \in Ag(X)$ as the graph distance between $pos(x, X)$ and $pos(y, X)$. For $x \in X$, define the *neighborhood of radius r about x in X*, denoted $B_r(x, X)$, as the subgraph $(N_B, E_B) \subset (N_X, E_X)$ where $N_B = \{y \in N_X | dist(x, y, X) \leq r\}$. Let $\mathcal{C}_{\mathcal{G}}(r)$ denote the set of all r-neighborhoods $B_r(x, X)$ for $x \in Ag(X)$ and $X \in \mathcal{C}_{\mathcal{G}}$. A configuration X is *r-connected* if $\bigcup_{x \in Ag(x)} B_r(x, X)$ is a connected subgraph of X, and r-disconnected if it is not r-connected. The set of r-connected configurations is denoted by $\mathcal{C}_{\mathcal{G}}[r]$. For two subconfigurations $A = (N_A, E_A)$ and $B = (N_B, E_B)$, define the "graph sum" $A \oplus B = (N_A \cup N_B, E_A \cup E_B)$.

The dynamics of the model are specified by local agent-based rules of motion. For each agent $x \in Ag(X)$, the rule looks at the neighborhood B of some fixed radius r around x and decides whether to move x from its initial position $pos(x, B)$ to some other point g' that is at most one position away. Formally, such a rule can be described by a map f such that for each r-neighborhood $B_r(x, X) = (N_B, E_B)$,

$$f : (N_B, E_B) \mapsto (N_B, E_B \setminus \{(x, pos(x, B))\} \cup \{(x, g')\}) \quad (1)$$

where $dist(x, g', B) \leq 1$.

Definition 2 ***[Local Rules]*** *A* local rule with information radius r *is any function* $f : \mathcal{C}_\mathcal{G}(r) \to \mathcal{C}_\mathcal{G}(r)$ *that obeys condition 1 above. Let* $\mathcal{D}_r$ *denote the set of all such* f*, and* $\mathcal{D} = \cup_r \mathcal{D}_r$*. We write* $r(f)$ *for the information radius of* f*.*

When f fixes an agent $x \in X$, we write $[f(x, X)] = S$ (S stands for "stay").

Given a sequence $s = (s_1, s_2, \dots,) \in Ag^\infty(X)$ of calls to agents and an initial configuration X_0, we could first apply f to (x_1, X_0); we could then apply f to $(x_2, X_1 = f(x_1, X_0))$, and repeat this iteratively. The resulting sequence represents trajectories generated by totally asynchronous updating, denoted by $f_s^n(X_0)$. Any such trajectory has an *ω-limit set*, denoted by $\Omega(f, s, X_0)$, which is the set of configurations that appear infinitely often in $\{f_s^n(X_0)\}$. $\Omega(f, s, X_0)$ is empty when no well-defined limit actually exists.

A choice of models for the concurrency of agent calls is given by specifying a set of sets $S = \{S_X | X \in \mathcal{C}_\mathcal{G}\}$ of valid call sequences that is invariant under the graph-equivalence $\cong$ on $\mathcal{C}_\mathcal{G}$. Totally asynchronous update (with no restriction on the relative frequence of calls to different agents) corresponds to setting $S_X^{asynch} = Ag(X)^\infty$, whereas totally synchronous update corresponds to the choice $S_X^{synch} = \{(Ag(X), Ag(X), Ag(X), \dots,)\}$.

Global properties are given by specificying a target set $P \subset \mathcal{C}_\mathcal{G}$ of 'desired' configurations. For example, *patterns*:

Definition 3 ***[Patterns]*** *Fix a graph* G*. Let* $(q)^{\oplus k}$ *denote the set of subconfigurations* Y *of graphs over* G *of the form* $Y = q_1 \oplus q_2 \oplus \dots \oplus q_k$*, where the* $q_1, \dots, q_k$ *are all disjoint and graph-equivalent to* q*. A* pattern *over* G *is a subset* $P \subset \mathcal{C}_G$ *for which there exists a finite* q *such that*

$$P = \bigcup_{k=0}^{<\infty} \{Y \oplus G \mid Y \in (q)^{\oplus k}\}.$$

In this case, we write $P = \langle q \rangle$*. We define the* length *of* P*, denoted* $|P|$*, to be the length of the underlying spatial graph of* Y *and the size of* P*, denoted* $|Ag(P)|$*, to be the number of agents in* q*.*

Given a property $P \subset \mathcal{C}_\mathcal{G}$, our motivating interest is to find (or prove the non-existence of) *solutions* to it, those dynamic rules whose trajectories satisfy the property.

Definition 4 ***[Solutions]*** *A dynamic rule* $f \in \mathcal{D}$ *is a* P-solution over $(\mathcal{G}, S)$ *if, for all* $X \in \mathcal{C}_\mathcal{G}$*, and all call-sequences* $s \in \mathcal{S}_X$*,* $\Omega(\{f_s^n(X)\}) \subset P$ *holds.*

The solution space $\mathcal{F}(\mathcal{G}, S, P)$ *is the set of all* φ*-solutions, that is*

$$\mathcal{F}(\mathcal{G}, S, P) = \{f \in \mathcal{D} | f \text{ is a } P\text{-solution over } (\mathcal{G}, S)\}.$$

Pi is locally solvable *when* $\mathcal{F}(\mathcal{G}, S, \varphi) \neq \emptyset$*. Otherwise,* P *is* locally unsolvable.

Local Checkability

Definition 5 *A* local check scheme with radius r *is a function* $\Theta : \mathcal{C}_\mathcal{G}(r) \to \{0, 1\}$*. The* check radius $\Xi(P)$ *of a property* P *is the minimal* $r > 0$ *such that there exists local check scheme* Θ *with* $r(\Theta) = r$ *such that*

$$\bigwedge_{a \in Ag(X)} (\Theta(B_r(a, X)) = 1) \ \wedge \ X \in \mathcal{C}_\mathcal{G} \ \Rightarrow \ X \in P,$$

if such exists. The connected check radius $\Xi^c(P)$ *is the minimal* $r > 0$ *such that there exists* Θ *with* $r(\Theta) = r$ *such that*

$$\bigwedge_{a \in Ag(X)} (\Theta(B_r(a, X)) = 1) \ \wedge \ X \in \mathcal{C}_\mathcal{G}[r] \ \Rightarrow \ X \in P$$

if it exists.

If $\Xi(P) < \infty$, we say P is locally checkable.

Proposition 1 *For all* $\mathcal{G}$*,* S *and* P*:* $f \in \mathcal{F}(\mathcal{G}, S, P)$ *only if* $r(f) \geq \Xi(P)$*.*

Hence, $\Xi(P) = \infty \ \Rightarrow \ \mathcal{F}(\mathcal{G}, S, P) = \emptyset$; and $\Xi^c(P) = \infty \ \Rightarrow \mathcal{F}(\mathcal{G}, S, P) = \emptyset$. Thus:

Corollary 1 *Static properties* P *is locally solvable over* $\mathcal{G}$ *only if it is are locally checkable over* $\mathcal{G}$*:*

Obviously, patterns *do* meet the local checkability condition.

Corollary 1 establishes a necessary condition that must be met for any static property to be emergeable. In the next section, we answer the converse question when restricted to the special case of connected patterns on $\mathcal{G} = \{\overrightarrow{\mathbb{Z}}\}, \{\overline{\mathbb{Z}}\}$, and $\mathcal{R}$, the locally one-dimensional lattices. We begin by defining some notation.

Patterns on the Directed Line

Given $X \in \mathcal{C}_{\overrightarrow{\mathbb{Z}}}$, and $x, y, \in X$, we say that x is *to the right* of y if there exists a directed path from $pos(x, X)$ to $pos(y, X)$ in $\overrightarrow{\mathbb{Z}}$, and if so, we write $y \leq x$. If such a path does not exist, then we say x is *to the left* of y, and write $x \leq y$. For any connected subconfiguration $Y \subset X$, let $min(Y)$ be the left-most spatial node in Y, and $max(Y)$ be that of the right-most. For $a, b \in \overrightarrow{\mathbb{Z}}$ with $a \leq b$, let $X[a, b]$ be the maximal connected subconfiguration of $Y \subset X$ with $min(Y) = a$ and $max(Y) = b$. For $x \in Ag(X) \in \mathcal{C}_{\overrightarrow{\mathbb{Z}}}$, define x^{+k} to be the kth agent to the right of x, if such exists; and x^{-k} to be the k-th agent to the left of x, if such exists. Define $re(X)$ to be the unique $x \in Ag(X)$ for which $x^+ = x^{+1}$ does not exist, and $le(X)$ to be the unique agent for which $x^- = x^{-1}$ does not exist. (These are the "right-end" and "left-end" agents of X, respectively.)

Any local rule f with radius r operating over the directed line $\overrightarrow{\mathbb{Z}}$ has three possible actions on any configuration $B_r(x, X)$. It can either move the agent x to the left, which we notate as $[f(x, X)] = L$; or it can move x to the right, which we notate as $[f(x, X)] = R$; or f can fix x, in which case $[f(x, X)] = S$ is just the same as previous defined notation.

It turns out that on $\overrightarrow{\mathbb{Z}}$, local checkability is sufficient for solutions to P to exist. To prove this, given a pattern P over $\overrightarrow{\mathbb{Z}}$, we construct a local rule that generates it. Let q be a generator in standard form of P. We first construct a rule that generates q when exactly $|Ag(q)|$ are present close enough together. In particular, let $\mathcal{Y}_q$ be the set of configurations Y with $|Ag(q)|$ agents and such that $dist(le(Y), re(Y)) \leq |q|$. Let $Y \in \mathcal{Y}_q$. Then define $T \in \mathcal{D}_{|q|}$ by

$$[T(a, Y)] = \begin{cases} L & \text{if } a = a_Y^* \\ S & \text{otherwise} \end{cases}$$

where

$$a_Y^* = re(\{a \in Ag(Y) | (pos(a) \leq pos(a'_Y)) \text{ and } (dist(a, a^-) > 1)\}).$$

in which

$$a'_Y = re(\{a \in Ag(Y) | (dist(a, a^+, Y) < dist(a, a^+, q)) \text{ or } (dist(a, a^-, Y) > dist(a, a^-, q))\}).$$

T has the three properties:

- $\Omega(T, s, Y) = \overrightarrow{\mathbb{Z}} \oplus q$, for all $Y \in \mathcal{Y}_q$ and $s \in S_Y$
- $[T(a, X)] \neq R$ for all $X \in \mathcal{C}_r$ and
- $dist(le(T_s^n(Y)), re(T_s^n(Y))) \leq |q|$ for all n, for all $Y \in \mathcal{Y}_q$ and $s \in S_Y$.

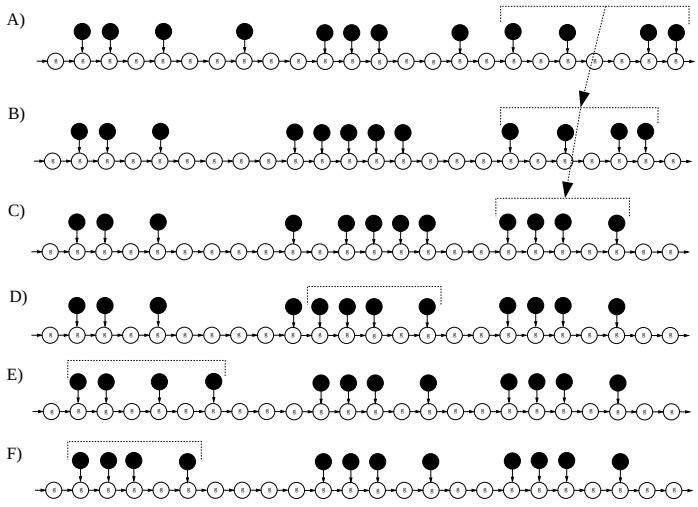


Figure 2: Illustration of the operation of the local rule constructed to solve pattern formation. A) Initial condition. B) Right-most agents comes together. C) Right-most group implements local rule T. In D) through F) this procedure repeats, propagating to the left.

In other words, T is like a solution to one unit of the pattern (Fig. 2). Our task is define a local rule which forces agents to come together closely enough to implement T, and properly draws boundaries between the units.

For $X \in \mathcal{C}$, and $a \in Ag(X)$, define $\widetilde{B}(a)$ to be the set of agents b in X with $pos(b) \leq pos(a)$ such that $X[pos(b), pos(b) + |q|]$ contains $|Ag(q)|$ or more agents but $X[pos(b^{+j}), pos(b^{+j}) + |q|]$ does not for $j = 1, \ldots, |Ag(q)| - 1$. If $\widetilde{B}(a) \neq \emptyset$, then let $a' = re(\widetilde{B}(a))$, the right-most agent in $\widetilde{B}(a)$. Then define $\gamma_X : Ag(X) \to \{0, 1, \ldots, |Ag(X)|\}$ by

$$\gamma_X(a) = \begin{cases} \text{k+1} & \text{where } a = (a')^{+k}, \text{ if } \widetilde{B}(a) \neq \emptyset \text{ and } k < |Ag(q) \\ 0 & \text{otherwise} \end{cases}.$$

Note that $\gamma_X(le(X)) \leq 1$ for all X. γ is a check which draws boundaries between units, each new unit starting again when γ_X is 1; and when not enough agents are close enough together for a unit to be declared, γ_X is zero.

We put γ and T together now. Define $f_T[P] \in \mathcal{D}_{2|q|+1}$ by: If $|Ag(X[pos(a) - 2|q| - 1, pos(a) - 1])| = 0$, then:

$$[f_T[P](a, X)] = \begin{cases} L, \text{ if } |X[pos(a), pos(a) + |q|]| \geq |Ag(q)| \\ \text{ and } \gamma_X(a) = 0 \\ [T(a, \overrightarrow{\mathbb{Z}} \oplus X[pos(a), pos(a) + |q|])], \text{ if } \gamma_X(a) = 1 \\ S, \text{ otherwise} \end{cases}$$

and if $|Ag(X[pos(a) - 2|q| - 1, pos(a) - 1])| > 0$, then

$$[f_T[P](a, X)] = \begin{cases} L, \text{ if } \gamma_X(a) = 0 \text{ and } dist(a, a^+) \neq 2|q| + 1 \text{ or} \\ X[pos(a), pos(a) + |q|] \cong q \text{ and} \\ dist(a, a^-) = 2|q| + 1 \\ [T(a, \overrightarrow{\mathbb{Z}} \oplus X[pos(a^{+l}), pos(a^{+l}) + |q| - 1])], \\ \varphi_X(a) \neq 0, \ dist(a, a^+) \neq 2|q| + 1 \\ S \text{ otherwise} \end{cases}$$

where $l = 1 - \gamma_X(a)$. The operation of $f_T[P]$ is described pictorially in figure 2. For each P, $f_T[P]$ is a local rule with information radius $2|q| + 1$. It can be shown that $f_T[P]$ is a solution to the connected P-pattern formation problem, yielding:

Theorem 1 *All connected patterns can be solved over* $\overrightarrow{\mathbb{Z}}$.

Symmetry

Any configuration X possess a *symmetry group*, $Sym(X)$; these are the permutations σ on the set $\mathbb{N}$ of node identity numbers that induce a graph automorphism on X. The underlying spatial graphs also of course have symmetry groups (think of them as trivial configurations with no agents). For example, $Sym(\overrightarrow{\mathbb{Z}}) = \mathbb{Z}$; for each $n \in \mathbb{Z}$, the symmetry σ_n corresponds to shifting all nodes in $\overrightarrow{\mathbb{Z}}$ by n steps. $Sym(\overline{\mathbb{Z}}) = \mathbb{Z} \oplus \mathbb{Z}_2$, because in addition to having all of the shift symmetries which generate the $\mathbb{Z}$ summand, the undirected line also has a flip symmetry σ^* which generates the $\mathbb{Z}_2$ summand. For any configuration X over $\overrightarrow{\mathbb{Z}}$ with $|Ag(X)| > 0$ (and finite), the symmetry group $Sym(X)$ is trivial. That is because none of the shift symmetries can take the "end-most" agent to anything other than itself. However, for all configurations Y over $\overline{\mathbb{Z}}$ that are symmetric about their middle agent(s), $Sym(Y) = \mathbb{Z}_2$.

We can put a measure on the amount of symmetry that a given geometry $\mathcal{G}$ has by measuring the size of the symmetry groups for non-trivial configurations over $\mathcal{G}$. In this sense, $\overline{\mathbb{Z}}$ has more symmetry than $\overrightarrow{\mathbb{Z}}$ (which has none). This non-zero amount of symmetry places prevent emergeability on this geometry, as reflected in the following:

Theorem 2 *No patterns can be solved over* $\overline{\mathbb{Z}}$ *in the synchronous model of concurrency.*

The proof of Theorem 2 relies heavily on the choice of the synchronous concurrency model. In fact, the small amount of symmetry-breaking which asynchrony allows is enough solve all patterns, via a construction analagous to that used in the case of $\overrightarrow{\mathbb{Z}}$ in theorem 1:

Theorem 3 *All patterns can be solved over* $\overline{\mathbb{Z}}$ *in the asynchronous concurrency model.*

Having considered $\overrightarrow{\mathbb{Z}}$ and $\overline{\mathbb{Z}}$, let us consider the set of ring graphs $\mathcal{R} = \{\mathcal{R}_m | m \in \mathbb{N}\}$, where $\mathcal{R}_m$ is the ring graph with m nodes.

Theorem 4 *Only trivial patterns over the ring lattices* $\mathcal{R}$ *can be solved.*

Theorem 4 says that the failure of existence of solutions to pattern formation is worse on $\mathcal{R}$ that it was on $\overline{\mathbb{Z}}$, since failure holds in both the synchronous and asynchronous concurrency models. The essence of this failure is, once again, symmetry. Note that $Sym(\mathcal{R}) = \bigcup_{m \in \mathbb{N}} \mathbb{Z}_m$. Unlike $\overrightarrow{\mathbb{Z}}$ which has no symmetries on non-trivial configurations, and $\overline{\mathbb{Z}}$, which has at most the finite group $\mathbb{Z}_2$, $\mathcal{R}$ has infinitely many symmetries over non-trivial configurations, and it is precisely these symmetries that are exploited in the proof of theorem 4.

In summary, proposition 1 and theorems 1, 2, 3, and 4, suggest that two rough criteria are important for whether a given global property is emergeable: A) whether it is **locally checkable** and B) whether the property **respects the symmetry** of the underlying space.

1. REFERENCES

[1] S. Camazine, J. L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organizing Biological Systems*. Princeton Univ. Press, 2001.

[2] R. Nagpal. *Programmable Self-Assembly: Constructing Global Shape Using Biologically-Inspired Local Interactions and Origami Mathematics*. PhD thesis, MIT, 2001.

[3] C. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Proc. SIGGRAPH*, 1987.

[4] L. Wolpert. Positional information and the spatial pattern of cellular differentiation. *J. Theor. Bio.*, 25(1), 1969.

Auctions and electronic markets

Sequences of Take-It-or-Leave-It Offers: Near-Optimal Auctions Without Full Valuation Revelation*

Tuomas Sandholm
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
sandholm@cs.cmu.edu

Andrew Gilpin
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
gilpin@cs.cmu.edu

ABSTRACT

We introduce *take-it-or-leave-it auctions (TLAs)* as an allocation mechanism that allows buyers to retain much of their private valuation information, yet generates close-to-optimal expected utility for the seller. We show that if each buyer receives at most one offer, each buyer's dominant strategy is to act truthfully. In more general TLAs, the buyers' optimal strategies are more intricate, and we derive the perfect Bayesian equilibrium for the game. We develop algorithms for finding the equilibrium and also for optimizing the offers so as to maximize the seller's expected utility. In several example settings we show that the seller's expected utility already is close to optimal for a small number of offers. As the number of buyers increases, the seller's expected utility increases, and becomes increasingly (but not monotonically) more competitive with Myerson's expected utility maximizing auction.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: General

General Terms

Economics, Algorithms

Keywords

auctions, optimal auction design, equilibrium computation, perfect Bayesian equilibrium, multiagent systems

1. INTRODUCTION

Auctions have emerged as a key mechanism for task and resource allocation in multiagent systems. We focus on the problem of allocating an indivisible good to one of a group of interested buyers, while maximizing the seller's expected utility.[1] In this setting, the *Myerson auction* yields the highest expected utility to the seller, compared to *any* other allocation mechanism [13]. Despite being the optimal allocation mechanism, the Myerson auction has several undesirable features. First, buyers must fully reveal their true valuations for the good.[2] Buyers may be unwilling to reveal private information because such information is proprietary or its revelation would have a negative strategic long-term impact. For example, after such an auction the buyers will be at a disadvantage in future negotiations with the seller (who may be able to extract more surplus in the future using information about the buyers' valuations). Similar long-term adverse effects have been observed in other truthful full-revelation auction mechanisms [17]. Second, buyers may not understand the complex, unintuitive rules of the Myerson auction, and therefore choose to not participate. Third, for a layman, it is unintuitive to submit true valuations because bid shading is rational in many commonly used auction mechanisms. Buyers are thus likely to shade bids even in Myerson auctions, with unpredictable and suboptimal results. For these reasons and potentially others, to our knowledge, the theoretically elegant Myerson auction is not being used anywhere.

In the most commonly used auction mechanisms—the *English auction* (a.k.a. ascending open-cry auction) and the *Dutch auction* (a.k.a. descending auction)—bidders do not reveal their valuations completely [9]. Unfortunately, in these auctions, the seller's expected utility is not optimal and in fact can be arbitrarily far from optimal. The *Vickrey auction* [20] (a.k.a. second-price sealed-bid auction) suffers from both full valuation revelation and seller's expected utility being arbitrarily far from optimal.

In this paper we present a mechanism that achieves close-to-optimal expected utility for the seller while allowing each buyer to retain much of her private valuation information. Our *take-it-or-leave-it auction (TLA)* mechanism consists of a sequence of take-it-or-leave-it offers made by the seller to the buyers, each buyer in turn. Before the auction is held, the seller announces the order in which the buyers receive offers and the amount of each offer. During a fixed time after receiving an offer, the buyer has to accept ("take") the offer, at which point the auction ends, or reject ("leave") the offer, in which case the seller may proceed to make other offers. After rejecting an offer, the buyer never has the option of

*This material is based upon work supported by the National Science Foundation under CAREER Award IRI-9703122, Grant IIS-9800994, ITR IIS-0081246, ITR IIS-0121678, and ITR IIS-0427858, and a Sloan Fellowship.

[1]All of our results apply to reverse auctions as well, in which a single buyer, facing many sellers, wants to minimize her procurement cost.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

[2]Recently, cryptographic techniques have also been proposed to address this drawback [14].

accepting that offer.[3] We show how rational buyers should act in such an auction and how the seller can optimize the sequence of offers given her knowledge about the buyers. In this sense, our approach is *automated mechanism design* [7].

Sequences of take-it-or-leave-it offers have been studied in the context of optimal auctions previously [3]. However, in that paper there is an assumption that the seller can decide the accuracy to which the buyers learn their own valuations. One way they can achieve this is by withholding information. In our model, the seller has no control of the buyers' information.

In many applications the seller may be aware of some statistical information about the buyers. By taking this information into account, the seller can design mechanisms so as to maximize her expected utility. In our setting, this is done by making the assumption that every buyer's valuation is drawn from some known probability distribution. This assumption violates the *Wilson doctrine* which states that mechanisms should be designed without the use of any priors. Some examples of auctions that follow the Wilson doctrine include Vickrey, English, Dutch, and first-price sealed bid auctions. These types of auctions are not optimal, and in fact can be arbitrarily far from optimal. Further, it is generally not possible to design utility maximizing auctions (or even approximately optimal auctions) without using priors. The Myerson auction, on the other hand, does not adhere to the Wilson doctrine, and as a result is able to extract more surplus from the buyers. Just as is done in the Myerson auction, we use priors in our mechanism.

2. THE SETTING AND BACKGROUND

We study the usual auction setting in which a seller has an indivisible good that she can allocate to any one (or none) of a set of buyers $B = \{1, \ldots, n\}$. Each buyer i has a valuation v_i for the good that is drawn from a distribution f_i with cumulative density F_i. The buyers' valuations are drawn independently.[4] The distributions f_i are common knowledge, but the valuation v_i is only known by the buyer i herself. The setting is *symmetric* if each buyer draws her valuation from the same distribution. Otherwise the setting is *asymmetric*. Finally, we make the standard auction theory assumption that there is no aftermarket.

By designing the rules of the auction, the seller tries to maximize her expected utility. The seller has a valuation v_0 (potentially zero) for the good. By the standard quasilinearity assumption about the seller, the seller's utility is v_0 if she keeps the item, and if she allocates the item to a buyer, her utility is the revenue that she receives from the buyers.

To motivate the buyers to participate in the auction, the rules of the auction have to guarantee that each buyer receives no less utility by participating than by not participating. Such *ex post* participation constraints imply that 1) a buyer who does not get the good cannot be charged, and 2) the buyer who receives the good cannot be charged more than her valuation.

[3]At first glance, it appears that the Dutch auction and TLAs are equivalent. This is not true. The Dutch auction is more powerful in the sense that the seller can make an infinite number of offers, and less powerful in the sense that the seller has to make the offers symmetrically to the buyers.

[4]If the valuations can have arbitrary correlations, then even designing an approximately optimal auction is hard computationally [16].

The optimal mechanism for our setting is the Myerson auction [13]. It is the benchmark against which we will compare our TLA mechanism.[5]

DEFINITION 1. (Myerson auction [13])

- *Every buyer i reveals a valuation $\hat{v}_i$.*
- *For each buyer i, compute that buyer's* virtual valuation*: $\psi_i = \hat{v}_i - \frac{1-F_i(\hat{v}_i)}{f_i(\hat{v}_i)}$.*
- *Select the buyer i^* with the highest virtual valuation ψ_{i^*}.*
- *If $\psi_{i^*} > v_0$, buyer i^* receives the item and pays the smallest valuation, $\underline{\hat{v}}_i$, that would still have made her win.*
- *If $\psi_{i^*} \leq v_0$, the seller keeps the good and no payments are made.*

In the symmetric setting, the Myerson auction is a second-price auction with an appropriately tailored reserve price. However, this is not the case in the asymmetric setting.

3. TAKE-IT-OR-LEAVE-IT AUCTION

In this section we define our idea of using a sequence of take-it-or-leave-it offers as an auction mechanism.

DEFINITION 2. (Take-it-or-leave-it auction (TLA)) *Let $\mathcal{A} = \langle b_1, a_1\rangle, \langle b_2, a_2\rangle, \ldots, \langle b_k, a_k\rangle$ be a sequence of k offers. Let an offer $\mathcal{A}_j$ be a tuple $\langle b_j, a_j\rangle$, where a_j is the amount of the offer and $b_j \in B$ is the buyer to whom the offer is made. The entire sequence of offers $\mathcal{A}$ is revealed by the seller to the buyers before the auction. The auction proceeds as follows:*

- *For j from 1 to k*
  - *Let $\langle b, a\rangle = \mathcal{A}_j$. The seller offers the good to buyer b for amount a.*
  - *If buyer b accepts, she gets the good and pays a to the seller. The auction ends.*
  - *If buyer b rejects offer a, the auction continues.*
- *If the auction has reached this point, the seller keeps the good and collects no payments.*

It is worth mentioning that every instance of a TLA is *ex ante* individually rational (the strongest such form) since a buyer can always reject any offer. The notion of incentive compatibility (*i.e.* truthtelling) does not apply here since it's not a direct revelation mechanism.

The following auction mechanism turns out to be an interesting special case:

DEFINITION 3. (Single-offer TLA) *A TLA is a* single-offer TLA *if each buyer gets at most one offer, i.e., $b_q \neq b_r$ for all $q, r \in \{1, 2, \ldots, k\}$, $q \neq r$.*

When we need to differentiate between a single-offer TLA and a TLA in which a buyer may receive multiple offers, we will refer to the latter as a *multiple-offer TLA*.

Note that in a TLA it is entirely possible for the seller to allocate the good to a buyer who does not have the highest valuation, thus making the mechanism *inefficient*. This is also the case in the Myerson auction. It is well-known that in the asymmetric setting there is no mechanism that maximizes both seller's expected utility and efficiency.

[5]The proof of optimality of the Myerson auction assumes that the setting is *regular*, i.e., the hazard rate $f_i/(1-F_i)$ is increasing for each buyer i [13]. This assumption is satisfied in all cases where we benchmark against the Myerson auction (in Section 7), so we are benchmarking against the real optimum.

3.1 Strategies and utility

A formal notation for strategies in a TLA is useful. Strategies for TLAs are slightly different than for other auctions since a single buyer may need to make many decisions during the course of an auction.

DEFINITION 4. (Strategy for a TLA) *A buyer's* strategy *for a TLA is a function from the buyer's valuation and the number of the offer (made to that buyer), to one of two possible actions:* accept *or* reject*:*

$$s_i : \mathbb{R} \times \{j \mid b_j = i, j \in \{1, 2, \ldots, k\}\} \rightarrow \{accept, reject\}.$$

A strategy profile *is* $s = (s_1, \ldots, s_n)$*. A* strategy profile excluding s_i *is* $s_{-i} = \{s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n\}$*, and a* strategy profile s with s_i replaced with s_i' *is* (s_{-i}, s_i')*.*

We make the standard assumption of quasilinearity, that is, a buyer's utility u_i is zero if she does not get the good (and does not pay anything), and $v_i - a$ if she gets the good and pays amount a. Given a strategy profile s and the valuation v_i, the expected utility $u_i(v_i, s)$ of buyer i in a TLA $\mathcal{A}$ is

$$u_i(v_i, s) = \sum_{\{j|b_j=i\}} \left(\prod_{l=1}^{j-1} R_l \right) u_i^j(v_i, s_i)$$

where $u_i^j(v_i, s_i) = \begin{cases} v_i - a_j & if \quad s_i(v_i, j) = accept \\ 0 & if \quad s_i(v_i, j) = reject. \end{cases}$

Here, R_l is the probability that offer l is rejected by buyer b_l if that offer is made. These probabilities depend on the buyers' strategy profile s, as discussed next.[6]

4. EQUILIBRIUM ANALYSIS

We use definitions from game theory to help us understand how rational buyers will behave in a TLA. We use two of the most prevalent solution concepts for sequential games: dominant strategies and perfect Bayesian equilibrium.

4.1 Single-offer TLAs

A strategy s_i^* for a buyer with valuation v_i is called a *dominant strategy* if for all other strategies s_i' and all strategies s_{-i}, we have $u_i(v_i, (s_i^*, s_{-i})) \geq u_i(v_i, (s_i', s_{-i}))$. As the following proposition illustrates, the strategies employed by rational buyers in a single-offer TLA are straightforward and truthful. They are dominant strategies, so a buyer is best off playing that way regardless of what the other buyers' preferences are and even if others play irrationally. Also, a buyer does not have to know much about her own valuation (the determination of which can be a complex optimization problem itself, see e.g. [18, 19, 15, 10]), merely whether it is greater or less than the offer, making a single-offer TLA easy to play.

PROPOSITION 1. *In a single-offer TLA, a buyer's dominant strategy is*

$$s_i^*(v_i, j) = \begin{cases} accept & if \quad v_i > a_j \\ reject & if \quad v_i \leq a_j. \end{cases}$$

The strategy profile $s^* = (s_1^*, s_2^*, \ldots, s_n^*)$ *is thus a dominant strategy equilibrium.*

[6]When buyer i computes the utility, $R_l \in \{0, 1\}$ when $b_l = i$, because the buyer knows her own strategy.

4.2 Multiple-offer TLAs

If the sequence of take-it-or-leave-it offers includes potentially multiple offers per buyer, the buyers' equilibrium strategies become significantly more intricate. The reason is that *a rational buyer will be reluctant to accept an offer at—or somewhat below—her valuation, because there is a chance that the auction will make her a significantly lower offer later on.* The analysis is further complicated by the fact that a buyer's passing on an offer gives a signal to the other buyers and the seller about the buyer's valuation.

In the equilibrium analysis that follows, we study *threshold strategies*. In any finite-action game where types are drawn from independent atomless distributions and each player's utility function does not directly depend on other's types, a pure strategy equilibrium always exists [12]. (This existence result can be extended to the setting with interdependent valuations in several auction-like settings [2].) Clearly, a TLA is such a game. Furthermore, since each buyer receives strictly decreasing offers in an optimal TLA (see Proposition 4), we can restrict our attention to a strategy in which a player can simply decide at the beginning of the game the first offer she will accept. This reasoning partitions the type space into regions, one for each offer. We say these regions are separated by *thresholds*. If the player's type falls within a region, then the first offer the player will accept is the offer corresponding to the region. Thus, without loss of generality we can focus on threshold-based pure strategies in our analysis of an equilibrium of a multiple-offer TLA.

Perfect Bayesian equilibrium (PBE) [8] is the most common solution concept for sequential games with observed actions and private valuations (types). In such a game, with independently drawn valuations, each buyer i has a strategy s_i and beliefs (for each step j of the game, the buyer has, for each other buyer k, a cumulative probability density function F_k^j about the type of k). A strategy profile s^* is a PBE if the following conditions hold: 1) Each buyer updates her beliefs using Bayes rule whenever possible; 2) Whenever it is buyer i's turn to move, s_i prescribes an action that maximizes i's expected utility from then on, given i's beliefs; and the three technical conditions: 3a) At every step of the game, buyer i's beliefs about other buyers are independent of buyer i's valuation, 3b) the beliefs about buyer i can only be changed when i acts, and 3c) at every step of the game, for every triple of buyers i, k, and l ($i \neq l$, $k \neq l$), i and k have the same beliefs about l.

We illustrate the PBE analysis through a 5-offer 2-buyer auction and present the general method after that. For readability, we refer to buyer 1 in the feminine and buyer 2 in the masculine. Suppose that the seller's chosen TLA prescribes that the first offer goes to buyer 1 and the remaining four offers alternate between the buyers.

Buyer 1, when facing the first offer, must decide whether or not she should accept it. If her valuation, v_1, is less than the first offer, a_1, then clearly she must reject. However, the converse is not as simple. *Even though she might stand to gain positive utility by accepting the first offer, she might expect to gain even more utility by rejecting the first offer and accepting a later offer.* For each offer made to a buyer, we would like to compute a *threshold* value above which the buyer will accept and below which she will reject. (If the buyer's valuation is equal to the threshold, she is indifferent between rejecting and accepting.) We will refer to buyer i's

threshold at offer j as t_i^j.

At the last offer made to each buyer, both buyers have dominant strategies. They simply accept if the offer is below their valuation and reject otherwise. This is analogous to the strategies employed by buyers in a single-offer TLA.

Now, consider the third offer (the second offer made to buyer 1). Buyer 1 knows what her utility is if she accepts this offer. She wishes to determine what her *expected* utility is if she rejects the third offer (in a gamble to face the fifth offer). To compute this she needs to compute the probability that buyer 2 rejects the fourth offer. At this point in the game, buyer 1 knows that buyer 2 has rejected the second offer. Assuming buyer 2 is playing in equilibrium, his valuation must be less than t_2^2, his threshold when facing the second offer. Buyer 1 must take this into account when computing the probability that buyer 2 will reject the fourth offer. *That is, buyer 1 updates her belief about the distribution from which buyer 2's valuation is drawn.* Buyer 1 is able to perform these computations using her knowledge about buyer 2's prior distribution, which is common knowledge. After buyer 2 has rejected the second offer, by Bayes rule, the right tail of his valuation distribution (probability density function) above t_2^2 can be cut, and the remaining distribution needs to be rescaled to make its integral one. Thus the new cumulative distribution is $F_2^4(x) = F_2(x)/F_2\left(t_2^2\right)$. (In general, we will write F_i^j to denote the world's belief about buyer i's distribution when facing offer j. If a buyer i is facing her first offer j, then $F_i^j = F_i$.) With this new notation, we can write equations describing the thresholds for buyers 2 and 1 when facing offers 2 and 3, respectively:

$$\begin{aligned} t_2^2 - a_2 &= F_1^3\left(t_1^3\right)\left(t_2^2 - a_4\right) \\ t_1^3 - a_3 &= F_2^4(a_4)\left(t_1^3 - a_5\right). \end{aligned}$$

We evaluate F_1^3 at t_1^3 since we know that buyer 1 will only reject if her valuation is below t_1^3.

Determining the strategy that buyer 1 should use at the first offer is slightly more complicated since buyer 1 could reject the first offer in a hope to accept the third offer or the fifth offer. The following equation describes buyer 1's threshold at the first offer:

$$t_1^1 - a_1 = F_2^2\left(t_2^2\right) \cdot \max\left\{t_1^1 - a_3, F_2^4(a_4)\left(t_1^1 - a_5\right)\right\}.$$

4.2.1 A simplifying observation

At this point we have completely *characterized* the strategies that the buyers will use, but we have said nothing about how to compute these strategies. Computing the strategies is complicated by the max operation in the above equation. The determination of the strategies could be split into cases, with each case corresponding to a particular term within the max operation being greatest. However, as we move beyond 5 offers, this becomes complex. As the number of offers increases by one, the number of max operations in the system increases by one, thus doubling the number of cases to analyze. Therefore, the number of cases is exponential in the number of offers.[7]

This computation would be easier if we could assume that the thresholds only depend on the next offer. In the above equation, this would mean that $t_1^1 - a_3 \geq F_2^4(a_4)\left(t_1^1 - a_5\right)$. We observe that if this inequality does not hold, the third offer would in effect be wasted because it certainly would be rejected. Since this paper is concerned with optimal TLAs, we can assume that such offers will never be made.[8]

4.2.2 Equilibrium for multiple buyers and offers

Algorithm 1 below is a general procedure for generating the system of equations that determines buyers' thresholds. Before describing the algorithm we need to introduce some additional notations. First, we need an expression for evaluating updated distributions:

$$F_i^j(x) = \begin{cases} F_i(x) & \text{if} \quad i \notin \{1, \ldots, b_{j-1}\} \\ \frac{F_i(x)}{F_i\left(t_i^{j'}\right)} & \text{if} \quad i \in \{1, \ldots, b_{j-1}\} \end{cases}$$

where j' is the most recent offer made to buyer i. Second, we need an expression for the probability that a buyer will reject a given offer j:

$$R_j = \begin{cases} F_{b_j}^j(a_j) & \text{if} \quad b_j \notin \{b_{j+1}, \ldots, b_k\} \\ F_{b_j}^j\left(t_{b_j}^j\right) & \text{if} \quad b_j \in \{b_{j+1}, \ldots, b_k\}. \end{cases}$$

The above value R_j depends on whether or not the buyer faces another offer or not. If the buyer does face another offer, then the expression involves a threshold value. Otherwise, the expression only involves the offer. The algorithm below makes a pass through the offers. If an offer is not the last one offered to a buyer, then an equation is output containing an expression for the threshold.

ALGORITHM 1.
// *Output system of threshold equations*

- *For j from 1 to k, If buyer b_j faces another offer j' after offer j*
  - *Then output the equation*

$$t_{b_j}^j - a_j = R_{j+1}R_{j+2}\cdots R_{j'-1}\left(t_{b_j}^j - a_{j'}\right)$$

The product of R_j's that is output for each equation is the expression describing the probability that the auction will go from offer $j+1$ to offer j' without a buyer accepting an offer.

With the above discussion in mind, we can now present our main equilibrium result.

THEOREM 2. *Suppose we have a multiple-offer TLA $\mathcal{A} = \langle b_1, a_1\rangle, \langle b_2, a_2\rangle, \ldots, \langle b_k, a_k\rangle$ that is optimal in the sense of maximizing the seller's utility, subject to the constraint that only k offers are made. Consider the following strategy for buyer i with valuation v_i facing offer a_j:*

$$s_i^*(v_i, a_j) = \begin{cases} accept & \text{if} \quad v_i > a_j, i \notin \{b_{j+1}, \ldots, b_k\} \\ reject & \text{if} \quad v_i \leq a_j, i \notin \{b_{j+1}, \ldots, b_k\} \\ accept & \text{if} \quad v_i > t_i^j, i \in \{b_{j+1}, \ldots, b_k\} \\ reject & \text{if} \quad v_i \leq t_i^j, i \in \{b_{j+1}, \ldots, b_k\} \end{cases}$$

where the t_i^j values solve the system of equations generated by Algorithm 1. The strategy profile $s^* = (s_1^*, s_2^*, \ldots, s_n^*)$ is a perfect Bayesian equilibrium.

[7]Actually, the number of cases is further increased by the fact that as an offer is added, the number of terms in the existing max operations of that buyer's threshold equations increases by one (a max operations with 2 terms is added in the equation where there was no max operation before).

[8]This assumes that the order in which buyers receive offers is not fixed. If this order is fixed (for example, due to external constraints), then making a surely rejected offer can be beneficial in order to *in effect* change the order without actually changing it.

PROOF. It is easy to see that the technical Conditions 3a-3c of PBE are satisfied. As described above, the buyers update their beliefs using Bayes rule whenever possible, so Condition 1 is satisfied. What remains to be shown is that Condition 2 is satisfied, i.e., that it is never profitable to deviate from the above strategy (in any of the four scenarios). For the first two scenarios, the proof is analogous to the proof of Proposition 1. For the other two scenarios, in which the buyers face another offer later in the auction, we need to make use of the thresholds. Consider buyer i with valuation v_i, facing offer a_j. Let $a_{j'}$ be the next offer that buyer i faces. Suppose $v_i > t_i^j$, but buyer i rejects. The highest expected utility she can expect to get later in the auction is $R_{j+1} \cdots R_{j'-1}(v_i - a_{j'})$. But, by the construction of the thresholds (and the fact that the thresholds only depend on the next offer), this is less than $v_i - a_j$. So buyer i is better off accepting the offer. The scenario where $v_i \leq t_i^j$ is proven similarly. □

5. OPTIMIZING THE OFFERS

So far we have described how we would expect buyers to behave in a TLA. Now we turn to the problem faced by the seller: how to maximize expected utility? The seller can choose the order in which the offers are made as well as the offer values. Before describing the optimization problem, we formalize the seller's objective.

DEFINITION 5. (Seller's expected utility) *Given an instance $\mathcal{A}$ of a TLA, the seller's expected utility from step j onward is defined recursively as*

$$\pi_j = \begin{cases} (1 - R_j)\, a_j + R_j v_0 & \text{if } j = k \\ (1 - R_j)\, a_j + R_j \pi_{j+1} & \text{if } j < k. \end{cases}$$

The seller's expected utility *is given by π_1, and we denote it by $\pi(\mathcal{A})$.*

We can now formally define the design problem faced by the seller.

DEFINITION 6. (TLA design problem) *Given a limit k_{max} on the number of offers made, the seller's TLA* design problem *is to compute $\mathcal{A}^* = \text{argmax}_{\{\mathcal{A} \mid |\mathcal{A}| \leq k_{max}\}} \pi(\mathcal{A})$.*

Note that in Definition 6 the k_{max} parameter is exogenous. It is a constraint imposed on the optimization problem that comes from some external source, for example a limit on the amount of time or resources the seller is able to commit to the auction.

We now discuss some characteristic properties of optimal TLAs.

PROPOSITION 3. *For any given TLA design problem, an optimal TLA exists in which no buyer receives consecutive offers.*

PROOF. If a buyer receives consecutive offers, she will never accept any of them except possibly the lowest. Thus, the other ones of those offers can be removed from $\mathcal{A}$. □

Offers might not decrease over time in optimal TLAs:

EXAMPLE 1. *Consider the 2-buyer, 3-offer setting where buyer 1's valuation is uniformly distributed on the interval $[0, 1]$ and buyer 2's valuation is uniformly distributed on the interval $[1, 4]$. The only optimal sequence of offers is $\mathcal{A} = \langle 2, 2.125\rangle, \langle 1, 0.5275\rangle, \langle 2, 2.0\rangle$.*

However, decreasing offers can be made to each buyer:

PROPOSITION 4. *For any given TLA design problem, an optimal TLA exists in which each buyer individually receives strictly decreasing offers.*

PROOF. If, in equilibrium, all the rejection probabilities R_j of the other buyers that get offers in between two offers of buyer i are zero, that is equivalent to buyer i receiving consecutive offers, which is unnecessary by Proposition 3. We can therefore restrict attention, without loss in the seller's expected utility, to TLAs where this never occurs.

We now show that a buyer will never accept an offer that is equal to or higher than her previous offer. The main idea is that since the buyer knows the sequence of offers at the start of the auction, she knows she can get more utility by accepting an earlier, lower offer, rather than accepting the later, higher offer. Suppose that in an optimal TLA, buyer i is offered a_j at some point in the auction, and the next offer to buyer i is $a_{j'}$, where $a_j \leq a_{j'}$. Without loss of generality, say buyer i's valuation v_i is at least $a_{j'}$. (If it is less than, then buyer i will clearly never accept the offer.) Then the expected utility from accepting offer a_j is $v_i - a_j$, while the expected utility from rejecting offer a_j and later accepting offer $a_{j'}$ is strictly less than $v_i - a_{j'}$ since the utility must be multiplied by the probability that the auction will continue to offer $a_{j'}$ without another buyer accepting an offer and thus ending the auction. Let r denote the probability that no other buyer accepts in between. Since $a_j \leq a_{j'}$ and $r < 1$ (by the argument in the paragraph above), we have $v_i - a_j > r(v_i - a_{j'})$, so buyer i will never accept offer $a_{j'}$. Thus, the wasted offer a'_j can be removed from the auction and the auction remains optimal. □

6. COMPUTATIONAL RESULTS

In this section we develop the computational methodology for solving for an optimal TLA (which involves as a subproblem solving for how the buyers will behave). Designing a TLA involves finding an order (the b_j values) in which the buyers receive the offers, and determining the value of each offer (the a_j values).

6.1 Multiple-offer TLAs

In computing the optimal multiple-offer TLA, we assume that the order of the offers is fixed (e.g., one can include an outer loop to try all orders) and then optimize the TLA for that order as follows.

We run Algorithm 1 to generate the equations for the thresholds t_i^j. Because the shorthand values R_j are expanded in each of those equations, each equation has on the right hand side the offer levels $a_1, a_2, \ldots, a_k$, the specific functional forms of the prior distributions on the types $F_1, F_2, \ldots, F_n$, and the thresholds t_i^j. To compute the optimal offers, we run a non-linear optimizer with these equations as constraints and the expanded recursion for $\pi(\mathcal{A})$ from Definition 6 as the objective, which is now just a function of the offer levels $a_1, a_2, \ldots, a_k$. The output of the optimizer is the values of the optimal offer levels $a_1^*, a_2^*, \ldots, a_k^*$.

The computational complexity of the above procedure is difficult to analyze. Indeed, non-linear optimization problems as a class have resisted rigorous analysis. However, algorithms for finding local optima behave well in practice.

6.1.1 A tractable case

Although the question of complexity remains open for the general case, we have developed a linear-time algorithm for the special case of two buyers with uniform distributions on the interval $[0, 1]$. This algorithm relies heavily on results obtained recently [5, 6]. The authors examine auctions in which the amount of information a buyer can transmit to the seller is severely limited. One of the main results is that when a buyer is limited to choosing from among k possible messages, the buyer's valuation range is partitioned into k continuous intervals (the intervals are, in general, asymmetric among the bidders). The buyer signals which interval her valuation happens to fall in. It turns out that the thresholds that separate the intervals are the same as the equilibrium thresholds in an optimal TLA. (Note that these are the equilibrium thresholds, not the offer levels.) This is easily seen by considering a TLA where each buyer receives k offers, and thus has k threshold values. The TLA could be converted to a direct mechanism where the buyer simply states the first offer that she is willing to accept. Then the problem of computing the thresholds in a TLA becomes the same as computing the thresholds in a communication-bounded auction. The following theorem from [5] (stated using our threshold notation) motivates Algorithm 2.

THEOREM 5. *([5]) When there are two buyers with valuations drawn uniformly from the interval* $[0, 1]$ *the thresholds in an optimal* $2k$*-offer TLA,* $k \geq 2$*, are:*

$$t_i^j = \begin{cases} t + \frac{(2k-j-1)\cdot(1-t)}{2k-3} & \text{if} \quad j \in \{1, 2, \ldots, 2k-1\} \\ \frac{1}{2} & \text{if} \quad j = 2k \end{cases}$$

where $t = \frac{-2\alpha+\sqrt{1+3\alpha}}{2(1-\alpha)}$ *and* $\alpha = \frac{1}{(2k-3)^2}$*. The first offer in the TLA goes to buyer 1, and the offers alternate after that.*

Algorithm 2 computes the offer levels for a multiple-offer TLA with two buyers having uniform distributions on the interval $[0, 1]$.

ALGORITHM 2.
// Let $k_{max} \geq 4$ *be an even number denoting the total number of offers made.*

1. *(Initialize.)* $k \leftarrow \frac{k_{max}}{2}+1$, $\alpha \leftarrow \frac{1}{(2k-3)^2}$, $t \leftarrow \frac{-2\alpha+\sqrt{1+3\alpha}}{2(1-\alpha)}$, $t_2^{k_{max}} \leftarrow \frac{1}{2}$
2. *(Compute thresholds.) For* j *from* 1 *to* $k_{max} - 1$
   (a) *If* j *is odd, then* $i \leftarrow 1$*, else* $i \leftarrow 2$
   (b) $t_i^j \leftarrow t + \frac{(k_{max}-j-1)(1-t)}{2k-3}$
3. *(Compute last offers.)* $a_{k_{max}} \leftarrow t_2^{k_{max}}$, $R_{k_{max}} \leftarrow \frac{a_{k_{max}}}{t_2^{k_{max}-2}}$, $a_{k_{max}-1} \leftarrow t_1^{k_{max}-1}$, $R_{k_{max}-1} \leftarrow \frac{a_{k_{max}-1}}{t_1^{k_{max}-3}}$
4. *(Compute offers.) For* j *from* $k_{max} - 2$ *down to 1*
   (a) *If* j *is odd, then* $i \leftarrow 1$*, else* $i \leftarrow 2$
   (b) $a_j \leftarrow t_i^j - R_{j+1}\left(t_i^j - a_{j+2}\right)$
   (c) *If* $j > 2$*, then* $R_j \leftarrow \frac{t_i^j}{t_i^{j-2}}$*, else* $R_j \leftarrow t_i^j$

THEOREM 6. *Algorithm 2 computes an optimal TLA for the setting with two buyers having uniform distributions on the interval* $[0, 1]$*.*

PROOF. Immediate from Theorems 2 and 5. □

6.2 Single-offer TLAs

Now we are considering single-offer TLAs, so we have $k_{max} \leq n$ by definition. Since making at least some offer to every buyer cannot hurt the seller's expected utility, we always have an optimal single-offer TLA with $k = k_{max}$. So, we set $k = k_{max}$.

For convenience, we will think of the optimization problem as two separate problems. The first problem is to determine the order in which the buyers receive the offers. The second problem is to compute the actual offer values.

First, consider symmetric settings, i.e., $F_i = F_j$ for all buyers i, j. In this case the order does not matter, so we set them arbitrarily. The offer values can be determined by first making an offer to the last bidder equal to the inverse virtual valuation of the reserve price and computing the expected profit from this offer as the new reserve price. Formally:

ALGORITHM 3.
// Compute offers for single-offer TLA

1. *(Initialize.)* $\pi \leftarrow v_0$
2. *(Compute offers.) For* i *from* k_{max} *down to 1*
   (a) $a_i \leftarrow argmax_a \left(1 - F_i(a)\right) a + F_i(a)\pi$
   (b) $\pi \leftarrow (1 - F_i(a_i))a_i + F_i(a_i)\pi$

PROPOSITION 7. *Algorithm 3 computes an optimal single-offer TLA in the symmetric settings.*

PROOF. Consider the last offer made in a TLA. The result of the last offer does not affect the result of any of the other offers. Thus, the seller should make an optimal offer, which is computed in step 2(a) during the first iteration. Now consider the next to last offer. Again, the result of this offer does not affect any of the previous offers. Also, the expected utility the seller earns if this offer is rejected has already been computed and is stored in π. Thus, the offer computed in step 2(a) is the optimal offer. Continuing with this backward induction, it is easy to see that Algorithm 3 correctly computes the optimal offer levels. □

If we assume that F_i can be evaluated in constant time and that the "argmax" computation can also be done in constant time, then Algorithm 3 runs in time linear in the number of offers k_{max}. For a wide range of common distributions, including uniform and exponential distributions, these assumptions hold.

Now, consider asymmetric settings. Clearly the order of offers now can make a difference to the seller's expected utility. For general asymmetric preferences, we do not know of an efficient algorithm for ordering the offers. (For example, always greedily making an offer to the remaining buyer from whom the seller can myopically expect the greatest utility is not generally optimal.) One could try all possible orderings, running Algorithm 3 to evaluate each ordering. However, for a large number of buyers this would be infeasible since the number of orderings is $k_{max}!$. Fortunately, we have been able to show that in some commonly used special cases one can order the buyers in polynomial time. Theorem 8 illustrates just one of many results. We denote a uniform distribution over the interval $[a, b]$ as $\mathrm{U}[a, b]$.

THEOREM 8. *With* $v_i \sim \mathrm{U}[0, w_i]$*, an optimal single-offer TLA can be determined in* $O(n \log n)$ *time by sorting the buyers in decreasing order of* w_i *and passing the sequence to Algorithm 3.*

PROOF. Omitted due to space. □

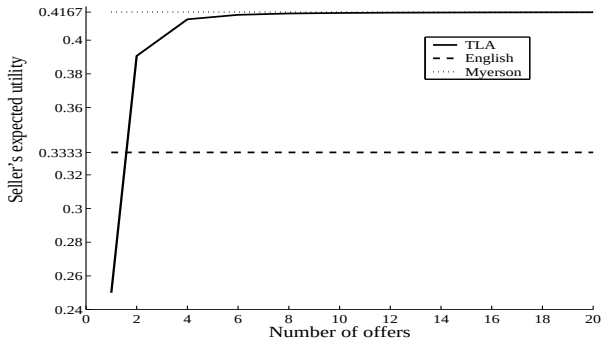


Figure 1: Seller's expected utility as the number of offers increases. The two buyers' valuations have uniform distributions on the interval $[0, 1]$.

7. ECONOMIC PERFORMANCE OF TLAS

In this section we present experiments and theoretical results to give an indication of how one can expect our TLAs, optimized as explained above, to perform in practice. (Throughout these experiments, we set the seller's valuation v_0 to zero.)

First we studied the performance of our optimal single-offer TLA as the number of buyers increases. Naturally, as the number of buyers increases, the expected utility of the single-offer TLA increases. Table 1 shows how well an optimal single-offer TLA fares relative to Myerson's utility-maximizing auction as the number of buyers increases (and each buyer is made exactly one offer).[9] The single-offer TLA is close to optimal. Interestingly, the relative performance does not improve monotonically, but does improve when the number of buyers is large.[10] The advantage of our single-offer TLA over the English auction is greatest when the number of buyers is not very large. Also, it tends to be greatest in the asymmetric settings. (Naturally, our multiple-offer TLAs would yield even greater expected utility for the seller.) In the case of 1 buyer, a TLA performs exactly as the Myerson auction, and the English auction does not generate any expected utility for the seller.

Figure 1 shows how the seller's expected utility in an optimal multiple-offer TLA increases as the number of offers increases. As can be seen, the TLA performs extremely well after a small number of offers. We also studied how well the seller's expected utility in an optimal multiple-offer TLA increases as the number of offers increases. The following result, which follows immediately from a result in [5] and the correspondence between communication-bounded auctions and TLAs, shows that we can expect a TLA to perform well for a wide variety of distributions.

THEOREM 9. *([6]) The revenue loss in an optimal k-offer TLA with symmetric buyers having regular distributions*[11] *is* $O\left(\frac{1}{k^2}\right)$.

For some distributions, including the uniform distribution, the above bound is tight. This result is asymptotic, but in practice TLAs yield close to optimal expected utility for the seller, as exemplified in Table 2 where we considered several 2-buyer settings, showing how close a TLA gets to the expected utility generated by Myerson's utility-maximizing auction as the number of offers increases. Even with a small number of offers, the expected utility of a TLA is close to optimal, and increases monotonically with the number of offers.

Total number of offers:	1	2	3	English
$v_1 \sim$ U$[0,1]$, $v_2 \sim$ U$[0,1]$	.600	.938	.979	.800
$v_1 \sim$ U$[0.5,2]$, $v_2 \sim$ U$[0,1]$	.857	.977	.992	.625
$v_1 \sim$ U$[1,4]$, $v_2 \sim$ U$[0,1]$	.932	.994	.997	.350
$v_1 \sim$ Exp$[2]$, $v_2 \sim$ Exp$[2]$	.551	.932	.977	.749
$v_1 \sim$ Exp$[2]$, $v_2 \sim$ Exp$[4]$	.726	.968	.986	.658

Table 2: Seller's expected utility of an optimal 2-buyer TLA and English auction divided by the expected utility of the Myerson auction. The number of offers for the TLA is varied from 1 to 3. In each case, it is optimal to give the first offer to buyer 1

In principle, it is possible to achieve optimal expected revenue using an ascending auction, albeit one with asymmetric prices. We call such an auction an *asymmetric ascending auction*. (An asymmetric descending auction can be constructed similarly.) In an ascending auction, there is a single price, visible by all bidders, that rises. Bidders drop out when they no longer would like to buy the item at the current price. As discussed previously, this auction does not generate optimal expected revenue for the seller and in fact can be arbitrarily far from optimal. Alternatively, in an asymmetric ascending auction, there is a single price (the *virtual price*) that rises. But instead of the auctioneer communicating this price to the bidders, she insteads communicates asymmetric prices to each bidder, using the inverse of the virtual valuation function evaluated at the current virtual price to compute prices for each bidder. It is straightforward to demonstrate that this method generates optimal expected revenue for the seller.[12]

One advantange of TLAs over asymmetric ascending auctions is that much less communication is required. This is especially important in auctions in which there is some cost to communication, for example when the bidders and the auctioneer communicate via a network. (In general, there is no upper bound on the amount of communication required for an ascending auction.) In fact, a TLA is, in some sense, an approximation of an asymmetric descending auction when there is some limit on the communication resources available to the bidders.

Finally, we mention that a single-offer TLA implements the Myerson auction in the case of a single buyer.

8. MECHANISMS SIMILAR TO TLAS

We are aware of one fielded mechanism that is similar to a TLA. Amazon.com has recently introduced the "Gold Box" on their web site [1]. Each day, registered users are given up to ten take-it-or-leave-it offers for ten different goods. The offers expire after one hour, and after an offer is rejected, it is no longer available to the user. The offer levels appear to

[9] We denote the exponential distribution having mean $1/\mu$ as Exp$[\mu]$ (its cumulative density is $F(x) = 1 - e^{-\mu x}$).

[10] In the third row, the optimal single-offer TLA deterministically allocates the good to the highest buyer, thus making the allocation *efficient* in this case, where the Myerson auction is inefficient.

[11] Recall that a distribution F_i is said to be regular if $f_i/(1 - F_i)$ is increasing.

[12] As in the proof of the optimality of the Myerson auction, the bidders' distributions must be regular for the result to hold.

Num buyers:	1	2	4	6	8	10	20	100
$v_i \sim \mathrm{U}[0,1]$	1 / 0	.938 / .800	.981 / .978	.894 / .997	.900 / 1	.907 / 1	.937 / 1	.992 / 1
$v_i \sim \mathrm{U}[0,i]$	1 / 0	.980 / .646	.959 / .844	.953 / .898	.952 / .922	.952 / .936	.958 / .963	.979 / .987
$v_i \sim \mathrm{U}[i,i+1]$	1 / 0	.979 / .735	.990 / .866	.993 / .910	.995 / .933	.996 / .946	.998 / .973	1 / .995
$v_i \sim \mathrm{Exp}[2]$	1 / 0	.932 / .749	.994 / .961	.847 / .991	.837 / .997	.833 / .999	.836 / 1	.872 / 1
$v_i \sim \mathrm{Exp}[i]$	1 / 0	.509 / .657	.917 / .813	.900 / .824	.918 / .822	.927 / .819	.942 / .813	.948 / .810

Table 1: The left (right) number in each cell is the ratio of the expected utility of an optimal single-offer TLA (English auction) to the expected utility of the Myerson auction.

be below the normal retail price, but we do not know what pricing strategy is in use. The techniques developed in this paper could be used to compute optimal values for these prices so as to maximize Amazon's expected utility (under certain reasonable assumptions).

A piece of closely related research concerns online auctions [11] and online exchanges [4]. In those papers the authors examine how bids should be treated in an online setting where the auctioneer (or exchange administrator) has to make accept/reject decisions about expiring bids before all potential future bids are received. The authors have developed online algorithms and analyzed how well their algorithms do compared to an omniscient algorithm that knows all future bids in advance. Many of those algorithms end up choosing bid acceptance thresholds, so in that sense they use (implicit) take-it-or-leave-it offers.

This setting differs from ours in that there is not only uncertainty about future decisions that will be made, but there is uncertainty about the types of decisions that will be faced. In our setting, the sequence of offers is known in advance by all parties. In addition, in the online model there are multiple units of the item available for exchange, while we have focused only on the single unit case.

Finally, most retail is of the form of making take-it-or-leave-it offers. In the typical retail setting the seller posts price for each item she wishes to sell. The seller generally does not have specific information about a given buyer, and so all buyers face the same prices (i.e. the prices are *non-discriminatory*). This distinguishes the typical retail setting from single-offer TLAs where the prices are *discriminatory*.

9. CONCLUSIONS AND FUTURE WORK

We introduced TLAs as a selling (or buying) mechanism that has low valuation revelation as do the most commonly used auction mechanisms (English and Dutch auctions), yet, unlike them, generates close-to-optimal expected utility for the seller. We showed that in single-offer TLAs, each buyer's dominant strategy is to act truthfully. In multiple-offer TLAs, the buyers' optimal strategies are more intricate, and we derived the perfect Bayesian equilibrium for the game. We developed algorithms for finding the equilibrium and for optimizing the offers in single-offer and multiple-offer TLAs so as to maximize the seller's expected utility (subject to an exogenous constraint on the number of offers), and proved that optimal TLAs have many desirable features. We applied a result due to Blumrosen and Nisan to TLAs to show that the seller's expected utility is close to optimal already for a small number of offers. We also showed in several examples that as the number of buyers increases, the seller's expected utility increases and becomes increasingly (but not monotonically) more competitive with Myerson's expected utility maximizing auction. Myerson's uses full valuation revelation and is arguably impractical because the rules are unintuitive, unlike ours in which the mechanism is simply a sequence of take-it-or-leave-it offers. Even our single-offer TLA tends to yield significantly greater utility to the seller than the English auction when the setting is asymmetric or the number of buyers is not very large.

Future research includes developing fast offer-ordering algorithms for asymmetric single-offer TLAs for arbitrary valuation distributions, as well as fast special-purpose algorithms for PBE finding, offer ordering, and offer value optimization in multiple-offer TLAs. We also plan to extend this work to auctions of multiple identical units and to auctions of multiple distinguishable items.

10. REFERENCES

[1] Amazon.com. Gold box offers. *http://www.amazon.com/gold-box*, 2003.

[2] S. Athey. Single crossing properties and the existence of pure strategy equilibria in games of incomplete information. *Econometrica*, 69(4):861–889, July 2001.

[3] D. Bergemann and M. Pesendorfer. Information structures in optimal auctions. Discussion Paper 1323, Yale U., 2001.

[4] A. Blum, T. Sandholm, and M. Zinkevich. Online algorithms for market clearing. In *SODA-02*, 2002.

[5] L. Blumrosen and N. Nisan. Auctions with severely bounded communication. In *FOCS-02*, 2002.

[6] L. Blumrosen, N. Nisan, and I. Segal. Multi-player and multi-round auctions with severely bounded communication. In *ESA*, pages 102–113, 2003.

[7] V. Conitzer and T. Sandholm. Complexity of mechanism design. In *UAI-02*, pp. 103–110, 2002.

[8] D. Fudenberg and J. Tirole. Perfect Bayesian equilibrium and sequential equilibrium. *J. of Economic Theory*, 53:236–260, 1991.

[9] V. Krishna. *Auction Theory*. Academic Press, 2002.

[10] K. Larson and T. Sandholm. Costly valuation computation in auctions. In *TARK VIII*, 2001.

[11] R. Lavi and N. Nisan. Competitive analysis of incentive compatible on-line auctions. In *ACM-EC*, 2000.

[12] P. Milgrom and R. Weber. Distributional strategies for games with incomplete infromation. *Mathematics of Operations Research*, 10:619–632, 1985.

[13] R. Myerson. Optimal auction design. *Mathematics of Operation Research*, 6:58–73, 1981.

[14] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *ACM-EC*, 1999.

[15] D. Parkes. Optimal auction design for agents with hard valuation problems. In *Agent-Mediated Electronic Commerce Workshop at IJCAI-99*, Stockholm, 1999.

[16] A. Ronen and A. Saberi. Optimal auctions are hard. In *FOCS*, pages 396–405, 2002.

[17] M. Rothkopf, T. Teisberg, and E. Kahn. Why are Vickrey auctions rare? *J. of Political Economy*, 98(1):94–109, 1990.

[18] T. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *AAAI*, pages 256–262, Washington, D.C., July 1993.

[19] T. Sandholm. Issues in computational Vickrey auctions. *International Journal of Electronic Commerce*, 4(3):107–129, 2000. Special Issue on Applying Intelligent Agents for Electronic Commerce.

[20] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16, 1961.

False-name-proof Combinatorial Auction Protocol: Groves Mechanism with Submodular Approximation

Makoto Yokoo, Toshihiro Matsutani, and Atsushi Iwasaki
Kyushu University
6-10-1 Hakozaki, Higashi-ku,
Fukuoka, 812-8581 Japan
{yokoo, iwasaki}@is.kyushu-u.ac.jp, matutani@lang.is.kyushu-u.ac.jp
lang.is.kyushu-u.ac.jp/~{yokoo, iwasaki}

ABSTRACT

This paper develops a new combinatorial auction protocol called the Groves Mechanism with SubModular Approximation (GM-SMA). This protocol satisfies the following characteristics: (1) it is false-name-proof, (2) each winner is included in a Pareto efficient allocation, and (3) as long as a Pareto efficient allocation is achieved, the protocol is robust against the collusion of losers and the outcome is in the core. As far as the authors are aware, the GM-SMA is the first protocol that satisfies all three of these characteristics.

The basic ideas of the GM-SMA are as follows: (i) It is based on the VCG protocol, i.e., the payment of a winner in this protocol is identical to the payment in one instance of the Groves mechanism, which is a class of protocols that includes the VCG. (ii) When calculating the payment of a bidder, we approximate the valuations of other bidders by using a submodular valuation function (submodular approximation). Simulation results show that the GM-SMA achieves a better social surplus and seller's revenue than existing false-name-proof protocols, as long as the submodular approximation is close enough to the original valuations.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multi-agent systems*

General Terms

Theory, Economics

Keywords

Combinatorial Auction, Strategy-Proof

1. INTRODUCTION

Internet auctions have become an integral part of Electronic Commerce and a promising field for applying AI and agent technologies. Among various studies related to Internet auctions, those on combinatorial auctions have lately attracted considerable attention (an extensive survey is presented in [3]). Although conventional auctions sell a single item at a time, combinatorial auctions sell multiple items with interdependent values simultaneously and allow the bidders to bid on any combination of items. In a combinatorial auction, a bidder can express complementary or substitutable preferences over multiple bids. By taking into account complementary/substitutable preferences, we can increase the participants' utilities and seller's revenue.

One desirable characteristic of an auction protocol is that it is *strategy-proof*. A protocol is strategy-proof if, for each bidder, declaring his/her true valuation is a *dominant strategy*, i.e., an optimal strategy regardless of the actions of other bidders. In theory, the revelation principle states that in the design of an auction protocol, we can restrict our attention to strategy-proof protocols without loss of generality [8]. In other words, if a certain property (e.g., Pareto efficiency) can be achieved using some auction protocol in a dominant-strategy equilibrium, i.e., a combination of dominant strategies of bidders, then the property can also be achieved using a strategy-proof auction protocol.

Furthermore, a strategy-proof protocol is also practical for Internet auctions. For example, if we use the first-price sealed-bid auction, which is not strategy-proof, the bidding prices must be securely concealed until the auction is closed. On the other hand, if we use a strategy-proof protocol, knowing the bidding prices of other bidders is useless; consequently, such security issues become less critical.

The Vickrey-Clarke-Groves (VCG) protocol [2, 4, 9] is a strategy-proof protocol that can be applied to combinatorial auctions. The VCG protocol satisfies Pareto efficiency. Under several natural assumptions, the VCG protocol is the only strategy-proof, Pareto efficient protocol [6].

However, the VCG protocol has several limitations, including the problems described below.

Vulnerability to false-name bids: The authors have pointed out the possibility of a new type of fraud called false-name bids, which utilizes the anonymity available on the Internet [11, 12]. False-name bids are bids submitted under fictitious names, e.g., multiple e-mail addresses. Such a dishonest action is very difficult to detect, since identifying each participant on the Internet is virtually impossible.

We say a protocol is *false-name-proof* if, for each bidder,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

declaring his/her true valuations using a single identifier (although the bidder can use multiple identifiers) is a dominant strategy. The VCG is not false-name-proof [11, 12].

Vulnerability to loser collusion: As shown in [1], in the VCG protocol, there is a chance that some bidders, who would be losers if they bid their true valuations, become winners and increase their utility if they collude and adjust their bids.

The outcome is not in the core: As shown in [1], in the VCG protocol, there is a chance that the result of the auction is not in the core. This means that a seller has an incentive to deviate from the protocol and to sell goods to the bidders who are not winners in the VCG protocol.

In this paper, we introduce a new combinatorial auction protocol called the *Groves Mechanism with SubModular Approximation* (GM-SMA). The basic ideas of this protocol are as follows.

- This protocol is based on the VCG protocol. More precisely, the payment of a winner in this protocol is identical to the payment in one instance of the Groves mechanism [4], which is a class of protocols that includes the VCG[1].
- In this protocol, when calculating the payment of bidder i, we approximate the valuations of bidders except for i, by using a *submodular* valuation function (submodular approximation).

The GM-SMA satisfies the following characteristics.

(1) The GM-SMA is false-name-proof.

(2) In the GM-SMA, each winner is a bidder who is included in a Pareto efficient allocation[2].

(3) In the GM-SMA, as long as the allocation is Pareto efficient, the protocol is robust against the collusion of the losers and the outcome is in the core.

As far as the authors are aware, the GM-SMA is the first protocol that satisfies all three of these characteristics.

The VCG protocol can always achieve a Pareto efficient allocation as long as there exists no false-name bid. However, it is not false-name-proof and cannot guarantee to achieve a Pareto efficient allocation when false-name bids are possible. In addition, even without false-name bids, the VCG protocol does not satisfy characteristic (3).

Furthermore, in existing false-name-proof protocols [10, 11], characteristic (2) is not satisfied, i.e., a bidder who is not included in any Pareto efficient allocation becomes a winner quite often. In such a case, the outcome is not in the core. Also, losers who are in a Pareto efficient allocation will have a strong incentive to collude and act as a single bidder.

In [1], an ascending protocol that utilizes a proxy is presented. In this protocol, a bidder cannot increase his/her utility by using multiple identifiers. Also, as long as each bidder bids straightforwardly, a Pareto efficient allocation is achieved and characteristic (3) is satisfied. However, unless the valuations of bidders are submodular, bidding straightforwardly is not a dominant strategy, i.e., a bidder may have an incentive to misreport his/her valuation.

The rest of this paper is organized as follows. First, we describe the model of combinatorial auctions (Section 2). Next, we describe a general framework for describing strategy-proof and false-name-proof protocols called the Price-Oriented, Rationing-Free (PORF) protocol [10] (Section 3). By describing a protocol as a PORF protocol, proving the protocol is strategy/false-name proof becomes much easier. Then, we describe the VCG protocol and its limitations (Section 4) as well as existing false-name-proof protocols and their limitations (Section 5). Furthermore, we describe the details of the GM-SMA (Section 6) and show its characteristics (Section 7). Finally, we evaluate the social surplus and the revenue of the seller in the GM-SMA through a simulation (Section 8).

2. MODEL

Assume there are a set of bidders $N = \{1, 2, \ldots, n\}$ and a set of goods $M = \{1, 2, \ldots, m\}$. Each bidder i has his/her preferences over $B \subseteq M$. Formally, we model this by supposing that bidder i privately observes a parameter, or signal, θ_i, which determines his/her preferences. We refer to θ_i as the *type* of bidder i. We assume θ_i is drawn from a set Θ. We also assume a *quasi-linear, private value* model with *no allocative externality*, defined as follows.

DEFINITION 1. *utility of a bidder*
The utility of bidder i, when i obtains a bundle, i.e., a subset of goods $B \subseteq M$ and pays $p_{B,i}$, is represented as $v(B, \theta_i) - p_{B,i}$.

We assume the valuation v is normalized by $v(\emptyset, \theta_i) = 0$. Furthermore, we assume *free disposal*, i.e., $v(B', \theta_i) \geq v(B, \theta_i)$ for all $B' \supseteq B$.

In a traditional definition [7], an auction protocol is (dominant strategy) *incentive compatible* (or *strategy-proof*) if declaring the true type/valuation is a dominant strategy for each bidder, i.e., an optimal strategy regardless of the actions of other bidders.

In this paper, we extend the traditional definition of incentive compatibility so that it can address false-name bid manipulations, i.e., we define that an auction protocol is (dominant strategy) incentive compatible if declaring the true type *by using a single identifier* is a dominant strategy for each bidder. To distinguish between the traditional and extended definitions of incentive compatibility, we refer to the traditional definition as strategy-proof and to the extended definition as *false-name-proof*.

As in the case of strategy-proof protocols, the revelation principle holds for false-name-proof protocols [12]. Thus, we can restrict our attention to false-name-proof protocols without loss of generality.

We say an auction protocol is *Pareto efficient* when the sum of all participants' utilities (including that of the auctioneer), i.e., the social surplus, is maximized in a dominant-strategy equilibrium. We have proved that there exists no false-name-proof protocol that satisfies Pareto efficiency [12]. Therefore, we need to sacrifice efficiency to some extent when false-name bids are possible.

[1]Strictly speaking, the GM-SMA is not an instance of the Groves mechanism, since it is not guaranteed to achieve a Pareto efficient allocation.

[2]Note that the converse is not true, i.e., a bidder who is included in a Pareto efficient allocation is not necessarily a winner. Thus, there is a chance that the GM-SMA could fail to achieve a Pareto efficient allocation. This is inevitable since the GM-SMA is false-name-proof [12].

3. PRICE-ORIENTED, RATIONING-FREE (PORF) PROTOCOL

In this section, we describe a general framework for describing strategy-proof protocols called Price-Oriented, Rationing-Free (PORF) protocol [10]. By describing a protocol as a PORF protocol, proving that the protocol is strategy-proof or false-name-proof becomes much easier.

A PORF Protocol is defined as follows.

DEFINITION 2. *PORF Protocol*

- *Each bidder i declares his/her type $\tilde{\theta}_i$, which is not necessarily the true type θ_i.*
- *For each bidder i and for each bundle $B \subseteq M$, the price $p_{B,i}$ is defined. This price must be determined independently of i's declared type $\tilde{\theta}_i$, while it can be dependent on declared types of other bidders.*
- *We assume $p_{\emptyset,i} = 0$ holds. Also, if $B \subseteq B'$, $p_{B,i} \leq p_{B',i}$ holds.*
- *For bidder i, a bundle B^* is allocated, where $B^* = \arg\max_{B \subseteq M} v(B, \tilde{\theta}_i) - p_{B,i}$. Bidder i pays $p_{B^*,i}$. If there exist multiple bundles that maximize i's utility, one of these bundles is allocated.*
- *The result of the allocation satisfies* allocation feasibility*, i.e., for two bidders i, j and bundles allocated to these bidders B_i^* and B_j^*, $B_i^* \cap B_j^* = \emptyset$ holds.*

It is straightforward to show that a PORF protocol is strategy-proof. The price of bidder i is determined independently of i's declared type, and he/she can obtain the bundle that maximizes his/her utility independently of the allocations of other bidders, i.e., the protocol is rationing-free.

On the other hand, in a PORF protocol, the prices must be determined appropriately to satisfy allocation feasibility. The definition of a PORF protocol requires that if there exist multiple bundles that maximize i's utility, then one of these bundles must be allocated, but it does not specify exactly which bundle should be allocated. Therefore, if there exist multiple choices, the auctioneer can adjust the allocation of multiple bidders in order to satisfy allocation feasibility.

In [10], it is shown that any strategy-proof protocol can be described as a PORF protocol, i.e., the following theorem holds.

THEOREM 1. *If a protocol is strategy-proof, then the protocol can be described as a PORF protocol.*

In this paper, we assume that a protocol satisfies the following condition.

DEFINITION 3. *weakly-anonymous pricing rule (WAP) For bidder i, the price of bundle B is given as a function of the types of other bidders, i.e., the price can be described as $p(B, \Theta_{N'})$, where N' is the set of bidders other than i, and $\Theta_{N'}$ is the set of types of bidders in N'.*

The above condition requires that if two bidders face the same types of opponents, their prices must be identical for all bundles. The WAP condition is intuitively natural and virtually all well-known protocols, including the VCG, satisfy this condition.

For a PORF protocol that satisfies the WAP condition, we define the following additional condition.

DEFINITION 4. *no super-additive price increase (NSA) For any subset of bidders $S \subseteq N$ and $N' = N \setminus S$, and for $i \in S$, let us denote B_i as a bundle that maximizes i's utility, then $\sum_{i \in S} p(B_i, \bigcup_{j \in S \setminus \{i\}} \{\theta_j\} \cup \Theta_{N'}) \geq p(\bigcup_{i \in S} B_i, \Theta_{N'})$.*

An intuitive description of this condition is that the price of buying a combination of bundles (the right side of the inequality) must be smaller than or equal to the sum of the prices for buying these bundles separately (the left side).

In [10], it is shown that for a PORF protocol with the WAP, the NSA is a sufficient condition for a protocol to be false-name-proof, i.e., the following theorem holds.

THEOREM 2. *If a PORF protocol with the WAP satisfies the NSA condition, then the protocol is false-name-proof.*

4. VCG PROTOCOL

Next, we show how the VCG protocol can be described as a PORF protocol. Since a PORF protocol is strategy-proof, in the rest of this paper, we assume each bidder i declares his/her true type θ_i.

To simplify the protocol description, we introduce the following notation. For a set of goods B and a set of bidders S, where Θ_S is a set of types of bidders in S, we define $V^*(B, \Theta_S)$ as the sum of the valuations of S when B is allocated optimally among S.

To be precise, for a feasible allocation $g = (B_1, B_2, \ldots)$, where $\bigcup_{i \in S} B_i \subseteq B$ and for all $i \neq i', B_i \cap B_{i'} = \emptyset$, $V^*(B, \Theta_S)$ is defined as $\max_g \sum_{i \in S} v(B_i, \theta_i)$, where θ_i is the type of bidder i.

The price of bundle B for bidder i (VCG price) is defined as follows:

$$p_{B,i} = V^*(M, \Theta_{N \setminus \{i\}}) - V^*(M \setminus B, \Theta_{N \setminus \{i\}}).$$

This protocol is identical to the Vickrey-Clarke-Groves (VCG) mechanism [2, 4, 9], i.e., if B is allocated to i in a Pareto efficient allocation, then $p_{B,i}$ is equal to the payment in the VCG protocol and B maximizes i's utility.

This protocol satisfies allocation feasibility since the following theorem holds.

THEOREM 3. *If B is allocated to i in a Pareto efficient allocation, then B maximizes i's utility at the VCG price.*

Proof: If we assume this theorem does not hold, then there exists another bundle $B' \neq B$ that maximizes i's utility. Thus, the following formula holds.

$$v(B', \theta_i) - p_{B',i} > v(B, \theta_i) - p_{B,i}$$

From this formula and the definition of $p_{B',i}, p_{B,i}$, we obtain:

$$v(B', \theta_i) + V^*(M \setminus B', \Theta_{N \setminus \{i\}}) > v(B, \theta_i) + V^*(M \setminus B, \Theta_{N \setminus \{i\}}).$$

This contradicts the assumption that allocating B to i is Pareto efficient. □

Note that this theorem also holds when B is an empty set. From this theorem, if there exist multiple Pareto efficient allocations, the auctioneer can choose one of them arbitrarily, since every bidder has the highest preference for the bundle that is allocated in that allocation (which can be an empty bundle). Thus, allocation feasibility is satisfied.

Let us describe how this protocol works.

EXAMPLE 1. *Assume there are two goods* 1 *and* 2*, and three bidders, bidder* 1*,* 2*, and* 3*, whose types are θ_1, θ_2, and θ_3, respectively. The valuation for a bundle $v(B, \theta_i)$ is determined as follows.*

	{1}	{2}	{1, 2}
θ_1	6	0	6
θ_2	0	0	8
θ_3	0	5	5

Accordingly, the prices of these bundles for each bidder are given as follows.

	{1}	{2}	{1, 2}
bidder 1	3	8	8
bidder 2	6	5	11
bidder 3	8	2	8

As a result, bidder 1 *obtains good* 1 *at price* 3*, and bidder* 3 *obtains good* 2 *at price* 2*.*

Now, we can see that this result is not in the core. Let us assume another allocation where both goods are sold to bidder 2 at price 7. Then, both the seller and bidder 2 prefer this allocation to the outcome of the VCG protocol. Such a coalition, in this case, the coalition of the seller and bidder 2, is called a *blocking* coalition.

Let us consider another situation.

EXAMPLE 2. *Assume there are two goods* 1 *and* 2*, and three bidders, bidder* 1*,* 2*, and* 3*, whose types are* θ_1, θ_2*, and* θ_3*, respectively. The valuation for a bundle* $v(B, \theta_i)$ *is determined as follows.*

	{1}	{2}	{1, 2}
θ_1	4	0	4
θ_2	0	0	8
θ_3	0	3	3

Accordingly, the prices of these bundles for each bidder are given as follows.

	{1}	{2}	{1, 2}
bidder 1	5	8	8
bidder 2	4	3	7
bidder 3	8	4	8

As a result, bidder 2 *obtains both goods at price* 7*.*

Now, we can see that the VCG protocol is vulnerable to the collusion of losers. If bidder 1 and bidder 3 collude and increase their valuations, they can create a situation identical to Example 1. Then, bidder 1 obtains good 1 at price 3, and bidder 3 obtains good 2 at price 2. For each bidder, this price is less than his/her valuation, thus bidders can increase their utilities by engaging in collusion.

5. EXISTING FALSE-NAME-PROOF PROTOCOLS

In this section, we describe a false-name-proof protocol called the Max Minimal-Bundle (M-MB) protocol [10] as a representative false-name-proof protocol.

To simplify the protocol description, we introduce a concept called a *minimal bundle*.

DEFINITION 5. *minimal bundle*
Bundle B *is called minimal for bidder* i *if for all* $B' \subset B$ *and* $B' \neq B$*,* $v(B', \theta_i) < v(B, \theta_i)$ *holds.*

In the M-MB protocol, the price of bundle B for bidder i is defined as follows:

- $p_{B,i} = \max_{B_j \subseteq M, j \neq i} v(B_j, \theta_j)$, where $B \cap B_j \neq \emptyset$ and B_j is minimal for bidder j.

In short, the price of bundle B is equal to the highest valuation of a bundle, which is minimal and conflicting with bundle B.

It is clear that this protocol satisfies allocation feasibility [10]. Also, as shown in [10], this protocol is false-name-proof since it satisfies the NSA condition.

Let us describe how this protocol works. Let us consider a situation identical to Example 1. The prices of these bundles are given as follows.

	{1}	{2}	{1, 2}
bidder 1	8	8	8
bidder 2	6	5	6
bidder 3	8	8	8

As a result, bidder 2 obtains both goods at price 6.

As shown in this example, in the M-MB protocol, some goods can be allocated to a bidder who is not included in any Pareto efficient allocation. This is true for other existing false-name-proof protocols such as the LDS protocol [11]. In such a case, losers who are in a Pareto efficient allocation will have a strong incentive to collude. For example, if bidder 1 and bidder 3 collude and act as a single bidder whose valuation for the set is 11, then they can obtain both goods at price 8, which is smaller than the sum of their valuations.

6. GROVES MECHANISM WITH SUBMODULAR APPROXIMATION (GM-SMA)

In this section, we describe the details of the Groves Mechanism with SubModular Approximation (GM-SMA).

6.1 Protocol Description

To define the prices used in the GM-SMA, we introduce a concept called submodular approximation.

DEFINITION 6. *submodular approximation* U^*
The function $U^*(B, \Theta_S)$ *defined below is called a submodular approximation of* V^**.*

For a feasible allocation $g = (B_1, B_2, \ldots)$*, where* $\bigcup_{i \in S} B_i \subseteq B$ *and for all* $i \neq i', B_i \cap B_{i'} = \emptyset$*,* $U^*(B, \Theta_S)$ *is defined as* $\max_g \sum_{i \in S} v'(B_i, \theta_i)$*.* v' *is a function that satisfies* $v'(B, \theta_i) \geq v(B, \theta_i)$ *for all* i, B*.*

Furthermore, we assume v' *is chosen so that* U^* *becomes submodular, i.e., for all* $S \subseteq N, B', B'' \in M$*, the following condition holds:*

$$U^*(B', \Theta_S) + U^*(B'', \Theta_S) \geq U^*(B' \cup B'', \Theta_S) + U^*(B' \cap B'', \Theta_S).$$

From this definition, U^* satisfies the following conditions:

- For all $S \subset N, B \subset M$, $U^*(B, \Theta_S) \geq V^*(B, \Theta_S)$ holds.
- For all $S \subset N, B \subset M$, $U^*(M, \Theta_N) \geq U^*(B, \Theta_S) + U^*(M \setminus B, \Theta_{N \setminus S})$ holds.

The Groves Mechanism with SubModular Approximation (GM-SMA) is described as a PORF protocol in which prices are defined as follows.

- When $B = \emptyset$, $p_{B,i} = 0$.
- Otherwise, $p_{B,i} = U^*(M, \Theta_{N\setminus\{i\}}) - V^*(M\setminus B, \Theta_{N\setminus\{i\}})$.

Note that we use U^* only for the first term.

The PORF description of a protocol is very convenient for proving the theoretical properties of the protocol, but implementing it can be rather costly. Alternatively, the GM-SMA can be implemented as follows.

1. The auctioneer finds a Pareto efficient allocation. The auctioneer can choose an arbitrary one if there exist multiple Pareto efficient allocations.
2. When B is allocated to bidder i in that allocation, the auctioneer calculates the GM-SMA price $p_{B,i}$. If $v(B, \theta_i) \geq p_{B,i}$, then B is allocated to bidder i at price $p_{B,i}$. If $v(B, \theta_i) < p_{B,i}$, B is not allocated to any bidder.

This protocol is quite similar to the Groves mechanism [4], since the payment of a winner in this protocol is identical to the payment in one instance of the Groves mechanism. However, it is not an instance of the Groves mechanism because it does not guarantee to achieve a Pareto efficient allocation. This protocol can be considered as a modification of the Groves mechanism, in which a winner has an option not to obtain the goods if the payment becomes too high.

6.2 Examples

Here, we describe how the GM-SMA works in the situation of Example 1. In this example, we use $v'(\{1\}, \theta_2) = v'(\{2\}, \theta_2) = 8/2 = 4$, and we use $v'(\cdot, \cdot) = v(\cdot, \cdot)$ for other bundles and bidders. By choosing v' in this way, U^* satisfies the submodular condition.

The GM-SMA prices are given as follows.

	$\{1\}$	$\{2\}$	$\{1,2\}$
bidder 1	4	9	9
bidder 2	6	5	11
bidder 3	10	4	10

As a result, bidder 1 obtains good 1 at price 4, and bidder 3 obtains good 2 at price 4.

This outcome is in the core. Since the revenue of the seller is 8, the seller does not have an incentive to deviate from the protocol with bidder 2 (who is willing to pay at most 8).

Next, we describe how the protocol works in the situation of Example 2. Here, v' is defined in a similar way to the previous example. The GM-SMA prices are given as follows.

	$\{1\}$	$\{2\}$	$\{1,2\}$
bidder 1	5	8	8
bidder 2	4	3	7
bidder 3	8	4	8

As a result, bidder 2 obtains both goods at price 7. In this case, even if bidder 1 and bidder 3 collude and create a situation identical to Example 1, their utilities cannot be positive since each must pay 4 in that situation. In this case, the GM-SMA gives the same result as the VCG.

6.3 Submodular Approximation

In this subsection, we discuss how to choose v' so that U^* becomes submodular. It is known that if each bidder's valuation satisfies a condition called the *gross substitutes* condition[3], then U^* becomes submodular [5].

DEFINITION 7. *gross substitutes*
Given a price vector $p = (p_1, \ldots, p_m)$, we denote

$$D_i(p) = \{B \subset M : \forall C \subset M, v(B, \theta_i) - \sum_{j \in B} p_j \geq v(C, \theta_i) - \sum_{j \in C} p_j\}.$$

$D_i(p)$ represents the collection of bundles that maximizes the net utility of bidder i under price vector p. We say that the gross substitutes condition is satisfied, if for any two price vectors p and p' such that $p' \geq p$, $p'_j = p_j$, and $j \in B \in D_i(p)$, then there exists $B' \in D_i(p')$ such that $j \in B'$.

In short, the gross substitutes condition states that if good j is demanded with price vector p, it is still demanded if the price of j remains the same, while the prices of some other goods increase.

We describe two representative valuations that satisfy the gross substitutes condition.

DEFINITION 8. *unit-demand valuation*
Bidder i's valuation is called unit-demand valuation if, for each good $j \in M$, a valuation v_{ij} is given and, for any bundle B, $v(B, \theta_i)$ is calculated as $\max_{j \in B} v_{ij}$.

In short, a bidder with a unit-demand valuation requires only one of M.

DEFINITION 9. *additive valuation*
Bidder i's valuation is an additive valuation if, for each good $j \in M$, a valuation v_{ij} is given and, for any bundle B, $v(B, \theta_i)$ is calculated as $\sum_{j \in B} v_{ij}$.

In short, the valuation for a bundle of an additive valuation bidder is the sum of the valuations of each good.

We can obtain v' by approximating v by unit-demand valuation or additive valuation. For example, if we approximate the function v by additive valuation, we first calculate a valuation vector $\vec{v}_i = (v_{i1}, v_{i2}, \ldots, v_{im})$ that satisfies the following conditions. Then, we define $v'(B, \theta_i)$ as $\sum_{j \in B} v_{ij}$.

- For each bundle B, $v(B, \theta_i) \leq \sum_{j \in B} v_{ij}$.
- $\vec{v}_i$ is minimal, i.e., the above condition is not satisfied if we decrease any v_{ij}.

So far, we have considered a method for constructing U^* by using a gross substitutes valuation. However, the gross substitutes condition is not a necessary condition for U^* to be submodular [5]. This means that there is a chance that we can construct U^* based on a weaker condition than the gross substitutes condition. In that case, we can expect to obtain a better approximation of v. We intend to examine such a method in our future works.

7. CHARACTERISTICS OF GM-SMA

In this section, we examine the characteristics of the GM-SMA.

[3]The fact that each v' is submodular is not suffcient to guarantee that U^* is submodular [5].

7.1 Allocation Feasibility

The following characteristic holds.

PROPOSITION 1. *The GM-SMA price of a bundle is larger than (or equal to) the VCG price. Also, for bidder i, the difference between the GM-SMA price and the VCG price is the same for every bundle except for an empty bundle, i.e., the difference is equal to* $U^*(M, \Theta_{N\setminus\{i\}}) - V^*(M, \Theta_{N\setminus\{i\}})$.

The proof is straightforward, since U^* is greater than or equal to V^*.

THEOREM 4. *The GM-SMA satisfies allocation feasibility.*

Proof: This is derived from the fact that the VCG protocol satisfies allocation feasibility. From Proposition 1, the GM-SMA prices are larger than the VCG prices by the same amount (except for an empty bundle). Thus, if a bidder is included in a Pareto efficient allocation, the bundle in the allocation maximizes the bidder's utility with the GM-SMA price, or the bidder prefers to obtain an empty bundle. Therefore, if there exist multiple Pareto efficient allocations, the auctioneer can choose one Pareto efficient allocation arbitrarily. Then, every bidder has the highest preference for the bundle that is allocated in that allocation, or an empty bundle. Thus, allocation feasibility is satisfied. □

On the other hand, since a bidder who obtains some goods in the VCG might prefer obtaining an empty bundle, the GM-SMA cannot guarantee Pareto efficiency. This is inevitable since there exists no false-name-proof protocol that satisfies Pareto efficiency.

7.2 False-name-Proofness

In this subsection, we show that the GM-SMA is false-name-proof.

THEOREM 5. *The GM-SMA is false-name-proof.*

Proof: This can be proven by showing that the GM-SMA prices satisfy the NSA condition. We show when bidder i' obtains B' and bidder i'' obtains B'', then $p_{B,i}$, i.e., the price for obtaining $B = B' \cup B''$ with a single identifier i, is less than (or equal to) $p_{B',i'} + p_{B'',i''}$. We can prove the case of more than two identifiers in a similar way.

From our definition of the GM-SMA prices, the following conditions hold. N' is a set of bidders except i, i', i''.

$$p_{B',i'} = U^*(M, \Theta_{N'\cup\{i''\}}) - V^*(M \setminus B', \Theta_{N'\cup\{i''\}})$$

$$p_{B'',i''} = U^*(M, \Theta_{N'\cup\{i'\}})) - V^*(M \setminus B'', \Theta_{N'\cup\{i'\}})$$

$$p_{B,i} = U^*(M, \Theta_{N'}) - V^*(M \setminus B, \Theta_{N'})$$

From the fact that i' obtains B' and i'' obtains B'', the following conditions hold.

$$V^*(M \setminus B', \Theta_{N'\cup\{i''\}}) = v(B'', \theta_{i''}) + V^*(M \setminus B, \Theta_{N'})$$

$$V^*(M \setminus B'', \Theta_{N'\cup\{i'\}}) = v(B', \theta_{i'}) + V^*(M \setminus B, \Theta_{N'})$$

Also, from the definition of U^*, the following conditions hold.

$$\begin{aligned} U^*(M, \Theta_{N'\cup\{i'\}}) &\geq U^*(B', \Theta_{\{i'\}}) + U^*(M \setminus B', \Theta_{N'}) \\ &\geq v(B', \theta_{i'}) + U^*(M \setminus B', \Theta_N) \end{aligned}$$

$$\begin{aligned} U^*(M, \Theta_{N'\cup\{i''\}}) &\geq U^*(B'', \Theta\{i''\}) + U^*(M \setminus B', \Theta_{N'}) \\ &\geq v(B'', \theta_{i''}) + U^*(M \setminus B'', \Theta_{N'}) \end{aligned}$$

From above conditions, we obtain:

$$\begin{aligned} & p_{B',i'} + p_{B'',i''} - p_{B,i} \\ &= U^*(M, \Theta_{N'\cup\{i''\}}) - V^*(M \setminus B', \Theta_{N'\cup\{i''\}}) \\ &\quad + U^*(M, \Theta_{N'\cup\{i'\}}) - V^*(M \setminus B'', \Theta_{N'\cup\{i'\}}) \\ &\quad - [U^*(M, \Theta_{N'}) - V^*(M \setminus B, \Theta_{N'})] \\ &\geq v(B'', \theta_{i''}) + U^*(M \setminus B'', \Theta_{N'}) \\ &\quad - v(B'', \theta_{i''}) - V^*(M \setminus B, \Theta_{N'}) \\ &\quad + v(B', \theta_{i'}) + U^*(M \setminus B', \Theta_{N'}) \\ &\quad - v(B', \theta_{i'}) - V^*(M \setminus B, \Theta_{N'}) \\ &\quad - U^*(M, \Theta_{N'}) + V^*(M \setminus B, \Theta_{N'}) \\ &\geq U^*(M \setminus B', \Theta_{N'}) + U^*(M \setminus B'', \Theta_{N'}) \\ &\quad - U^*(M, \Theta_{N'}) - V^*(M \setminus B, \Theta_{N'}) \\ &\geq U^*(M \setminus B', \Theta_{N'}) + U^*(M \setminus B'', \Theta_{N'}) \\ &\quad - U^*(M, \Theta_{N'}) - U^*(M \setminus B, \Theta_{N'}) \end{aligned}$$

Since U^* is submodular, the following condition holds:

$$\begin{aligned} & U^*(M \setminus B', \Theta_{N'}) + U^*(M \setminus B'', \Theta_{N'}) \\ &\geq U^*(M, \Theta_{N'}) + U^*(M \setminus B, \Theta_{N'}). \end{aligned}$$

Therefore, the above formula is non-negative. Thus, $p_{B',i'} + p_{B'',i''} \geq p_{B,i}$ holds. □

7.3 Robustness against Collusion of Losers

Next, we examine the effect of the collusion of losers.

DEFINITION 10. *strict loser*
We call a bidder who is not included in any Pareto efficient allocation a strict loser.

In the GM-SMA, a loser can be a strict loser or a non-strict loser. A non-strict loser is included in some Pareto efficient allocation but fails to obtain any good.

THEOREM 6. *In the GM-SMA, if strict losers collude and obtain some goods, the sum of their utilities becomes negative.*

Proof: Let us assume a set of strict losers S colludes and obtains B using a single identifier i.

From the fact that each bidder in S is a strict loser, the following formula holds. N' is a set of bidders other than S.

$$V^*(B, \Theta_S) + V^*(M \setminus B, \Theta_{N'}) < V^*(M, \Theta_{N'})$$

Also, the following condition holds for $p_{B,i}$.

$$\begin{aligned} p_{B,i} &= U^*(M, \Theta_{N'}) - V^*(M \setminus B, \Theta_{N'}) \\ &\geq V^*(M, \Theta_{N'}) - V^*(M \setminus B, \Theta_{N'}) \\ &> V^*(B, \Theta_S) \end{aligned}$$

Thus, the sum of the utilities of S becomes negative. Also, since the GM-SMA prices satisfy the NSA condition from Theorem 5, even if S uses multiple identifiers, the sum of the prices does not decrease. Therefore, the sum of the utilities of S remains negative even if S uses multiple identifiers. □

On the other hand, the VCG prices do not satisfy the NSA condition. Therefore, as shown in Example 2, the sum of the losers' utilities can be positive if they collude and use multiple identifiers.

If a Pareto efficient allocation is achieved in the GM-SMA, all losers are strict losers. Thus, the following theorem holds.

THEOREM 7. *When a Pareto efficient allocation is achieved in the GM-SMA, if losers collude and obtain some goods, the sum of their utilities becomes negative.*

7.4 Core Outcome

Next, we examine whether the outcome of the GM-SMA is in the core. First, we show that the following lemma holds.

LEMMA 1. *If U^* is submodular for goods, then U^* is bidder-submodular, i.e., for any set of goods B and for any set of bidders Z, W, the following formula holds: $U^*(B, \Theta_Z) + U^*(B, \Theta_W) \geq U^*(B, \Theta_{Z\cup W}) + U^*(B, \Theta_{Z\cap W})$.*

The details of the proof are described in [12].

The following theorem holds.

THEOREM 8. *When the GM-SMA achieves a Pareto efficient allocation, then the outcome is in the core, i.e., there exists no blocking coalition.*

Proof: Let us assume that goods are allocated to a set of bidders $S_{ww}\cup S_{wl}$ in a Pareto efficient allocation, and bundle B_i is allocated to bidder i in $S_{ww}\cup S_{wl}$. Now, let us assume there exists a blocking coalition that consists of the seller and $S_{ww}\cup S_{lw}$, where $S_{wl}\cap S_{lw} = \emptyset$. In other words, in the blocking coalition, the auctioneer takes away the goods from S_{wl} and re-allocates goods among $S_{ww}\cup S_{lw}$. We assume B'_i is re-allocated to bidder i in $S_{ww}\cup S_{lw}$.

In the blocking coalition, the utility of the seller must be larger than the utility in the GM-SMA, i.e., $\sum_{i\in S_{ww}\cup S_{wl}} p_{B_i,i}$. To maximize the revenue of the seller, while each bidder in $S_{ww}\cup S_{lw}$ obtains the same utility as the GM-SMA, each bidder i in S_{lw} should pay $v(B'_i,\theta_i)$, and each bidder i in S_{ww} should pay $v(B'_i,\theta_i) - (v(B_i,\theta_i) - p_{B_i,i})$. This is because if a bidder pays more than this, the bidder will lose the incentive to join the coalition.

The revenue of the seller in the blocking coalition is given as follows. We represent $B_{S_{ww}} = \bigcup_{i\in S_{ww}} B_i$, i.e., a set of goods allocated to S_{ww} in the Pareto efficient allocation.

$$\begin{aligned}&\sum_{i\in S_{lw}} v(B'_i,\theta_i) + \sum_{i\in S_{ww}} [v(B'_i,\theta_i) - (v(B_i,\theta_i) - p_{B_i,i})]\\ &= V^*(M,\Theta_{S_{ww}\cup S_{lw}}) - V^*(B_{S_{ww}},\Theta_{S_{ww}}) + \sum_{i\in S_{ww}} p_{i,B_i}\end{aligned}$$

By subtracting the revenue of the seller in the blocking coalition from the revenue of the seller in the GM-SMA, the remainder is given as follows:

$$\sum_{i\in S_{wl}} p_{B_i,i} + V^*(B_{S_{ww}},\Theta_{S_{ww}}) - V^*(M,\Theta_{S_{ww}\cup S_{lw}}).$$

The following condition holds. We represent $B_{S_{wl}} = \bigcup_{i\in S_{wl}} B_i$, i.e., a set of goods allocated to S_{wl} in the Pareto efficient allocation.

$$\begin{aligned}&\sum_{i\in S_{wl}} p_{B_i,i}\\ &= \sum_{i\in S_{wl}} [U^*(M,\Theta_{N\setminus\{i\}}) - V^*(M\setminus B_i,\Theta_{N\setminus\{i\}})]\\ &= \sum_{i\in S_{wl}} [U^*(M,\Theta_{N\setminus\{i\}}) - V^*(B_{S_{ww}},\Theta_{S_{ww}})\\ &\qquad - V^*(B_{S_{wl}}\setminus B_i,\Theta_{S_{wl}\setminus\{i\}})] \qquad (1)\end{aligned}$$

From the fact that U^* is bidder-submodular, the following conditions hold.

For $i,j\in S_{wl}, i\neq j$,

$$\begin{aligned}&U^*(M,\Theta_{N\setminus\{i\}}) + U^*(M,\Theta_{N\setminus\{j\}})\\ &\geq U^*(M,\Theta_N) + U^*(M,\Theta_{N\setminus\{i,j\}}).\end{aligned}$$

For $i,j,k\in S_{wl}, i\neq j, j\neq k, k\neq i$,

$$\begin{aligned}&U^*(M,\Theta_{N\setminus\{i\}}) + U^*(M,\Theta_{N\setminus\{j\}}) + U^*(M,\Theta_{N\setminus\{k\}})\\ &\geq 2\cdot U^*(M,\Theta_N) + U^*(M,\Theta_{N\setminus\{i,j,k\}}).\end{aligned}$$

By repeating this, we obtain:

$$\begin{aligned}&\sum_{i\in S_{wl}} U^*(M,\Theta_{N\setminus\{i\}})\\ &\geq (|S_{wl}|-1)U^*(M,\Theta_N) + U^*(M,\Theta_{N\setminus S_{wl}}). \qquad (2)\end{aligned}$$

Also, the following formula holds:

$$\begin{aligned}&\sum_{i\in S_{wl}} V^*(B_{S_{wl}}\setminus B_i,\Theta_{S_{wl}\setminus\{i\}})\\ &= (|S_{wl}|-1)V^*(B_{S_{wl}},\Theta_{S_{wl}}). \qquad (3)\end{aligned}$$

In short, for each $i\in S_{wl}$, $v(B_i,\theta_i)$ is counted $|S_{wl}|-1$ times in the above formula.

From formulae 1,2, and 3, we obtain:

$$\begin{aligned}&\sum_{i\in S_{wl}} p_{B_i,i} + V^*(B_{S_{ww}},\Theta_{S_{ww}}) - V^*(M,\Theta_{S_{ww}\cup S_{lw}})\\ &\geq (|S_{wl}|-1)U^*(M,\Theta_N) + U^*(M,\Theta_{S_N\setminus S_{wl}})\\ &\quad -|S_{wl}|V^*(B_{S_{ww}},\Theta_{S_{ww}}) - (|S_{wl}|-1)V^*(B_{S_{wl}},\Theta_{S_{wl}})\\ &\quad +V^*(B_{S_{ww}},\Theta_{S_{ww}}) - V^*(M,\Theta_{S_{ww}\cup S_{lw}})\\ &\geq (|S_{wl}|-1)U^*(M,\Theta_N)\\ &\quad -(|S_{wl}|-1)(V^*(B_{S_{ww}},\Theta_{S_{ww}}) + V^*(B_{S_{wl}},\Theta_{S_{wl}}))\\ &\quad +U^*(M,\Theta_{S_N\setminus S_{wl}}) - V^*(M,\Theta_{S_{ww}\cup S_{lw}})\\ &\geq 0.\end{aligned}$$

As a result, the revenue of the seller in this coalition does not exceed the revenue of the GM-SMA. Therefore, this coalition cannot be a blocking coalition. □.

8. EVALUATION

In this section, we evaluate the social surplus and the revenue of the seller in the GM-SMA through a simulation. As a representative of existing false-name-proof protocols, we use the M-MB protocol [10]. This is because using another false-name-proof protocol, i.e., the LDS protocol, would require us to tune various parameters to obtain a good social surplus [11].

We decide the valuation of bidder i by the following method.

- With probability r, where $0\leq r\leq 1$, bidder i is a *single-minded* bidder, and with probability $1-r$, bidder i is a unit-demand valuation bidder.
- If bidder i is a single-minded bidder, i wants to have bundle B, where $|B| = k$. B is chosen randomly, and the valuation for B is randomly chosen from $[0, 1000k]$. The gross utility of i is 0 if i fails to obtain any single good in B.
- If bidder i is a unit-demand valuation bidder, i wants to have any one of k goods. These goods are chosen randomly, and the valuation for the good is randomly chosen from $[0, 1000]$.

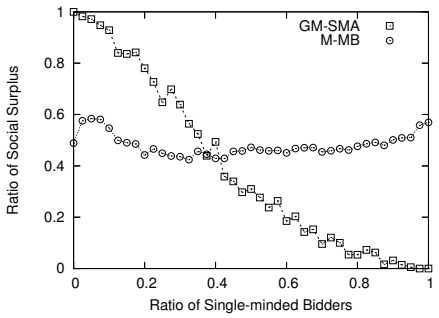


Figure 1: Ratio of Social Surplus

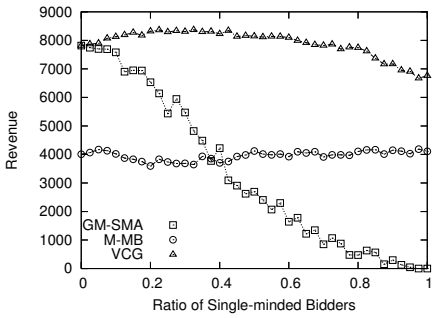


Figure 2: Revenue

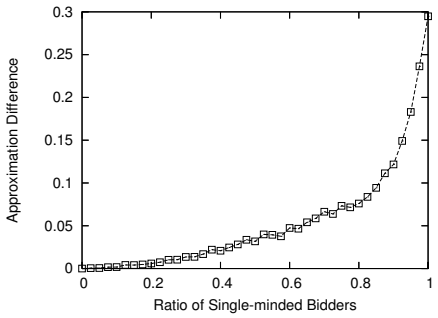


Figure 3: Approximation Difference

If bidder i is a unit-demand valuation bidder, we use $v'(\cdot,\cdot) = v(\cdot,\cdot)$. If bidder i is a single-minded bidder, we approximate v using an additive valuation, where $v_{ij} = v(B,\theta_i)/|B|$ for $j \in B$.

We set $N = 50$, $M = 10$, and $k = 4$, and generated 100 instances. Figure 1 shows the average social surplus of the GM-SMA and the M-MB Protocol by varying the ratio of single-minded bidders. If the ratio is zero, all bidders have unit-demand valuation and their valuations are submodular. Thus, the outcome of the GM-SMA is equivalent to that of the VCG. The social surplus is normalized by the social surplus of a Pareto efficient allocation. Figure 2 shows the revenue of the GM-SMA, M-MB, and VCG protocols. In the VCG protocol, we assume each bidder declares his/her true valuation without using false-name bids. We can see that when the ratio of single-minded bidders is less than 0.4, the GM-SMA outperforms the M-MB protocol both in terms of the efficiency and the revenue.

Also, to evaluate the accuracy of the submodular approximation, for each bidder, we calculate *approximation difference* defined as follows:

$$\frac{U^*(M,\Theta_{N\setminus\{i\}}) - V^*(M,\Theta_{N\setminus\{i\}})}{V^*(M,\Theta_{N\setminus\{i\}})}.$$

If this approximation difference were smaller, the submodular approximation would be more accurate. Figure 3 shows the average approximation difference. We can see that the efficiency and the revenue of the GM-SMA critically depend on the accuracy of the submodular approximation.

9. CONCLUSIONS

In this paper, we developed a new combinatorial auction protocol called the Groves Mechanism with SubModular Approximation (GM-SMA). The basic ideas of this protocol are as follows: (i) it is based on the VCG protocol, (ii) it utilizes submodular approximation, i.e., when calculating the payment of bidder i, we approximate the valuations of bidders other than i by using a submodular valuation function.

The GM-SMA satisfies the following characteristics: (1) it is false-name-proof, (2) each winner is a bidder who is included in a Pareto efficient allocation, (3) as long as the allocation is Pareto efficient, the protocol is robust against the collusion of losers and the outcome is in the core. The GM-SMA is the first protocol that satisfies all three of these characteristics.

Simulation results showed that the GM-SMA achieves a better social surplus and seller's revenue than an existing false-name-proof protocol, as long as the submodular approximation is close enough to the original valuations.

The efficiency and the revenue of the GM-SMA critically depend on the accuracy of the submodular approximation. Our future works include investigating better approximation methods.

10. REFERENCES

[1] L. M. Ausubel and P. R. Milgrom. Ascending auctions with package bidding. *Frontiers of Theoretical Economics*, 1(1), 2002.

[2] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 2:19–33, 1971.

[3] S. de Vries and R. V. Vohra. Combinatorial auctions: A survey. *INFORMS Journal on Computing*, 15, 2003.

[4] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[5] F. Gul and E. Stacchetti. Walrasian equilibrium with gross substitutes. *Journal of Economic Theory*, 87:95–124, 1999.

[6] B. Holmstrom. Groves' scheme on restricted domains. *Econometrica*, 47(5):1137–1144, 1979.

[7] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[8] R. B. Myerson. Optimal auction design. *Mathematics of Operation Research*, 6:58–73, 1981.

[9] W. Vickrey. Counter speculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[10] M. Yokoo. The characterization of strategy/false-name proof combinatorial auction protocols: Price-oriented, rationing-free protocol. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 733–739, 2003.

[11] M. Yokoo, Y. Sakurai, and S. Matsubara. Robust combinatorial auction protocol against false-name bids. *Artificial Intelligence*, 130(2):167–181, 2001.

[12] M. Yokoo, Y. Sakurai, and S. Matsubara. The effect of false-name bids in combinatorial auctions: New fraud in Internet auctions. *Games and Economic Behavior*, 46(1):174–188, 2004.

Reducing Price Fluctuation in Continuous Double Auctions through Pricing Policy and Shout Improvement

Jinzhong Niu and Kai Cai
Dept of Computer Science
Graduate Center
City University of New York
365, 5th Avenue
New York, NY 10016, USA
{jniu,kcai}@gc.cuny.edu

Simon Parsons and Elizabeth Sklar
Dept of Computer and
Information Science
Brooklyn College
2900 Bedford Avenue
Brooklyn, NY 11210, USA
{parsons,sklar}@sci.brooklyn.cuny.edu

ABSTRACT

Auction mechanism design is a subfield of game theory dedicated to manipulating the rules of an auction so as to achieve specific goals. The complexity of the dynamics of auctions, especially continuous double auctions, makes it difficult to apply the usual analytic game-theoretic methods to do this. This paper takes an experimental approach, searching a parameterized space of possible auction types, and presents a new pricing policy for continuous double auctions. The paper further demonstrates how this policy, together with a shout improvement rule, helps to reduce the fluctuation of transaction prices in auctions involving agents with minimum intelligence while keeping the overall efficiency high.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, design, economics, experimentation, measurement, performance

Keywords

Continuous double auction, pricing policy, shout improvement rule

1. INTRODUCTION

Auctions, as a subclass of *markets*, have been widely used in solving real-world resource allocation problems, and in structuring stock or futures exchanges. This is due to the fact that auctions, when well designed [9], achieve desired economic outcomes like high *allocative efficiency*. Auctions are also an excellent scenario in which to study the dynamics and characteristics of micro-economics and they play an important role in computer-based control systems [2].

Research on auctions originally interested economists and mathematicians. They view auctions as games and have successfully applied traditional analytic methods from game theory to some kinds of auctions [11], for example the second-price sealed-bid auctions or Vickrey auctions [23]. The high complexity of the dynamics of some other auction types, especially *double-sided auctions* [5] or DAs[1], however makes it difficult to go further in this direction [10, 20, 25].

As a result, researchers turned to experimental approaches. Starting in 1955, Smith pioneered the experimental approach [21] conducting auctions involving human traders that revealed many of the properties of double auctions. For example, his work showed that in *continuous double auctions* or CDAs[2], even a handful of traders can lead to high overall efficiency, and transaction prices can quickly converge to the theoretical equilibrium. Smith's focus was mainly on the convergence of transaction prices in different scenarios rather than examining why high efficiency is obtained—this is the same topic that interests us. In a computerized world, a question that arises naturally is whether these results can be replicated in electronic auctions. In Smith's experiments, as in real markets traditionally, the traders are human beings, but computer programs are supposed to be automatic and work without human involvement. Obviously humans are intelligent creatures, but programs are not, at least for the foreseeable future. Is it intelligence that contributes to the high efficiency, or something else?

Gode and Sunder [7, 8] were among the first to address this question, claiming that no intelligence is necessary for the goal of achieving high efficiency; so the outcome is due to the auction mechanism itself. They introduced two trading strategies: *zero intelligence without constraint* (ZI-U) and *zero intelligence with constraint* (ZI-C), and showed that ZI-U, the more naïve version, which shouts an offer at a random price without considering whether it is losing money or not, performs poorly. In contrast, ZI-C — which lacks the motivation of maximizing profit just like ZI-U, but guarantees no

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ..$5.00

[1] In DAs, both competing sellers and buyers can make offers, in contrast to the most common auction mechanisms, such as the English auction, where only buyers bid.

[2] A CDA is a continuous DA in which any trader can accept an offer and make a deal any time during the auction period.

loss — generates high efficiency solutions [7] and provides a lower bound on the efficiency of markets [8]. These results were however questioned by Cliff and Bruten [2], who thought Gode and Sunder's conclusion was not convincing because the scenarios considered were not as comprehensive as in Smith's experiments, and showed [2] that in different scenarios the ZI-C agents performed poorly, especially in terms of convergence to the theoretical equilibrium.

Cliff and Bruten further designed an adaptive trading strategy called *zero intelligence plus* or ZIP, and showed ZIP worked better than ZI-C, generating high efficiency outcomes and converging to the equilibrium price. This led Cliff and Bruten to suggest that ZIP embodied the minimum intelligence required by traders. Subsequent work has led to the development of further trading strategies, including that proposed by Roth and Erev [18], which we call RE, and that suggested by Gjerstad and Dickhaut [6], commonly referred to as GD.

This work on trading strategies is only one facet of the research on auctions. Gode and Sunder's results suggest that auction mechanisms play an important role in determining the outcome of an auction, and this is further bourne out by the work of Walsh *et al.* [25] (which also points out that results hinge on both auction design and the mix of trading strategies used). For example, if an auction is *strategy-proof*, traders need not bother to conceal their private values and in such auctions complex trading agents are not required. While typical double auctions are not strategy-proof, McAfee [12] has derived a form of double auction that is strategy-proof (though this strategy-proofness comes at the cost of lower efficiency).

The work described here differs from the bulk of the literature. It differs because that bulk of work considers the design of trading strategies, looking for approaches that can achieve high efficiency [6, 17] or high profit [19, 22]. In contrast, we follow the work of Phelps *et al.* in looking to optimize several metrics — price fluctuation and efficiency — by manipulating the design of the auction itself, and we aim to do this, as Walia *et al.* [24] do, for populations of ZI-C agents. In particular, this paper presents some preliminary experimental results in designing auction mechanisms, showing how a new *pricing policy* and a new *shout improvement rule* can augment a traditional continuous double auction involving zero intelligence agents to reduce the fluctuation of transaction prices, while keeping overall efficiency high.

This paper is organized as follows. Section 2 discusses recent related work in auction mechanism design. Section 3 introduces our new pricing policy and shout improvement rule, and describes the experimentation work and results. We follow that with a discussion of our results in Section 4 and conclude with a brief summary in Section 5.

2. AUCTION DESIGN

Mechanism design applied to auctions explores how to design the rules that govern auctions to obtain specific goals.

2.1 Auction types

Generally an auction involves buyers and sellers who aim to exchange commodities at agreed prices. The most common auction form is English auction, in which there is a single seller, and multiple buyers compete by making increasing bids for the commodity — the one who offers the highest price wins the right to purchase the good being auctioned. Since only one type of trader makes offers in an English auction, it belongs to the class of *single-sided auctions*. Accordingly, there are *double-sided auctions* or DAs, in which both sellers and buyers make *shouts*[3]. Two most common forms of DAs are *clearing house* or CH and *continuous double auction* or CDA. In a CH, an auctioneer collects *bids* — shouts made by buyers — and *asks* — shouts made by sellers; the market-clearing (or equilibrium) price is then determined by maximizing the number of transactions and guaranteeing that if a trader is involved in a transaction, those with more competitive offers are also involved; and the market finally clears at that price. In a CDA, a trader can make a shout and accept an offer from someone at any time, which makes a CDA able to process many transactions in a short time. Both kinds of double auction are of practical importance as well, with, for example, CDA variants being widely used in real-world stock or trading markets including the New York Stock Exchange (NYSE) and Chicago Mercantile Exchange.

One can think of different forms of auctions as varying according to certain aspects of the auction rules, forming a parameterized auction space. For example, Wurman *et. al.* [27] parameterized auction rules using the following classification:

- Bidding rules: How many sellers and buyers are there? Are both groups allowed to make shouts? How is a shout expressed? Does a shout have to beat the corresponding market quote if one exists?
- Information revelation: When and what market quotes are generated and announced? Are shouts visible to all traders?
- Clearing policy: When does clearing a market take place? When does a market close? How are crossing shouts matched? How is a transaction price determined?

2.2 Performance metrics

Auctions with different values for the parameters described above may vary greatly in performance. Popular performance measurements include, but are not limited to, *allocative efficiency* and Smith's *coefficient of convergence* α.

The allocative efficiency, E_a, of an auction is used to measure how much social welfare is obtained through an auction. The *actual overall profit*, P_a, of an auction is:

$$P_a = \sum_i |v_i - p_i| \tag{1}$$

where p_i is the price of a trade made by agent i and v_i is the private value of agent i for all agents who trade. The *theoretical* or *equilibrium profit*, P_e, is:

$$P_e = \sum_i |v_i - p_0| \tag{2}$$

for all agents whose private value is no less competitive than the equilibrium price, where p_0 is the equilibrium price, the price at which, if all agents bid their private value, supply would equal demand. Given these:

$$E_a = \frac{100 P_a}{P_e} \tag{3}$$

[3]Another term for offers made by traders.

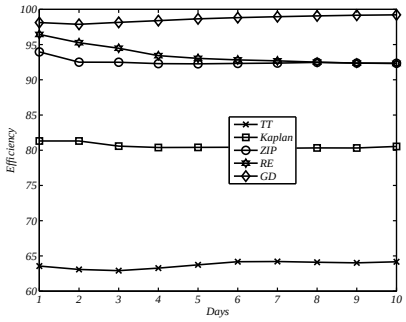


Figure 1: Typical values of E_a for CDA markets populated with common trading strategies

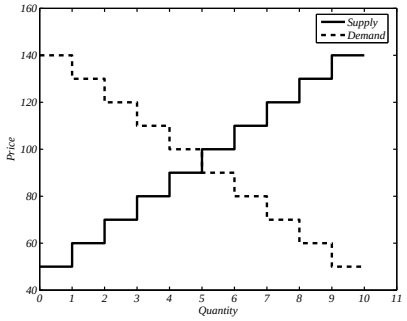


Figure 2: The supply and demand schedules

By way of indicating typical efficiencies achieved in a CDA, Figure 1 shows the trend of the overall efficiencies of homogeneous CDAs lasting 10 days with 50 rounds per day[4] in which 10 buyers and 10 sellers all use the same strategy, one of: TT[5], Kaplan[6], ZIP, RE, and GD. The results are averaged over 400 iterations and obtained in JASA—an extensible Java-based auction simulation environment [13].

The convergence coefficient, α, was introduced by Smith [21] to measure how far an active auction is away from the equilibrium point. It actually measures the deviation of transaction prices from the equilibrium price:

$$\alpha = \frac{100}{p_0}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - p_0)^2} \qquad (4)$$

2.3 Auction design and performance

It is not yet clear how auction design, and thus the choice of parameter values, contributes to observed performance. Thus it is not possible to determine how to create an auction with a particular specification. It *is* possible to design simple mechanisms in a provably correct manner from a specification, as shown by Conitzer and Sandholm [3, 4]. However it is not obvious that this kind of approach can be extended to mechanisms as complex as double auctions. As a result, it seems that we will have to design double auction mechanisms experimentally, at least for the foreseeable future.

Of course, doing things experimentally does not solve the general problem. A typical experimental approach is to fix all but one parameter, creating a one-dimensional space, and then measure performance across a number of discrete sample points in the space, obtaining a fitness landscape that is expected to show how the factor in question correlates to a certain type of performance and how the auction can be optimized by tweaking the value of that factor [16]. In other words, the experimental approach examines one small part of a mechanism and tries to optimize that part[7]. The situation is complicated when more than one factor needs to be taken into consideration — the search space then becomes complex and multiple dimensional, and the computation required to map and search it quickly becomes prohibitive. Instead of manual search, some researchers have used evolutionary computation to automate mechanism design in these situations.

For example, Cliff [1] explored a continuous space of auction mechanisms by varying the probability of the next shout (at any point in time) being made by a seller, denoted by Q_s. The continuum includes the CDA ($Q_s = 0.5$) and also two purely single-sided mechanisms that are similar to the English auction ($Q_s = 0.0$) and the Dutch flower auction ($Q_s = 1.0$). Cliff's experiments used genetic algorithms and found that a Q_s that corresponds to a completely new kind of auction led to a better α value than that obtained for other auctions when all traders used the ZIP strategy. Walia, Byde and Cliff [24] continued with this work, showing that the new "irregular" mechanism can lead to high efficiency in ZI-C markets as well. Taking another tack, Phelps *et al.* [14, 16] demonstrated how genetic programming can be used to find an optimal point in a space of pricing policies, where the notion of optimality is based on efficiency and trader market power, and transaction prices are determined from matching bids and asks and market quotes.

In this paper we make some preliminary steps towards an evolutionary computation analysis that extends the work of [24]. In particular, we demonstrate that a new policy for setting transaction prices and a new shout improvement rule are capable of lowering the fluctuations in transaction price while keeping efficiency high. Both these new mechanisms are parameterized, and so the space of all such mechanisms can be searched using evolutionary techniques.

3. EXPERIMENTAL SETUP

We start by looking at the impact of a new pricing policy and then go on to look at a shout improvement rule.

3.1 A new pricing policy

A pricing policy defines how transaction prices are determined from the offers made by traders. In traditional CDAs, a typical pricing policy sets the price in the middle of the

[4] In an experimental environment, an auction typically includes a specific number of *days* (called *periods* in some of the literature), and each day in turn consists of a number of consecutive *rounds*. At the beginning of each day, traders are initialized and the auction begins/resumes running until the day ends. Days in an auction are totally isolated from each other except that knowledge obtained by traders over the previous days may remain. The division of days helps to identify the change of performance caused by the adaptive behavior of traders with the accumulation of knowledge over time.

[5] TT denotes the Truth-telling strategy, in which agents truthfully report their private values.

[6] "Kaplan" refers to Todd Kaplan's sniping strategy, in which agents wait until the last minute before attempting to steal the deal [19].

[7] And of course there are rarely any guarantees as to the optimality of the results.

matching ask-bid pair using the following expression with $k = 0.5$

$$p = k \cdot p_a + (1 - k) \cdot p_b \quad (5)$$

where p_a and p_b are prices offered respectively by the seller and the buyer. This is the usual *k-double auction* pricing rule [20], which is clearly a discriminatory policy[8], and which we denote as the KP*ricing*P*olicy* hereafter. In contrast to this, CHs usually adopt a non-discriminatory or *uniform* pricing policy. Phelps *et al.* [16] have shown that $k = 0.5$ is the best value to use when optimizing a combination of allocative efficiency and trader market power[9].

Determining the transaction price merely by the corresponding bid and ask, however, may be problematic. In a CDA with ZI-C traders using the KPricingPolicy (denoted as kCDA-ZIC hereafter), the transaction prices fluctuate wildly as shown in Figure 3(a). This result, which is due to the randomness with which ZI-C makes offers, is seen in Gode and Sunder's results [7]. In contrast, in markets involving more complex trading strategies like GD, the phenomenon is less obvious. Figure 3(d) shows the same measurements as Figure 3(a) but for all-GD traders (a setup called kCDA-GD hereafter).

The above results are obtained by running kCDA-ZIC and kCDA-GD in JASA with similar settings to the auctions that gave rise to Figure 1. Figure 3(a) and Figure 3(d) show the first 10 iterations, each with a different random seed, detailing the dynamic characteristics of the transaction prices set in these auctions. It is important to note that this is 10 separate runs over 10 days, with each run starting from scratch. If we did not start over from scratch at the end of each 10 days of trading, the prices for GD would fluctuate even less because there would be no "start of run" variation. The supply and demand schedules are fixed for all the iterations and those in Figure 2.

There are several reasons why we consider it worth studying the fluctuation in transaction prices in markets populated by ZI-C traders. First, as Walia *et al.* [24] argue, auction mechanisms that do well with ZI-C are likely to be "robust to trader-irrationality", in other words they are markets that will work reliably well. Second, markets in which one can trade with minimal intelligence and still have good performance are nearly as good as markets that are strategy-proof — traders are free from some of the burden of having to make good offers. Finally, reducing fluctuation is good for participants in the market — participants in a low fluctuation market are guaranteed not to be given a transaction price too far from the equilibrium, and this may mean traders are more likely to use this kind of market than another with higher fluctuations.

To reduce the fluctuation, we used a new pricing policy that we call the NP*ricing*P*olicy*. This keeps a sliding window of matching ask-bid pairs and uses the average price of those shouts to set the transaction price. If n is the window length and t the number of matching ask-bid pairs at a given point in time, then the transaction price for the tth transaction is given by:

$$p_t = \frac{1}{2n} \sum_{i=(t-n+1)}^{t} (p_{a_i} + p_{b_i}) \quad (6)$$

where p_{a_i} and p_{b_i} are the ask and bid prices corresponding to the ith transaction respectively. The NPricingPolicy generalizes the usual k-double auction pricing policy (KPricingPolicy) since when $n = 1$, it becomes the KPricingPolicy with $k = 0.5$, and when n is the number of matching ask-bid pairs in the entire auction, the NPricingPolicy is the rule commonly used to set prices in a CH[10].

We replaced the KPricingPolicy with the NPricingPolicy, $n = 4$, in the same auctions as before, giving nCDA-ZIC and nCDA-GD respectively. Figures 3(b) shows that the price fluctuation is reduced for ZI-C markets, but is still higher than that of kCDA-GD[11]. This qualitative conclusion is supported by the quantitative data provided by measuring α for the different markets. The means and standard deviations of α and E_a in CDA-ZICs are plotted in Figures 4(a)–4(b), and Table 1 gives the numeric values of α and E_a over the full 10 day run of these markets. Averages and standard deviations are those obtained from 400 iterations of each 10 day run.

The values from Table 1 tell us $\alpha = 12.2$ for the nCDA-ZIC. This value plus twice its standard deviation (1.5) is less than $\alpha = 15.6$ for the kCDA-ZIC. Since the 95% confidence interval is two standard deviations around the mean, we can conclude that the NPricingPolicy significantly reduces α in ZI-C markets. By the same calculation, the improvement of α in the nCDA-GD is not significant (it is less than one standard deviation).

We note that the NPricingPolicy merely sets transaction prices differently, which affects the distribution of profit across the traders but not the overall efficiency. This is confirmed by the fact that the E_as of the nCDA-ZIC and nCDA-GD are identical to their kCDA counterparts as shown in Figure 4(b), where the values are superimposed on each other, and Table 1.

Overall, then, we can conclude for now that the nCDA is significantly better than the traditional CDA in terms of α when populated by ZI-C agents and no worse in terms of efficiency. The results for GD agents suggest that nCDA is robust against different types of trading strategies. To confirm this we need to assess its performance with other trading strategies.

3.2 Shout improvement rule

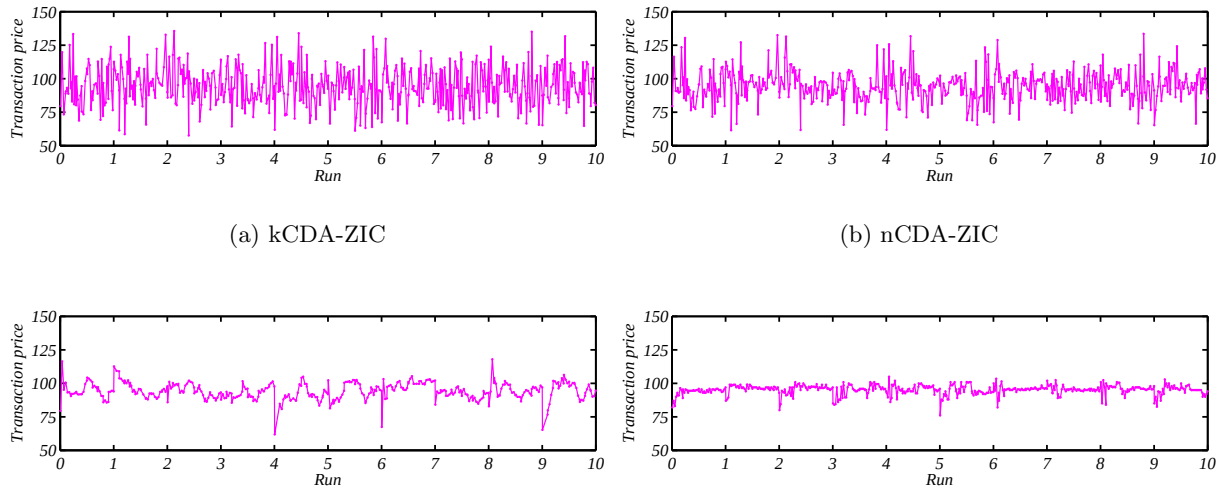
Though the NPricingPolicy does well in reducing fluctuation of transaction prices in the kCDA-ZICs, the difference between $\alpha = 12.2$ for the nCDA-ZIC vs. $\alpha = 4.0$ for the nCDA-GD in Table 1 tells us that there may still be space to improve the performance of markets with ZI-C traders. Why do we still get the obvious fluctuation of transaction prices in Figure 3(b) even when we use the NPricingPolicy? The answer is that the random selection of offers by ZI-C causes some variation even though the NPricingPolicy does its best to smooth out prices. This variation is less than that

[8] Meaning it varies from trade to trade.

[9] [16] does not just search through possible values of k. Instead it searches through all possible arithmetic combinations of shout prices and market quotes, evolving a horribly complex expression which is virtually indistinguishable from the $k = 0.5$ version of (5).

[10] Thus n makes a continuum of auction spaces with CDA and CH at either end, and exploring other points in this space is a subject of our current work.

[11] The result for nCDA-GD is similar to that for kCDA-GD in Figure 3(d), so omitted here.



(a) kCDA-ZIC

(b) nCDA-ZIC

(c) nCDAEE-ZIC

(d) kCDA-GD

Figure 3: Transaction price curves

Auction Type	α mean	α stdev	E_a mean	E_a stdev
kCDA-ZIC	15.613	1.352	95.732	1.537
nCDA-ZIC	**12.180**	1.532	**95.732**	1.537
kCDAEE-ZIC	**9.360**	1.039	93.296	2.061
nCDAEE-ZIC	***6.028***	1.353	93.788	1.938
nCDAEEd5-ZIC	**6.221**	1.300	**96.072**	1.657
nCDAEEd10-ZIC	**6.879**	1.117	***96.868***	1.187
nCDAEEd15-ZIC	**8.000**	1.017	**96.864**	1.179
nCDAEEd20-ZIC	**9.220**	1.002	**96.744**	1.144

Auction Type	α mean	α stdev	E_a mean	E_a stdev
kCDA-GD	4.198	1.140	99.224	0.548
nCDA-GD	**4.000**	1.1221	**99.224**	0.548
kCDAEE-GD	5.374	4.909	84.612	26.165
nCDAEE-GD	5.271	5.245	83.460	26.783
nCDAEEd5-GD	4.242	2.779	95.996	13.126
nCDAEEd10-GD	**4.054**	1.262	98.768	1.092
nCDAEEd15-GD	**4.016**	1.256	99.132	0.684
nCDAEEd20-GD	***3.994***	1.217	***99.224***	0.548

Table 1: Metrics for kCDAs, nCDAs, and nCDAEEs measured over all 10 days of the auctions. Bold face indicates the corresponding market outperforms or equals its traditional kCDA counterpart. *Bold italic* points out the best result in the corresponding ZI-C or GD market group.

when we do not smooth prices, but more than that typically generated by GD.

To see how this happens, consider $(p_{a_{(t-1)}}, p_{b_{(t-1)}})$ and (p_{a_t}, p_{b_t}) to be two consecutive matching ask-bid pairs. If the agents are GD, the two spreads — $[p_{a_{(t-1)}}, p_{b_{(t-1)}}]$ and $[p_{a_t}, p_{b_t}]$ — will likely be close if not overlapping since GD makes shouts that are close to previous shouts. This is not true for ZI-C agents for which p_{a_t} and p_{b_t} are selected randomly and independently. Thus for ZI-C agents, $[p_{a_t}, p_{b_t}]$ may easily be well away from the previous spread $[p_{a_{(t-1)}}, p_{b_{(t-1)}}]$. If this is the case, the transaction price, p_t, determined by (6) may fall outside of the spread and the transaction price is selected to be the closest boundary of $[p_{a_t}, p_{b_t}]$ to p_t to make sure neither agents will make a loss.

Whichever of p_{a_t} and p_{b_t} is used in this case, it will be some distance from $[p_{a_{(t-1)}}, p_{b_{(t-1)}}]$. Since $p_{(t-1)}$ has to fall into $[p_{a_{(t-1)}}, p_{b_{(t-1)}}]$ for the same reason, we will get two transaction prices p_t and p_{t-1} that are not near to each other. If the auction mechanism forces ZI-C agents to "adapt" their shouts, then the fluctuation might be avoided.

Our approach is to use a *shout improvement rule* that forces buyers to bid above a running estimate of the equilibrium price, and sellers to ask below it. Thus any matching ask-bid pair must reside on the respective side of the estimate. When we use the NPricingPolicy, the price that is set will not vary far from the previous transaction price. If the estimate changes smoothly and converges to the equilibrium price, the transaction price curve is expected to be smooth as well[12].

More specifically, the shout improvement rule works in a way that is similar to the NPricingPolicy, maintaining a sliding window over a series of m transaction prices and using the average of the prices within the window as the estimate, p_e:

$$p_e = \frac{1}{m} \sum_{i=t-m+1}^{t} p_i \tag{7}$$

where p_i is the transaction price obtained at the time i (de-

[12]Interestingly, we designed the shout improvement rule originally to improve the overall efficiency, since if the estimate is totally accurate (though this is impossible with incomplete knowledge), the efficiency can be guaranteed to be 100.

termined by (6) for the NPricingPolicy). Now, at the beginning of auctions, the transaction prices may deviate greatly from the equilibrium price, and so the rule may cut off those shouts that are actually on the right side of the equilibrium price. As a result, we introduce an additional parameter, δ, to relax the restriction. All bids above $p_e - \delta$ and all asks below $p_e + \delta$ are permitted.

Note that this shout improvement rule is very different from the common "NYSE rule" that forces subsequent offers to be improvements on one another. That rule applies to offers for the same good, and so the value is "reset" by a match between bid and ask. Our rule, in contrast, maintains continuity between the offers made for successive goods (and so, as discussed later, we eventually need to consider its response to price shocks).

We call the auction mechanism obtained by adding the shout improvement rule CDA *with Estimated Equilibrium* or CDAEE. We replace CDA with CDAEE with $n = 4$ and $\delta = 0$ in the 4 experiments described in the previous section and get markets we call *k*CDAEE-ZIC, *n*CDAEE-ZIC, *k*CDAEE-GD, and *n*CDAEE-GD. The results of the 2 ZI-C markets are plotted in Figures 4(c)-4(d) and all numeric values are presented in Table 1. Figure 3(c) demonstrates the dynamic transaction price curve in nCDAEE-ZIC.

Clearly our results show average α is lower for nCDAEE-ZIC (6.0) than for nCDA-ZIC (12.2) and the difference is even more significant than between nCDA-ZIC and kCDA-ZIC (the 95% confidence intervals do not overlap), but average E_a for nCDAEE-ZIC is slightly lower than for nCDA-ZIC. Worse results are observed in nCDAEE-GD, where average α increases and average E_a is down.

One reason for this may be that GD agents adapt their shouts gradually even if their previous shouts were on the wrong side of the estimated equilibrium price and got rejected. Within the limited amount of time as the auctions last, GD agents may not be able to adapt themselves enough to make transactions, which leads to low efficiency.

This indicates that it may be beneficial to relax the shout improvement rule a little and we do this using different positive values of δ—5, 10, 15, 20—in nCDAEE markets, getting 4 new auction mechanisms denoted as *n*CDAEE*d5*, *n*CDAEE*d10*, *n*CDAEE*d15*, and *n*CDAEE*d20*. We ran these mechanisms using ZI-C and GD agents respectively and the results are shown in Figures 4(e)-4(f)[13], and Table 1.

The nCDAEE markets with $\delta \geq 5$ seem to avoid the low efficiency in the nCDAEEs with $\delta = 0$ and the kCDAEEs. Table 1 shows that the nCDAEEd5, 10, 15, 20-ZIC markets perform better than the nCDA-ZIC, and the nCDAEEd20-GD performs no worse than the nCDA-GD in terms of both α and E_a. Though only the improvement on α in markets using ZI-C traders is close to being statistically significant — obtaining low α in the ZI-C markets is our major goal in this paper — we believe that these results show that the shout improvement rule has the potential to augment the improvements demonstrated by the NPricingPolicy.

3.3 Evolving DAs using genetic algorithms

The above manual parameter tweaking is obviously not an ideal method for auction design. Our ultimate goal is to automate the process. As the first step in this direction, we conducted a series of experiments by encoding various auction parameters into a binary vector of 24 bits and evolving a population of 100 DA individuals, each associated with a randomly generated vector at the very beginning.

Each vector specifies (1) what policy is used to determine transaction prices, KPricingPolicy or NPricingPolicy, (2) what rule is used to regulate shout improvement, the NYSE rule, the CDAEE rule using sliding window mechanism as in Section 3.2, or the CDAEE rule using the Widrow-Hoff learning algorithm[14] [26] to adapt the equilibrium price estimate, and (3) when to clear the market, or, in particular, a probability $\tau \in [0, 1]$ with which the market clears whenever a new shout arrives. The search space is thus multi-dimensional, and includes points corresponding to the usual CDA ($\tau = 1$), CH ($\tau = 0$), and all the other auction mechanisms discussed above. This enables us to optimize auction mechanisms automatically by searching the space.

We first experimented up to 100 generations with minimizing α as the goal[15] and all DA individuals were evaluated in the same setting as before using homogeneous ZI-C traders. The results showed that the CDAEE shout improvement rule with the sliding window mechanism not only dominated the other choices across the whole population over generations but also appeared in the fittest mechanism individual found during evolution. This fittest individual takes 0 for δ, which is consistent with the trend exhibited by the nCDAEEd*-ZIC markets in Table 1, where nCDAEE-ZIC, or nCDAEEd0-ZIC, has the lowest average α (4.0). We also tested with maximizing E_a as the goal and found that a similar shout improvement rule prevailed with $\delta = 12$. This result is also confirmative since Table 1 shows that nCDAEEd10-ZIC has the highest average E_a among nCDAEEd*-ZIC markets.

Besides δ, the evolutionary experiments identified the values of other parameters for the fittest auction mechanisms as well. For example in the case of E_a defined as fitness, $\tau = 0.69$ was found to be optimal. This is unexpected since a CH is more efficient than a CDA with the same setup and intuitively clearing a market less frequently helps avoid less competitive shouts being involved in transactions. We leave this issue and others as our future work.

4. DISCUSSION

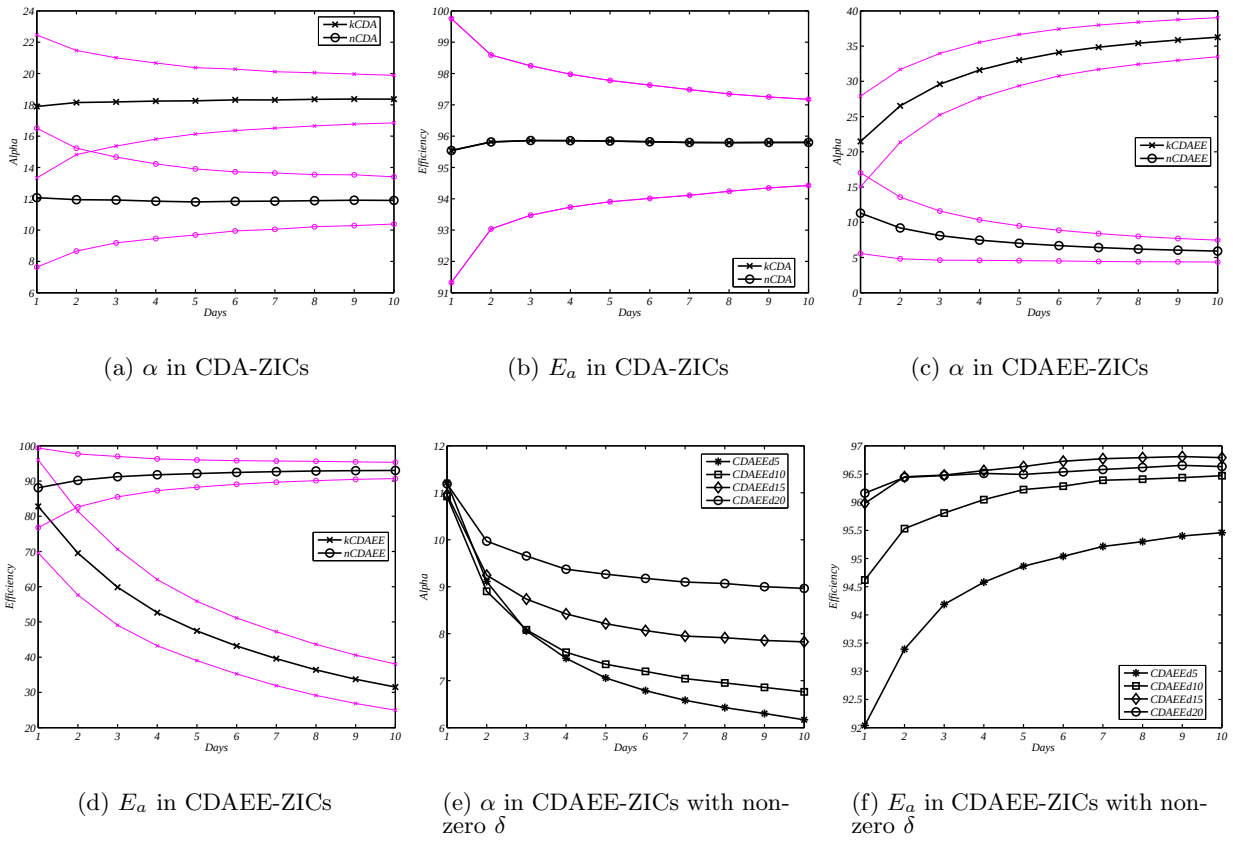
The main result of this paper, then, is that we have identified a new variation on the double auction, a continuous double auction with a new pricing policy and a new shout improvement rule, in which zero intelligence agents can perform well. As Walia *et al.* argue, that we get high efficiency and low α even for ZI-C agents, agents that are on the borderline between rational and irrational trading behavior, suggests that the new auction is pretty robust.

However, there is more we can do to test the robustness of this auction. For a start, we need to run further iterations of the existing experiments to identify the significance or otherwise of the improvements due to the shout improvement rule on top of the new pricing policy. We also need to examine how the new auction performs on other performance metrics and when using trading strategies other than ZI-C and GD, and we are currently running experiments with the other trading strategies in JASA, namely ZIP, Kaplan, RE,

[13] Only ZI-C markets are shown, again to save space, and standard deviations were omitted from these figures to make them cleaner.

[14] the Widrow-Hoff learning algorithm is used in the ZIP and PVT strategies to calculate the target price towards which the shout price should move.

[15] The fitness function is $F = 100 - \alpha$.



(a) α in CDA-ZICs (b) E_a in CDA-ZICs (c) α in CDAEE-ZICs

(d) E_a in CDAEE-ZICs (e) α in CDAEE-ZICs with non-zero δ (f) E_a in CDAEE-ZICs with non-zero δ

Figure 4: Values of α and E_a for ZI-C markets, (a)-(c) with standard deviations, the values in (b) are so close that the points are superimposed on one another.

and PVT, the strategy introduced by Preist and van Tol [17]. If these tests show the mechanism to be worthwhile, we will continue with a heuristic strategy analysis of the kind introduced by Walsh *et al.* [25], using it as in [15] to evaluate the expected profit generated by the auction when traders are free to choose trading strategy.

Finally, we need to examine whether there are negative aspects of the new auction other than those already identified for some parameter settings. We note that the windows used to determine transaction price and to enforce shout improvement are similar to the history used by GD to set its offers. Experiments with GD [6] show that, as one would expect, increasing the window length slows convergence. This can be problematic when the market suffers *price shocks* — external events that alter agents' private values. This is not a scenario much considered in the literature, where traders are usually optimized for long term profit and efficiency rather than transient behavior (GD being an exception), but it seems important that the market itself be able to respond swiftly to external pressures, and response to price shocks is something we need to investigate in detail.

5. SUMMARY

This paper has described some experiments in mechanism design for auctions. As a first step towards new work on automated mechanism design, we have explored by hand some aspects of the continuous double auction. In particular, we looked at setting the transaction price by averaging over n different matched bids and asks, and incorporating a shout improvement rule that forces offers to beat the last m offers plus or minus some value δ, and showed that these modifications created an auction that can achieve high efficiency and low variation in transaction price even for agents with zero intelligence.

This new auction type is interesting on its own, as a counterpart to the mechanisms discovered by [1] and [24], but more interesting, we believe, is that n, m, and δ, along with Q_s from [1] provide a parameter space that can be searched automatically for additional new kinds of auction that optimize performance on metrics other than E_a and α, and for agents employing other kinds of trading strategies. The final contribution of this paper is to demonstrate that such an automated search can discover interesting new types of auction.

6. ACKNOWLEDGMENTS

This work was partially supported by NSF IIS-0329037. We thank the anonymous reviewers for their valuable suggestions.

7. REFERENCES

[1] D. Cliff. Evolution of market mechanism through a continuous space of auction-types. Technical report, Hewlett-Packard Research Laboratories, Bristol, England, 2001.

[2] D. Cliff and J. Bruten. Minimal-intelligence agents for bargaining behaviours in market-based environments. Technical report, Hewlett-Packard Research Laboratories, Bristol, England, 1997.

[3] V. Conitzer and T. Sandholm. Automated mechanism design: Complexity results stemming from the single-agent setting. In *The 5th International Conference on Electronic Commerce (ICEC03)*, pages 17–24, Pittsburgh, PA, USA, September 2003.

[4] V. Conitzer and T. Sandholm. An algorithm for automatically designing deterministic mechanisms without payments. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1 (AAMAS'04)*, pages 128–135, New York City, NY, USA, July 2004.

[5] D. Friedman. The double auction institution: A survey. In D. Friedman and J. Rust, editors, *Santa Fe Institute Studies in the Sciences of Complexity*, chapter 1, pages 3–25. Perseus Publishing, Cambridge, MA, 1993.

[6] S. Gjerstad and J. Dickhaut. Price formation in double auctions. *Games and Economic Behavior*, 22:1–29, 1998.

[7] D. K. Gode and S. Sunder. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy*, 101(1):119–137, 1993.

[8] D. K. Gode and S. Sunder. Lower bounds for efficiency of surplus extraction in double auctions. In D. Friedman and J. Rust, editors, *The Double Auction Market: Institutions, Theories and Evidence*, Santa Fe Institute Studies in the Sciences of Complexity, chapter 7, pages 199–219. Perseus Publishing, Cambridge, MA, 1993.

[9] P. Klemperer. How (not) to run auctions: The European 3G telecom auctions. *European Economic Review*, 46(4–5):829–845, 2002.

[10] A. Madhavan. Trading mechanisms in securities markets. *Journal of Finance*, 47(2):607–641, 1992.

[11] E. S. Maskin and J. G. Riley. Auction theory with private values. *American Economic Review*, 75(2):150–155, 1985.

[12] R. P. McAfee. A dominant strategy double auction. *Journal of Economic Theory*, 56(2):434–450, 1992.

[13] S. Phelps. JASA — Java Auction Simulation API. `http://www.csc.liv.ac.uk/~sphelps/jasa/`.

[14] S. Phelps, P. McBurney, S. Parsons, and E. Sklar. Co-evolutionary auction mechanism design: a preliminary report. In *Proceedings of AMEC IV*, 2002.

[15] S. Phelps, S. Parsons, and P. McBurney. An evolutionary game-theoretic comparison of two double-auction market designs. In *Proceedings of AMEC VI*, 2004.

[16] S. Phelps, E. Sklar, and S. Parsons. Using genetic programming to optimise pricing rules for a double auction market. In *Proceedings of the workshop on Agents for Electronic Commerce,*, Pittsburgh, PA, 2003.

[17] C. Preist and M. van Tol. Adaptive agents in a persistent shout double auction. In *Proceedings of the 1st International Conference on Information and computation economies*, pages 11–18. ACM Press, 1998.

[18] A. E. Roth and I. Erev. Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. *Games and Economic Behavior*, 8:164–212, 1995.

[19] J. Rust, J. H. Miller, and R. G. Palmer. *Behaviour of trading automata in a computerized double auction market*, chapter 6, pages 155–199. Santa Fe Institute Studies in the Sciences of Complexity. Perseus Publishing, Cambridge, MA, 1993.

[20] M. Satterthwaite and S. R. Williams. The Bayesian theory of the k-double auction. In D. Friedman and J. Rust, editors, *The Double Auction Market: Institutions, Theories and Evidence*, Santa Fe Institute Studies in the Sciences of Complexity, chapter 4, pages 99–123. Perseus Publishing, Cambridge, MA, 1993.

[21] V. L. Smith. An experimental study of competitive market behaviour. *Journal of Political Economy*, 70(2):111–137, April 1962.

[22] G. Tesauro and R. Das. High-performance bidding agents for the continuous double auction. In *Proceedings of the 3rd ACM Conference on Electronic Commence*, pages 206–209, 2001.

[23] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–27, 1961.

[24] V. Walia, A. Byde, and D. Cliff. Evolving market design in zero-intelligence trader markets. Technical report, Hewlett-Packard Research Laboratories, Bristol, England, 2003.

[25] W. Walsh, R. Das, G. Tesauro, and J. O. Kephart. Analyzing complex strategic interactions in multi-agent systems. In P. Gmytrasiewicz and S. Parsons, editors, *Proceedings of 2002 Workshop on Game-Theoretic and Decision-Theoretic Agents (GTDT-02)*, Edmonton, Alberta Canada, July 2002. AAAI.

[26] B. Widrow and M. E. Hoff. Adaptive switching circuits. In *1960 IRE Western Electric Show and Convention Record*, pages 96–104. August 1960.

[27] P. R. Wurman, M. P. Wellman, and W. E. Walsh. A parametrization of the auction design space. *Games and Economic Behavior*, 35:304–338, 2001.

Instantiating the Contingent Bids Model of Truthful Interdependent Value Auctions

Takayuki Ito*
Center for Coordination Science,
Sloan School of Management,
Massachusetts Institute of Technology
3 Cambridge Center, NE20-336
Cambridge, MA 02142
takayuki@mit.edu

David C. Parkes
Division of Engineering and Applied Sciences,
Harvard University,
Cambridge MA 02138 USA.
parkes@eecs.harvard.edu

ABSTRACT

We consider the problem of auction design with agents that have interdependent values, i.e. values that depend on each others' private signals. We adopt the *contingent bids* model of Dasgupta and Maskin [3], and allow agents to submit bids of the form "if player 1 bids \$x for good A then I will bid \$y." Our main contribution is to identify a specific *linear* valuation model for which there exists an efficient auction for a single item, and then extend this to provide an approximately efficient combinatorial auction with single-minded bidders. In both auction, winners and payments are computed from the fixed point of the valuation mapping defined by contingent bids. We also adopt search in order to construct a variation on the single-item auction with improved revenue. In closing, we discuss the (many) challenges in moving to more general models of interdependent valuations.

1. INTRODUCTION

We consider the problem of auction design with agents that have interdependent values. In this setting, the value of an allocation can depend on the private signals of the other agents.

Allocation problems with interdependent values often arise in auction settings. Consider, for example, the sale of a case of vintage wine to a group of buyers with distributed information about the actual quality (and thus value) of the wine. Some of the bidders may have tasted similar wines, some may have read reviews, while still others may be complete novices. Interdependent values can also arise when a bidder's value for an allocation depends on the details of the allocation of goods to *other* bidders, and thus its value depends (indirectly) on the values of other bidders. Consider an auction for wireless spectrum. A wireless company might assign more value to some bundle of licenses if non-competing companies (e.g. with different business models) win the remaining licenses.

*Visiting from Nagoya Institute of Technology, Japan

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

In addition to settings of electronic commerce, we believe that interdependent value models will be useful for many coordination problems in multiagent systems. Consider, for example decision making about resource allocation on *computational grids*, for instance in allowing a team of scientists to evaluate their value for receiving the right to be the first to work with a new data set. Accurately determining this value may require aggregating the private information of a number of different teams.

The design of efficient (and optimal) auctions with interdependent values has received some attention in the economic theory literature. In particular, under the assumption that the private information (signal) of a bidder can be captured by a single number (in combination with monotonicity requirements that prevent arbitrary information from being encoded in this number), then there exist *ex post* efficient auctions [10]. For more general models with multi-dimensional signals there are strong negative results about the ability to implement efficient outcomes (or even much at all!) in an equilibrium [3, 8].

We too will work within the one-dimensional signal model. Specifically, we follow Dasgupta and Maskin [3], but depart from the other literature on interdependent value models, by working with *contingent bids*. As we discuss in the next section, the alternative is to work in a direct revelation framework, but this appears much more cumbersome for bidders. With contingent bids, a bidder can state information such as "my bid value is \$x when bidder 2's value is \$y." Bidders do not need to describe either their own signal to the mechanism, nor the domain of signals for the other bidders. We agree with Dasgupta and Maskin that this seems very difficult. In our wine example, all bidders would have to agree with the auctioneer on a language to talk about the quality of wine. In our grid computing example, all research groups would have to agree on a common language to discuss the potential value of some new data. In this aspect, our focus on contingent bids differentiates this work from the earlier work in distributed AI of Dash et al. [4] and Ito et al. [5, 7], who consider direct revelation mechanisms.

We adopt contingent bids as first-order representations of agents' private valuation models. Our main contribution is to provide a specific instantiation of Dasgupta and Maskin's [3] contingent bid model. In particular, we define a linear contingent-bid language and establish necessary and sufficient conditions for bids to satisfy the technical conditions required for the existence of an efficient auction. We first introduce this for the single item allocation problem and later extend the language to allow for combinatorial auctions with *single minded* bidders. A single minded bidder is interested in just one particular bundle of goods.

In pursuit of an auction with improved revenue properties, for instance of importance to a seller of vintage wine, we then generalize the auction to include a "dummy" agent that represents the interest of the seller and submits a bid to increase competition and also to have the effect of setting a reservation price. Following the ideas introduced in Likhodedov and Sandholm [12, 13] we then use search to identify an auction with better revenue properties within this class of generalized auctions. Unlike the existing theory on optimal auctions with interdependent values [2], in which *interim* individual rationality (IR) is provided,[1] our auctions have the more reasonable property of *ex post* IR. This means that an agent will never pay more than its value for an allocation.

In closing, we discuss the many challenges in moving towards more general models of interdependent valuations. Even within the setting of one-dimensional signals we identify a significant practical problem in extending multi-unit auctions to allow for more than single-minded bidders. The problem relates to the amount of information that bidders are required to report to an auction about the possible valuations of other bidders.

1.1 Related Work

Dash et al. [4] define a multi-item variant on the efficient, direct-revelation mechanism for multiple identical items in Krishna [10]. Unlike the model of Dasgupta and Maskin [3] (DM), which we adopt in this paper, the model of Dash et al. requires that the mechanism knows the valuation function and signal domains of each bidder. Bidders report their signal and the mechanism determines the outcome and payments. Ito and colleagues [5, 6, 7] also study a model with one-dimensional signals in the context of a multi-item allocation problem. Unlike Dash et al. [4], their mechanisms do not need to know the valuation functions of agents. However, they remain direct revelation schemes, in that agents bid both valuation functions and signals. Again, the language for describing an agent's signal must be common knowledge to all agents and the mechanism. Ito and colleagues assumed special structure to the problem, namely that the interdependency is acyclic: there are "experts" whose signal affect the values of the other agents ("amateurs"). The experts' values are unaffected by the signal of any amateur and no amateur's signal affects the value of any other amateur. This allows for positive results in their model, which is otherwise quite general.

Jehiel et al. [9] and earlier Jehiel and Moldovanu [8], provide strong limits on *ex post* Nash implementation with interdependent valuations. When signals are multi-dimensional then efficiency is generally impossible, and it can be the case that *no* interesting social choice function can be implemented. Although, see Bikhchandani [1] for a discussion of when this negative result breaks down (and implementation is again possible.) In grid computing, an example of a multi-dimensional signal is when an agent has two signals, one relates to the *novelty* of the data set and one relates to the *noise* in the data set. One cannot combine these into one dimension while retaining properties such as increasing value with signal, also required for *ex post* implementation. Multidimensional signals present a new challenge, unseen in private value settings, because there are typically multiple signal values that correspond to the same value on some bundle of goods and no good way to provide incentives for an agent to report the correct *pair* of signals. Reports of the correct pair of signals are important to allow other agents to learn their correct value.

Finally, work on two-stage mechanisms [16] and other ascending price designs (see Krishna [10]) also shares the concern of DM for limiting the knowledge required of the mechanism and agents. We are not aware of previous attempts to instantiate the contingent bid model of DM, as we do in this paper.

[1]Cremer and McLean [2] show that if there is even a small amount of correlation between agent signals then the seller can extract *all* of the surplus from the participants. However, this requires interim IR. We are not aware of any revenue optimality results for the more reasonable requirement of *ex post* IR.

2. PRELIMINARIES

Consider a single item problem. Each agent $i \in \{1, \ldots, N\}$ has a signal $s_i \in S_i$ from some signal space S_i. The value $z_i(s) \geq 0$ of agent i for the item depends on the joint signals $s = (s_1, \ldots, s_N)$.[2] The efficient allocation is to allocate the item to the agent with maximal value. Assume quasilinear utilities, with utility $z_i(s) - p$ for the item at price p. Throughout the paper, vector notation s_{-i} denotes $(s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_N)$.

First suppose that the auctioneer has knowledge of valuation functions $(z_1, \ldots, z_N)$, but not signals. One can construct a truthful and efficient "second-price" auction for some technical restrictions as follows. Each agent reports its signal $\hat{s}_i$, perhaps untruthfully. The auctioneer computes $z_i(\hat{s})$ for every i, allocates the item to the agent with $\max_i\{z_i(\hat{s})\}$ and then charges the agent $p_i =$

$$\min_{s_i'} \quad z_i(s_i', \hat{s}_{-i}) \tag{1}$$
$$\text{s.t.} \quad z_i(s_i', \hat{s}_{-i}) \geq \max_{j \neq i}\{z_j(s_i', \hat{s}_{-i})\}$$

The auction is efficient if valuations satisfy *monotonicity*,

$$\frac{\partial z_i(s)}{\partial s_i} \geq 0, \quad \forall s, \forall i, \tag{2}$$

and the *single-crossing condition* (SCC),

$$\frac{\partial z_i(s)}{\partial s_i} > \frac{\partial z_j(s)}{\partial s_i}, \tag{3}$$

for all s such that $z_i(s) = \max_{j \neq i}\{z_j(s)\}$.

THEOREM 1. *[10] The second-price auction is efficient when the monotonicity and single-crossing property holds.*

PROOF. (sketch) Each agent i faces a price p_i that solves Eq. (1) and is independent of its own report. Suppose the other agents bid truthfully, with $\hat{s}_{-i} = s_{-i}$. Then, by SCC and monotonicity, if agent i bids truthfully ($\hat{s}_i = s_i$) then it is allocated the good whenever its value is greater than p_i, and not allocated the good otherwise. (Fix s_{-i}. Then SCC implies that for all signals less than that which defines p_i agent i is not a winner, and for all signals greater agent i is the winner.) By reporting its true signal, the agent maximizes its utility with respect to a price that is independent of its own report. This is an *ex post* Nash equilibrium (but not dominant strategy equilibrium). Truthful reporting is only best-response when the other agents are also truthful (but whatever their true signals), because the reports of the other agents are required for agent i's true value to be correctly determined. □

This auction reduces to the Vickrey auction for private value models. Yet, there is a practical problem with this protocol. As argued by DM, the auctioneer must know the signal space of agents (in order to define an expressive language for signals), and also the

[2]Notation $z_i : S_1 \times \ldots S_N \rightarrow \mathbb{R}_{\geq 0}$ denotes an agent's value as a function of the signals of all agents. Below, $b_i : \mathbb{R}_{\geq 0}^{N-1} \rightarrow \mathbb{R}_{\geq 0}$ denotes a contingent bid function, and $v_i \in \mathbb{R}_{\geq 0}$ is reserved to denote the *value* of agent i: as determined either via $z_i(s)$ or as the fixed point defined by agents' contingent bids.

valuation function z_i of each agent. This seems to be a heavy burden. Moreover, each agent must report its signal. In general, a signal (for instance information about a case of wine, or the data set in grid computing) can be difficult for a participant to describe.

Instead, DM work in the framework of *contingent bids*. Let $v_i = z_i(s) \geq 0$ denote agent i's value for the good. Agents submit *contingent bids*, $b_i : \mathbb{R}_{\geq 0}^{N-1}$, where $b_i(v_{-i})$ is the (reported) value given values $v_{-i} = (v_1, \ldots, v_{i-1}, v_{i+1}, v_N)$ of other agents. Given contingent bids, the auctioneer determines reported values as the *fixed point* of the mapping:

$$(v_1, \ldots, v_N) \longmapsto (b_1(v_{-1}), \ldots, b_N(v_{-N})) \tag{4}$$

We refer to this fixed point as a *valuation equilibrium*, and denote the fixed point as v°. Agent i is said to bid truthfully when v° corresponds to interdependent valuations $z(s)$ given signals, i.e.

$$b_i(z_{-i}(s_i, s'_{-i})) = z_i(s_i, s'_{-i}), \quad \forall s'_{-i}, \forall s_i \tag{5}$$

Each agent submits a contingent bid, so that its value in the valuation equilibrium is equal to its true value given the signal information s_{-i} that is implicit in the fixed-point values of the other agents. *When all agents bid truthfully the fixed point* $(v_1^\circ, \ldots, v_N^\circ) = (z_1(s), \ldots, z_N(s))$.

DM propose the following contingent bid auction, which is well defined when there exists a *unique* valuation equilibrium:

1. Given bids, compute the valuation equilibrium $(v_1^\circ, \ldots, v_N^\circ)$.
2. Allocate the item to the agent with the maximal fixed-point value, breaking ties at random.
3. For the winner, agent i, compute payment p_i as $\min v_i'$ s.t. $v_i' \geq \max_{j \neq i}\{v_j^*\}$, where $v_j^* = b_j(v_i', v_{-ij}^*)$ for all $j \neq i$, denoting the fixed-point valuations to other agents when agent i's bid is adjusted to some value v_i'.

The payment to the winner is computed as the minimal (non-contingent) value that the winner could have submitted and still remained the winner.

THEOREM 2. *[3] The contingent-bid auction is efficient (in an ex post Nash equilibrium) when valuations satisfy monotonicity and SCC, and when the fixed point of the mapping defined by contingent bids is unique.*

The proof follows (modulo technicalities about the uniqueness of fixed points) from the structure of the proof of Theorem 1. Agents will bid truthfully in equilibrium.

Contingent bids make interdependent value auctions practical. The mechanism does not need to know the signal spaces or valuation functions of agents. On the other hand, DM note that the bidders must now have knowledge of each others' valuation functions to be able to deduce implicit signal information in the valuation equilibrium.[3] We adopt a different view on this issue. Let us simply consider the contingent-value model as *the* underlying model of value interdependency. With this view point, the contingent bid model requires an agent to know enough about other agents (and that this is known by other agents and so on *ad infinitum*) to define its value contingent on their values.

[3]In fact, DM remark that all that is necessary is that the agents are able to make correct inferences at the inflection point that defines the price, i.e. where the winner changes from one agent to another agent.

3. A LINEAR VALUATION MODEL

An actual instantiation of the contingent-bid auction requires a concrete bidding language, designed so that bids submitted in the language define a mapping (Eq. 4) with a unique fixed point, and so that valuations at this fixed point satisfy monotonicity and SCC. In this paper we adopt a linear bidding language,

$$b_i(v_{-i}) = v_{i0} + \sum_{j \neq i} \alpha_{ij} v_j, \tag{6}$$

where $v_{i0} \geq 0$ denotes agent i's stand-alone value when other agents have zero value, and $\alpha_{ij} \in [0, 1)$ is a weight assigned by agent i to the value of other bidders. Conceptually, the stand-alone value v_{i0} plays the role of the *signal* and the α-weights indicate implicit structure about the relationships between valuation functions $z = (z_1, \ldots, z_N)$ in the economy.

3.1 Example

Suppose there are three companies, $\{1, 2, 3\}$, bidding in an auction. Their values are interdependent. For example, company 1 is paying attention to the values of company 2 and company 3:

$$b_1(v_2, v_3) = 50 + 0.3v_2 + 0.5v_3 \tag{7}$$

In Eq. (7), company 1 places the weight 0.3 on the value of company 2 and weight 0.5 on the value of company 3. Similarly, we assume 2 and 3 have the following (truthful) contingent bids:

$$b_2(v_1, v_3) = 60 + 0.4v_1 + 0.4v_3$$
$$b_3(v_1, v_2) = 70$$

Namely, 1 and 2 are paying attention to the values of the other companies, while 3 ignores the others' values. The sum of weights are less than 1.0, which ensures uniqueness of the valuation equilibrium (the details of this result are shown in Section 3.2.) The valuations implied by these bids are computed as the fixed point $(v_1^\circ, v_2^\circ, v_3^\circ)$ of mapping Eq. (4). In this case, the fixed point can be found by solving:

$$v_1^\circ = 50 + 0.3v_2^\circ + 0.5v_3^\circ \tag{8}$$
$$v_2^\circ = 60 + 0.4v_1^\circ + 0.4v_3^\circ \tag{9}$$
$$v_3^\circ = 70, \tag{10}$$

for $(v_1^\circ, v_2^\circ, v_3^\circ) = (126.6, 138.6, 70.0)$. Because at the fixed point, company 2 has the maximum value it is the winner. Its payment is computed as v_2' that solves $v_2' = \max\{v_1^*, v_3^*\}$, where $v_1^* = b_1(v_2', v_3^*)$ and $v_3^* = b_3(v_2', v_1^*)$ (i.e. the updated fixed point values for the other companies when 2's bid is $b_2(v_1, v_3) = v_2'$). We have $v_1^* > v_3^*$, since $v_1^* = b_1(v_2', v_3^*) = 50 + 0.3v_2' + 0.5v_3^*$ and $v_3^* = b_3(v_2', v_1^*) = 70$, and so $v_1^* = 85 + 0.3\max\{v_1^*, v_3^*\}$. Solving, $v_2' = v_1^*$ and $v_1^* = 850/7 \approx 121.4$ solves $v_1^* = 85 + 0.3v_1^*$. Thus, company 2 wins for a price of 121.4. Notice that this is less than v_1°, since the contingent value of company 1 is also reduced in determining the threshold bid at which 2 would have just won.

3.2 Establishing Technical Conditions

Given the linear bidding language we must show the uniqueness of a fixed point, and then also monotonicity and SCC. First, we prove conditions on weights α_{ij} for which the fixed point of the mapping $(v_1, \ldots, v_N) \mapsto (b_1(v_{-1}), \ldots, b_N(v_{-N}))$ is unique. For this we adopt Banach's Fixed Point Theorem (see Vohra [17]).

THEOREM 3 (BANACH'S FIXED POINT THEOREM). *(BFPT) Let (Y, d) be a complete metric space*[4] *and let $f : Y \to Y$*

[4]A distance d is a *metric* for set Y if it satisfies: $d(x, y) + d(y, z) \geq$

be a contraction mapping on Y, i.e. such that there is some real number $q < 1$ for which

$$d(f(x), f(y)) \leq q \cdot d(x, y)$$

for all $x, y \in Y$. Then, the function f admits a unique fixed point x^*, which can be computed as the limit of applying f from any arbitrary point x_0 in Y.

Note that BFPT requires a complete metric space but allows a metric space that is not compact. So, for Y adopt the space of non-negative valuations, i.e. $\mathbf{R}^N_{\geq 0}$ for N bidders. Instantiate mapping f as $b(v) = (b_1(v_{-1}), \ldots, b_N(v_{-N}))$, where b_i is the contingent bid function, Eq. (6). This is a function into $\mathbb{R}^N_{\geq 0}$. Define the distance metric, d, as:

$$d(v, w) = \max_{i \in \{1,\ldots,N\}} |v_i - w_i|, \tag{11}$$

for $v, w \in \mathbb{R}^N_{\geq 0}$. For example, consider two agents with $\alpha_{12} = \alpha_{21} = 0.5$, and consider values $v = (v_1, v_2) \in \mathbf{R}^2_{\geq 0}$ and $w = (w_1, w_2) \in \mathbf{R}^2_{\geq 0}$. Then, we have $d(v, w) = \max(|v_1 - w_1|, |v_2 - w_2|)$ and $d(b(v), b(w)) = \max(|s_1 + v_2/2 - (s_1 + w_2/2)|, |s_2 + v_1/2 - (s_2 + w_1/2)|) = \max(|v_1/2 - w_1/2|, |v_2/2 - w_2/2|)$. Clearly, $\max(|v_1/2 - w_1/2|, |v_2/2 - w_2/2|) < \max(|v_1 - w_1|, |v_2 - w_2|)$. This analysis extends to the general case.

LEMMA 1. *There is a unique valuation equilibrium when $\sum_{j \neq i} \alpha_{ij} < 1$ for all bidders i, and this equilibrium can be computed as the limit to a sequence of the contingent-bid mapping $b : \mathbb{R}^N_{\geq 0} \to \mathbb{R}^N_{\geq 0}$, starting from an arbitrary $v^0 \in \mathbf{R}^N_{\geq 0}$.*

PROOF. Fix some $v, w \in \mathbb{R}^N_{\geq 0}$. To show is $\max_i\{|b_i(v) - b_i(w)|\} < \max_i\{|v_i - w_i|\}$. Consider any agent i. We will establish $|b_i(v) - b_i(w)| < \max_{j \neq i}\{|v_j - w_j|\}$, which implies the required result. Expanding, we require $|s_i + \sum_{j \neq i} \alpha_{ij} v_j - (s_i + \sum_{j \neq i} \alpha_{ij} w_j)| < \max_{j \neq i}\{|v_j - w_j|\}$. Simplifying, we require $|\sum_{j \neq i}(v_j - w_j)| < \max_{j \neq i}\{|v_j - w_j|\}$. For this, it is sufficient to show that $\sum_{j \neq i} \alpha_{ij} |v_j - w_j| < \max_{j \neq i}\{|v_j - w_j|\}$, which is true since $\sum_{j \neq i} \alpha_{ij} < 1$ and $\alpha_{ij} \in [0, 1)$. □

Thus, for uniqueness it is sufficient that the language restricts the total weight that each bidder i assigns to the values of other bidders so that $\sum_{j \neq i} \alpha_{ij} < 1$. For truthful bidding to form an *ex post* Nash equilibrium we also need the analogs to monotonicity and SCC to hold in the contingent-bid model.

LEMMA 2. *For all contingent bids with $\sum_{j \neq i} \alpha_{ij} < 1$ for all bidders i, then $\partial v_j^\circ / \partial v_{i0} < \partial v_i^\circ / \partial v_{i0}$ for all bidders i, j (i.e. the analog to the single-crossing condition).*

PROOF. Consider some finite number of agents, N. Adopt w_i to denote v_{i0} in agent bids. Towards a contradiction, assume

$$\frac{\partial v_1^\circ}{\partial w_1} \leq \frac{\partial v_2^\circ}{\partial w_1} = \alpha_{21} \frac{\partial v_1^\circ}{\partial w_1} + \sum_{j \neq \{1,2\}} \alpha_{2j} \frac{\partial v_j^\circ}{\partial w_1} \tag{12}$$

Since $\sum_{j \neq 1} \alpha_{1j} < 1$ and $\alpha_{1j} \geq 0$ this implies $N > 2$ and $\partial v_j^\circ / \partial w_1 > \partial v_2^\circ / \partial w_1$ for some $j \neq \{1, 2\}$. W.l.o.g., suppose that $\partial v_3^\circ / \partial w_1 > \partial v_2^\circ / \partial w_1 \geq \partial v_1^\circ / \partial w_1$. Considering agent 3, this requires

$$\alpha_{31} \frac{\partial v_1^\circ}{\partial w_1} + \alpha_{32} \frac{\partial v_2^\circ}{\partial w_1} + \sum_{j \neq \{1,2,3\}} \alpha_{3j} \frac{\partial v_j^\circ}{\partial w_1} > \frac{\partial v_2^\circ}{\partial w_1} \tag{13}$$

Thus, we must have $N > 3$ agents. W.l.o.g., suppose that $\partial v_4^\circ / \partial w_1 > \partial v_3^\circ / \partial w_1$. This induction continues indefinitely, leading to a contradiction for any finite number of agents. □

Informally, the system of equations cannot sustain a fixed point at which the dependency of some v_j°, for $j \neq i$, on v_{i0} is greater than that of v_i° itself because the effect is "dampened" in all bids by some multiplier less than one. Monotonicity, which requires $v_i^\circ / \partial v_{i0} \geq 0$ is immediate since all coefficients in the mapping $v \mapsto b(v)$ are non-negative.

THEOREM 4. *The contingent-bid auction defined with the linear contingent-bid language is efficient (in an ex post Nash equilibrium) when the language is expressive.*

By "expressive", we mean that a bidder should be able to use the language to define a bid value that is equal to its true value, whatever the values of the other bidders. The proof follows from Lemmas 1 and 2, together with Theorem 2. Informally, once uniqueness is ensured so that the auction is well defined, fix the bids b_{-i} from agents $j \neq i$. Agent i faces an agent-independent price, defined in terms of the smallest uncontingent bid $b'_i(v_{-i}) = v'_{i0}$ at which it would win in the valuation equilibrium, given bids b_{-i}. Then, by SCC and monotonicity, and assuming the other agents are bidding truthfully, the agent will win (and make this payment) when it bids truthfully whenever its true value is greater than this price and not otherwise. Realize that an agent cannot gain by misstating either its stand-alone value v_{i0} or its α-weights.

4. TOWARDS REVENUE OPTIMALITY

We can consider a family of modified interdependent value auctions to find an auction with improved revenue. Following the direction set by Likhodedov and Sandholm [12] (in private value combinatorial auctions), we introduce a "dummy agent" (indexed zero) to represent the seller and then modify the winner-determination problem by defining static *weights* $\beta_i \in [0, 1]$ on each agent. An allocation $x = (x_0, x_1, \ldots, x_N)$, with $x_i \in \{0, 1\}$, $x_0 \in \{0, 1\}$ to indicate whether the dummy agent receives the item and the item goes unsold, and $\sum_{i=0}^N x_i = 1$, is selected to solve:

$$\max_x \left\{ \beta_0 v_0(x) + \sum_{i=1}^N \beta_i v_i^\circ(x) \right\}, \tag{14}$$

where $v_i^\circ(x) = v_i^\circ$ if $x_i = 1$ and 0 otherwise, $v_0(x) = v_0$ if $x_0 = 1$ and 0 otherwise denotes the value of the dummy agent (v_0 can be thought of as the reservation price), and v_i° are agent's fixed point values.

Value v_0 and weights β are fixed before bids are received and heuristically optimized by the auction designer using prior information about agent values. The weighted contingent-bid auction is defined as:

1. Define v_0 and weights $\beta \in [0, 1]^{N+1}$.
2. Receive bids and compute the fixed point $(v_1^\circ, \ldots, v_N^\circ)$. (Note that agents cannot make their bids contingent on the dummy agent's value, and that the dummy agent's value is uncontingent and fixed at v_0.)

$d(x, z)$, $d(x, y) = d(y, x)$, $d(x, x) = 0$, and $d(x, y) = 0 \Rightarrow x = y$ for all $x, y, z \in Y$. A metric space is *complete* if it contains all limit points of *Cauchy sequences*, where a Cauchy sequence is a sequence in which $\lim_{m \to \infty} d(x_m, x_{m+1}) = 0$. E.g., (0,1) is not complete because $1/2, 1/3, \ldots$ is Cauchy but does not have a limit in the space. On the other hand, $[0, 1]$ is complete, as are $\mathbf{R}^d$ and $\mathbf{R}^d_{\geq 0}$. A complete metric space need not be closed (or compact).

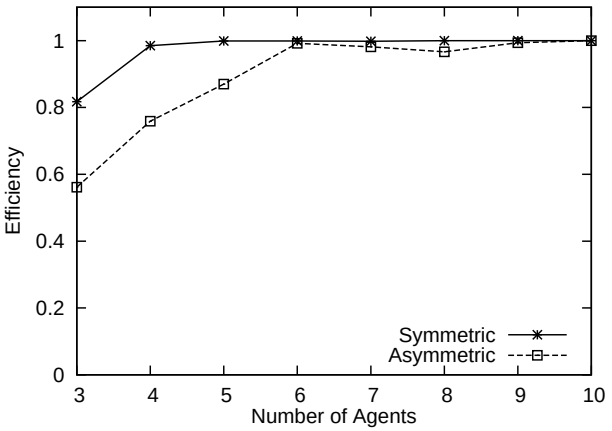


Figure 1: Average Efficiency in the Optimal Auction for the Symmetric and Asymmetric Environments.

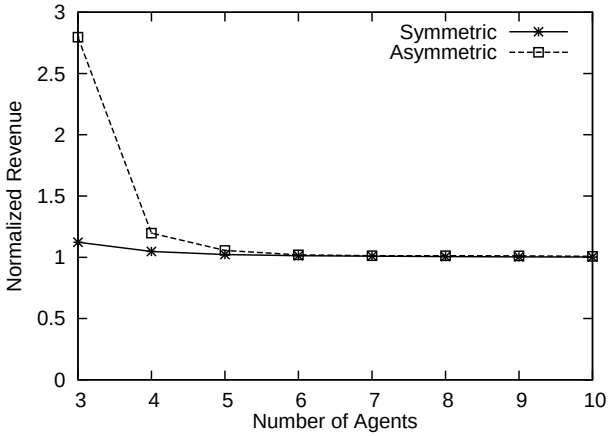


Figure 2: Average Normalized Revenue in the Optimal Auction for the Symmetric and Asymmetric Environments.

3. Compute the winner as the agent with the maximal weighted value. If the dummy agent wins then the good is not sold and no revenue is received.

4. If the item is sold to agent $i \neq 0$, then compute the payment: Decrease v_i' until $\beta_i v_i' = \max_{j \neq i, j=0}\{\beta_j v_j^*\}$, where v_j^* is the new fixed point for agent j given value v_i' to agent i (i.e., $v_j^* = b_j(v_i', v_{-ij}^*)$ for all j). This value, at which i would be just about to stop winning, is the payment.

Clearly, this auction reduces to the standard auction when the weights are symmetric and $v_0 = 0$. From Myerson's [15] work on optimal auctions, we should expect weights that bias in favor of agents that are *a priori* less competitive. It should also be fairly clear that the auction remains truthful. One needs to show that the price to the winner is still independent of its bid, and that the winner wins when its fixed point valuation is above the price, and only then. (The proof is omitted in the interest of space.)

In our experiments, we consider a symmetric environment with the signal s_i for each agent $i \neq 0$ is sampled $s_i \sim Uniform(0, 100)$, and an asymmetric environment with the signal for agent i is sampled from distribution $Uniform(0, L_i)$, where $L_i = 10 + i\delta$ and $\delta = \frac{300-10}{N}$. This has the effect of making the expected signal value of agents increase with the index of the agent in the asymmetric environment. Then, for each agent i we determine the α-weights that define the contingent valuations as follows:

(1) Each $j \neq i$ has $\alpha_{ij} = 0$ with prob. 0.5, otherwise sample α_{ij} from $Uniform(0, 1)$.
(2) Determine a total alpha-weight, m_i, as follows:
(3.1) For the *symmetric* environment, by sampling from a Gaussian distribution with mean 0.5 and standard deviation 0.25, truncated to $[0, 1)$.
(3.2) For the *asymmetric* environment, by sampling from a Gaussian distribution with mean $\mu_i = 0.3 + i\delta'$ and standard deviation 0.25, truncated to $[0, 1)$, with $\delta' = \frac{1.0-0.3}{N}$. The effect is to cause the expected total weight assigned to the values of other agents to increase with the index of the agent.
(4) Normalize weights α_i to have total weight m_i, by multiplying each α_{ij} by $\frac{m_i}{\sum_{j \neq i} \alpha_{ij}}$.

We use search over auction parameters $(v_0, \beta_0, \beta_1, \ldots, \beta_N)$, to maximize the expected revenue. In all cases, we normalize weights so that $w_0 = 1/N$ and $\sum_{i=1}^{n} w_i = 1$, so that $\sum_{i=0}^{n} = (1 + N)/N$. Normalization is achieved by setting weights on $i > 1$ and making w_1 a dependent variable. The search space is discretized, with $v_0 \in \{0, 10, \ldots, 100\}$, and weights $w_i \in \{0, 0.1, \ldots, 1\} \cup \{1/N\}$. Because our problem was quite small and because our goal was to get an initial understanding of whether revenue can be improved in interdependent value auctions we simply enumerate all grid points and select the optimal auction parameters. Future work will explore local search and other approximation methods, as in Likhodedov and Sandholm [13].

To evaluate a parameterization, we average over 4000 instances for the symmetric environment and 5000 instances for the asymmetric environment. We compare the average revenue and efficiency in the optimized auctions with the revenue and efficiency of the standard contingent bid auctions.

Figure 1 illustrates the average efficiency of the optimal auction, i.e. the average ratio of total value in the optimal auction to the maximal value (achieved in the standard auction), for different numbers of agents. Figure 2 illustrates the average normalized revenue of the optimal auction, i.e. the average ratio of total revenue in the optimal auction to the total revenue in the efficient auction, for different numbers of agents. As expected, we see that the auctions can be optimized for revenue, although at some loss in efficiency and with the benefit quickly dropping off as the number of agents increases to 5 or more. In addition, for our distributions, the possible benefit is greater in the asymmetric setting than in the symmetric setting.

Looking at the optimal parameters, the optimal revenue was achieved in the symmetric environment with symmetric weights. For the asymmetric environment, the optimal weights were sometimes asymmetric, for instance when there are 3 agents the optimal weight assignment was 0.5 for agent 1, 0.3 for agent 2, and 0.2 for agent 3. Moreover, optimal weights were always decreasing with agents as we expected from Myerson's [15] study of asymmetric private value auctions. This suggests that for local search it would be useful to restrict the search space to only allow states with weights that are (weakly) decreasing with agent index. The optimal choice for v_0 in the asymmetric environment varied from 50 to 110, and averaged around 80.

5. SINGLE-MINDED CAS

Many interesting auction scenarios have *multiple items* and inter-

dependent values. For instance, consider selling a mix of different bottles and allowing bidders to put together there own "cases" of wine. By adopting *single-minded* bidders, interested in a particular bundle of goods, we consider the simplest possible model of interdependent value CAs. Agents have *two*-dimensional, consisting of the description of the bundle in which it is interested and its signal, conveying information about the value of its bundle (and perhaps about the value of other agents for other bundles.) We construct an approximately efficient auction, which satisfies a generalization of the single-crossing condition.

5.1 Extending the Contingent-Bid Language

Let G denote the set of goods. A single-minded bidder has an *interesting bundle* $W_i \subseteq G$ and an interdependent valuation function $z_i(X, s) \geq 0$ on bundles $X \subseteq G$, defined in terms of signals $s = (s_1, \ldots, s_N)$ as:

$$z_i(X, s) = \begin{cases} z_i(s) & \text{if } X \supseteq W_i \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

Here, we assume *free-disposal*, so that an agent's value is (weakly) increasing with additional goods. Function, $z_i(s) \geq 0$, defines the agent's value for an interesting bundle given signals s.

The contingent-bid language is extended to allow a bid by agent i to report an interesting bundle, $\hat{W}_i$ (perhaps untruthfully), a stand-alone value $v_{i0} \geq 0$, and *weight-bundle pairs* $\{(\alpha_{i1}, X_{i1}), \ldots, (\alpha_{iN}, X_{iN})\}$ for agents $j \neq i$. Taken together, this defines the following contingent bid function:

$$b_i(X, v_{-i}) = \begin{cases} v_{i0} + \sum_{j \neq i} \alpha_{ij} v_j(X_{ij}), & \text{for } X \supseteq \hat{W}_i \\ 0, & \text{otherwise} \end{cases} \quad (16)$$

We follow the earlier notation: b_i is a contingent-bid, now for each bundle X, and $v_i(X)$ denotes an agent's value for a bundle (determined either in a fixed point, or via its valuation function z_i and signals of others). We also require $\alpha_{ij} \in [0, 1)$ and $\sum_{j \neq i} \alpha_{ij} < 1$ for all i, as before. The valuation equilibrium is defined as a fixed point of the mapping $(v_1, \ldots, v_N) \mapsto (b_1(v_{-1}), \ldots, b_N(v_{-N}))$, where $v_i : 2^G \to \mathbb{R}_{\geq 0}$ and $b_i(v_{-i}) : 2^G \to \mathbb{R}_{\geq 0}$ (i.e. the restriction of the agent's contingent bid function, given values of other agents for each bundle.)

An agent's value for some bundle X is defined in terms of the value of other agents $j \neq i$ for other bundles X_{ij}, and also depends on whether $X \supseteq \hat{W}_i$. Realize that X_{ij}, the bundle that bidder i picks out for bidder j, in defining its dependency on the valuation function of bidder j, need not be the same as either $\hat{W}_i$ or $\hat{W}_j$. This provides useful expressiveness as well as ensuring that the knowledge requirements are reasonable. (See the example in Section 5.3.)

THEOREM 5. *The single-minded contingent bid model has a unique fixed point when $\sum_{j \neq i} \alpha_{ij} < 1$ for all i.*

PROOF. By reduction to the single-item contingent bid model. Given a set of contingent bids b, simplify the bids as follows: for each agent i, replace X_{ij} with $\hat{W}_j$ if $X_{ij} \supseteq \hat{W}_j$ and replace X_{ij} with $\emptyset$ otherwise. Since $v_i(\emptyset) = 0$ for all agents, this has the effect of removing any dependence on bundles of this kind. Given this simplification, the fixed point can be characterized as the fixed point to the system of equations

$$b_1(\hat{W}_i, v_{-1}) = v_{10} + \sum_{j \neq 1} \alpha_{1j} v_j(\hat{W}_j) \quad (17)$$

$$\cdots \qquad \cdots$$

$$b_N(\hat{W}_N, v_{-N}) = v_{N0} + \sum_{j \neq N} \alpha_{Nj} v_j(\hat{W}_j), \quad (18)$$

which have exactly the same form as for the single-item contingent bid model because the same bundle $\hat{W}_i$ appears for agent i in all equations. Thus, there is a unique solution. □

5.2 Failure of Generalized SCC

It is tempting to generalize the efficient auction described for the single item case as follows: (1) compute the fixed point valuation functions v° defined by the mapping induced by Eq. (16); (2) compute the efficient allocation $X^* = (X_1^*, \ldots, X_N^*) = \arg\max_{X \in \Gamma} \sum_i v_i^\circ(X_i)$ to maximize total value (breaking ties at random), where Γ denotes the feasible allocations (restricted to allocations in which agent i either receives $\emptyset$ or $\hat{W}_i$); and (3) compute payment to each winner i, as:

$$\begin{aligned} &\min \quad v_i' \\ &\text{s.t.} \quad v_i' + \sum_{j \neq i} v_j^*(X_j^*) \geq \max_{X \in \Gamma} \sum_{j \neq i} v_j^*(X_j), \end{aligned} \quad (19)$$

where $v_j^*(X) = b_j(X, (\vec{v}_i', v^*_{-ij}))$, and $\vec{v}_i'(X) = v_i'$ for $X \supseteq \hat{w}_i$ and zero otherwise. This has the same structure as the previous auction: v^*_{-i} denotes the revised fixed point when agent i bids value v_i' on its interesting bundle $\hat{W}_i$; thus, agent i pays the smallest amount it could have bid and still won, also accounting for the effect of its new report on the values of other agents.

For truthful bidding to be an *ex post* Nash equilibrium in this auction, it would be sufficient for the valuations to satisfy a generalized form of the monotonicity and single-crossing conditions,[5] which combine as:

DEFINITION 1 (GENERALIZED SCC). *If, for signals $(s_1, \ldots, s_N)$ there is a pair of allocations X and X', tied for value, i.e.:*

$$\sum_{j=1}^{N} z_j(X_j, s) = \sum_{j=1}^{N} z_j(X_j', s) = \max_{X'' \in \Gamma} \sum_j z_j(X_j'', s) \quad (20)$$

then for every agent i such that $X_i \neq X_i'$, we require:

$$\left[\frac{\partial}{\partial s_i} z_i(X_i, s) > \frac{\partial}{\partial s_i} z_i(X_i', s) \right]$$
$$\iff \left[\frac{\partial}{\partial s_i} \sum_{j=1}^{N} z_j(X_j, s) > \frac{\partial}{\partial s_i} \sum_{j=1}^{N} z_j(X_j', s) \right] \quad (21)$$

In words, generalized SCC requires that the value of agent i for X is improving more quickly than for X' with respect to its signal whenever this is true for the economy of agents in aggregate, and vice versa. Conceptually, this will be true when the marginal effect of an agent's signal on its own value dominates the marginal effect of its signal on the aggregate economy.

THEOREM 6. *[14] The second-price interdependent value single-minded CA auction is efficient in an ex post Nash equilibrium when the generalized SCC property holds.*

For private value single-minded CAs, this auction is equivalent to the VCG mechanism and truthful bidding is a dominant strategy equilibrium. But, we will show that generalized SCC, and thus incentive compatibility, fails in our interdependent value model. Consider Eq. (21) for single-minded CAs. The LHS of the bidirectional condition is only true when $X_i = W_i$ and $X_i' = \emptyset$. We see that generalized SCC requires that whenever an allocation in

[5]Personal communication [14]. Earlier conditions, as described in Dasgupta and Maskin [3], appear too strong to be useful.

which agent i wins and an allocation in which agent i loses are tied for total value, then if a change in agent i's signal changes the allocation it must be that an *increase* in its signal causes the allocation in which the agent is the winner to be selected. Thus, generalized SCC reduces to the following requirement: *for any signals s_{-i} of other agents, if an agent i is a winner for some signal s_i then the agent must remain a winner for all higher signals.* Notice that the allocation of items to other agents can change for higher signals, as long as agent i always retains bundle W_i.

5.3 Example

Consider goods $G = \{A, B, C\}$ and 5 agents, with (truthful) contingent bids:

$$b_1(AB, v_{-1}) = 10 + 0.5v_2(C) \quad (22)$$

$$b_2(C, v_{-2}) = 10 + 0.4v_1(A) \quad (23)$$

$$b_3(ABC, v_{-3}) = 5 + 0.2v_1(AB) + 0.6v_2(ABC) \quad (24)$$

$$b_4(A, v_{-4}) = 1 + 2/3v_1(ABC) \quad (25)$$

$$b_5(BC, v_{-5}) = 1 + 2/3v_1(ABC) \quad (26)$$

This can be simplified, replacing X_{ij} with W_j if $X_{ij} \supseteq W_j$ and $\emptyset$ otherwise (and removing this term.) We are left with $b_2(C, v_{-2}) = 10 + 0.4v_1(\emptyset) = 10$, $b_1(AB, v_{-1}) = 10 + 0.5v_2(C) = 15$, $b_3(ABC) = 14$, and $b_4(A) = b_5(BC) = 11$.

Suppose agents bid truthfully with $b_i = v_i$ for all agents. The final allocation is $(AB, C, \emptyset, \emptyset, \emptyset)$. To compute the payment by agent 1, we solve (by Eq. 19) $\min v_1'$ s.t. $v_1' + 10 \geq 5 + 0.2v_1' + 6$ (for allocation $(\emptyset, \emptyset, ABC, \emptyset, \emptyset)$), $v_1' + 10 \geq 10 + 1 + 2/3v_1'$ (for allocation $(\emptyset, C, \emptyset, A, \emptyset)$, and $v_1' + 10 \geq 1 + 2/3v_1' + 1 + 2/3v_1'$ (for allocation $(\emptyset, \emptyset, \emptyset, A, BC)$. The first two constraints require $v_1' \geq 1.25, v_1' \geq 3$, while the third constraint not binding. Thus, agent 1's payment is $p_1 = 3$. At a bid equal to this price, the auctioneer would be indifferent between allocation $(AB, C, \emptyset, \emptyset, \emptyset)$ and $(\emptyset, C, \emptyset, A, \emptyset)$. For agent 2, we solve $\min v_2'$ s.t. $10 + 0.5v_2' + v_2' \geq 5 + 0.2(10 + 0.5v_2') + 0.6v_2'$ (allocation $(\emptyset, \emptyset, ABC, \emptyset, \emptyset)$) and $10 + 0.5v_2' + v_2' \geq 2 + 4/3(10 + v_2'/2)$ (allocation $(\emptyset, \emptyset, \emptyset, A, BC)$. Only the second constraint is binding, and requires $v_2' \geq 32/5$. Thus, agent 2's payment is 32/5=6.4.

To see the failure of generalized SCC, let w_1 denote agent 1's stand-alone value and consider what happens as w_1 is increased from 10. When $w_1 = 19$ we have $v_1(AB, v_{-1}) = 19 + 0.5v_2(C)$, and solving for the valuation equilibrium we find $v_1(AB) = 24, v_2(C) = 10, v_3(ABC) = 79/5 = 15.8, v_4(A) = 17$ and $v_5(BC) = 17$. At this point, allocations $X = (AB, C, \emptyset, \emptyset, \emptyset)$ and $X' = (\emptyset, \emptyset, \emptyset, A, BC)$ are tied for value. In addition, $\partial v_1^\circ(X_1)/\partial w_1 > \partial v_1^\circ(X_1')/\partial w_1 = 0$. For the analog to generalized SCC in the contingent-bid model, we need:

$$\frac{\partial}{\partial w_1}\left[v_1^\circ(X_1) + v_2^\circ(X_2)\right] > \frac{\partial}{\partial w_1}\left[v_4^\circ(X_4') + v_5^\circ(X_5')\right], \quad (27)$$

but this does not hold because $\partial(v_1^\circ(X_1) + v_2^\circ(X_2))/\partial w_1 = 1 < \partial(v_4^\circ(X_4') + v_5^\circ(X_5'))/\partial w_1 = 4/3$.

The implication of this failure of generalized SCC is that if agent 1 has true value greater than 34, and the other contingent valuations of agents were unchanged its equilibrium strategy is not to bid truthfully. For example, if its stand-alone value was 20 but its α-weights were unchanged from the example, and if the bids b_{-1} were unchanged, then the agent would *lose* if bidding truthfully. The selected allocation would give A to agent 4 and BC to agent 5. The agent could do better by reporting an uncontingent bid of $b_1(AB, v_{-1}) = 15$; it would win and make a payment of 3.

Failure of generalized SCC occurs in this example because agent 1's value does not affect the value of the other agent (agent 2) with which it must form a winning coalition. On the other hand, its value affects the value of both agents 4 and 5 in the competing coalition.

5.4 A Truthful Single-Minded CA

We can instead define an approximately efficient auction using a *greedy* allocation rule, which is parameterized with some constant $c > 0$. The auction generalizes the auction proposed in Lehmann et al. [11] for single-minded bidders and private values to the setting of interdependent values.[6] It also reduces to our single-item auction when all agents bid on bundles with overlapping goods, so that only one agent can win. The auction is defined as follows:

1. Compute fixed point values $(v_1^\circ, \ldots, v_N^\circ)$ on reported bundles bundles $(\hat{W}_1, \ldots, \hat{W}_N)$.
2. Sort bids in *decreasing* order of $v_i^\circ/|\hat{W}_i|^c$, breaking ties at random. Constant $c > 0$ determines the dependence of bid rank on the number of items in the interesting bundle. Walk down the list, accepting bids if the goods demanded are still unallocated. This is the greedy allocation rule.
3. Let $g_i^c(v^\circ) \in \{0, 1\}$ denote whether agent i is allocated its interesting bundle in the greedy allocation, given fixed point values v°. For a winner, with $g_i^c(v^\circ) = 1$, define the payment as the $\min v_i'$ s.t. $g_i^c((v_i', v_{-i}^*)) = 1$ where v_{-i}^* denotes the new fixed point given (uncontingent) bid $b_i(X, v_{-i}) = v_i'$ on $X \supseteq \hat{W}_i$ and 0 otherwise.

LEMMA 3. *For all contingent bids with $\sum_{j \neq i} \alpha_{ij} < 1$ for all bidders i, the greedy allocation rule satisfies the generalized single-crossing condition (with stand-along value v_{i0} in the role of signal s_i.).*

PROOF. (sketch) From Lemma 2 we have $\frac{\partial}{\partial v_{i0}} v_j^\circ(\hat{W}_j) < \frac{\partial}{\partial v_{i0}} v_i^\circ(\hat{W}_i)$, for all $j \neq i$. So, if agent i is winning for some stand-alone value, v_{i0}, then it will continue to win for a higher signal because its value will increase by more than the value of any other agent and therefore its rank in the ordered list of bids to accept can only improve. □

The greedy rule recovers the generalized SCC requirement because agent i must only compete unilaterally with each other agent that demands one or more of the same goods. By comparison, the efficient allocation rule considered competition between coalitions of agents in deciding whether agent i was a winner.

THEOREM 7. *The greedy contingent-bid auction for single-minded bidders and multiple items supports truthful bidding in an ex post Nash equilibrium when the bidding language is expressive. Moreover, when parameter $c = 1/2$ the greedy auction is $\sqrt{m}-$competitive w.r.t. efficiency, for m goods.*

We omit the proof of equilibrium in the interest of space, and because it is in all essential aspects the same as that of the single-item contingent bid auction. Truthfulness follows from the property of generalized SCC (Lemma 3). The worst-case approximation of $\sqrt{m}$ for the greedy rule with ordering metric $v_i^\circ/|W_i|^{1/2}$ follows immediately from Lehmann et al. [11]. The average case efficiency may be much better.

[6] Lehmann et al. [11] adopted the greedy allocation rule because the efficient winner-determination problem is NP-hard. Here, we adopt the greedy allocation rule because the efficient allocation rule is not implementable in an ex post Nash equilibrium when agents have interdependent values. Of course, computational complexity also provides another reason to prefer this rule in interdependent value problems with many bids.

5.5 Example: Truthful Greedy Auction

For this example, adopt $c = 1$ so that the rank of bids is determined by computing the per-item bid value. Consider the greedy auction on the earlier example, where we have $v_1^\circ(AB) = 15, v_2^\circ(C) = 10, v_3^\circ(ABC) = 14, v_4^\circ(A) = 11$ and $v_5^\circ(BC) = 11$. The auction assigns adjusted values $15/2, 10, 14/3, 11$ and $11/2$ to each bid (i.e. the per-item bid values), and places the bids in decreasing order as bids $4, 2, 1, 5$ and 3. Agent 4 is allocated A, agent 2 is allocated C, and then no other agents receive any goods because one or more items in their interesting bundles are already allocated. The allocation is $(\emptyset, C, \emptyset, A, \emptyset)$. (Note that this is not the efficient allocation.) Agent 2's payment is determined by considering the values of bids that it displaces, namely the bids from agents 3 and 5. (It must simply remain ranked above agents 3 and 5 to remain a winner.) Considering agent 3, the constraint is $v_2' \geq 1/3(5 + 0.2(10 + 0.5v_2') + 0.6v_2')$ (where the factor $1/3$ is included because agent 3 bids on 3 items, and thus this reduces its rank), which gives $v_2' \geq 70/23$. Considering agent 5, the constraint is $v_2' \geq 1/2(1+2/3(10+0.5v_2'))$ (factor $1/2$ because agent 5 bids on 2 items), which reduces to $v_2' \geq 23/5$. The latter constraint is binding, and agent 2's payment is $p_2 = 23/5 = 4.6$. Agent 4 displaces bids from agents 1 and 3, and its payment is computed as $\min v_4'$ s.t. $v_4' \geq \max\{15/2, 14/3\} = 15/2 = 7.5$.[7]

6. CONCLUSIONS AND FUTURE WORK

We have instantiated the contingent bid model of Dasgupta and Maskin [3] to provide a concrete protocol for interdependent value auctions. A simple linear bidding language is demonstrated to satisfy the technical conditions required for the efficiency of a contingent-bid second-price auction. Seeking good revenue properties, we also illustrated protocols with better expected revenue.

In addition, we described a protocol for interdependent-value combinatorial auctions (CAs) with single-minded bidders. Thus, we find a positive counterpoint to the negative general conclusions in the literature on interdependent value auctions with multi-dimensional private information [8, 9].[8] An important future direction is to study other models that provide structure to value interdependencies and allow for similarly positive results.

In principle, a more general pricing method described in Dasgupta and Maskin [3] can be used to design truthful auctions in combinatorial auctions without the restriction to single-minded bidders. (As long as the generalized SCC condition continues to hold, and with single-dimensional private information.) However, we wish to draw attention to what we believe to be a severe practical limitation of the generalized method:

*in addition to reporting contingent bids, agents are required to report the residual domain of valuations that it believes to be possible for **other** agents, given its own signal and given its knowledge about the functional form of the valuations of other agents.*

This requirement, which is also discussed in Perry and Reny [16], is used to ensure that agent-independent payments can be computed that are consistent (in a utility-maximizing sense) with the efficient allocation. Requiring that every bidder reports exact and complete information on the residual valuation domains of other bidders seems untenable in practical systems. It seems likely to place a large burden on participants. We think future work should seek mechanisms for interdependent values that are (necessarily) inefficient when agents do not reveal adequate information, but fully efficient in the complete-revelation limit.

[7] It is instructive to consider the effect of agent 1 increasing its value in the greedy auction. Recall that this is where we observed the failure of generalized SCC in the proposed efficient auction. In order to win in the greedy auction, agent 1 must rank above agents 3 and 4, of which agent 4 dominates. To rank above agent 4, we need $v_1'/2 \geq 1 + 2/3v_1'$, which is impossible for $v_i' \geq 0$. So, agent 1 can never win in this auction given the contingent bids of the other agents.

[8] See also Bikhchandani [1] for some additional possibility results.

Acknowledgments

Parkes is supported in part by National Science Foundation grants IIS-0238147 and an Alfred P. Sloan Foundation award.

7. REFERENCES

[1] S. Bikhchandani. The limits of ex-post implementation revisited. *Working Paper*, UCLA, 2005.

[2] J. Cremer and R. P. McLean. Full extraction of surplus in bayesian and dominant strategy aucitons. *Econometrica*, 56:1247–1257, 1988.

[3] P. Dasgupta and E. Maskin. Efficient auctions. *The Quarterly Journal of Economics*, CXV:341–388, 2000.

[4] R. K. Dash, A. Rogers, and N. R. Jennings. A mechanism for multiple goods and interdependent valuations. In *Proc. 6th Int. Workshop on Agent-Mediated E-Commerce*, pages 197–210, 2004.

[5] T. Ito, M. Yokoo, and S. Matsubara. Designing an auction protocol under asymmetric information on nature's selection. In *Proc. of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS02)*, pages 61–68, 2002.

[6] T. Ito, M. Yokoo, and S. Matsubara. Towards a combinatorial auction protocol among experts and amateurs: The case of single-skilled experts. In *Proc. of the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS03)*, pages 481–488, 2003.

[7] T. Ito, M. Yokoo, and S. Matsubara. A combinatorial auction protocol among versatile experts and amateurs. In *Proc. of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS04)*, pages 378–385, 2004.

[8] P. Jehiel and B. Moldovanu. Efficient design with interdependent valuations. *Econometrica*, 69(5):1237–1259, 2001.

[9] P. Jehiel, M. M. ter Vehn, B. Moldovanu, and W. R. Zame. The limits of ex-post implementation. *Working Paper*, 2005.

[10] V. Krishna. *Auction Theory*. Academic Press, 2002.

[11] D. Lehmann, L. I. O'Callaghan, and Y. Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. *Journal of the ACM*, 49:577–602, 2002.

[12] A. Likhodedov and T. Sandholm. Methods for boosting revenue in combinatorial auctions. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 232–237, 2004.

[13] A. Likhodedov and T. Sandholm. Approximating revenue-maximizing combinatorial auctions. In *Proc. of the National Conference on Artificial Intelligence*, pages 267–274, 2005.

[14] E. Maskin. *Private communication*, October 7 2005.

[15] R. B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6:58–73, 1981.

[16] M. Perry and P. J. Reny. An efficient auction. *Econometrica*, pages 1199–1212, 2002.

[17] R. Vohra. *Advanced Mathematical Economics*. Routledge, 2005.

Designing a Successful Trading Agent for Supply Chain Management

Minghua He, Alex Rogers, Xudong Luo and Nicholas R. Jennings
School of Electronics and Computer Science, University of Southampton, U.K.
{mh,acr,xl,nrj}@ecs.soton.ac.uk

ABSTRACT

This paper describes the design and evaluation of SouthamptonSCM, the runner-up in the 2005 International Trading Agent Supply Chain Management Competition (TAC SCM). In particular, we focus on the way in which our agent purchases components using a mixed procurement strategy (combining long and short term planning) and how it sets its prices according to the prevailing market situation and its own inventory level (because this adaptivity and flexibility are key to its success). We analyse our buying and selling strategies in the actual competition and in controlled experiments. Through this evaluation, we show that SouthamptonSCM performs well across a broad range of environments.

1. INTRODUCTION

Effective supply chain management is vital to today's economy and increasingly organisations are looking to the agility and automation afforded by agent-based approaches [4]. To this end, the International Trading Agents Competition for Supply Chain Management (`http://www.sics.se/tac`) (TAC SCM) provides a benchmarking environment for testing and evaluating agents in a challenging and realistic setting. Specifically, in the TAC SCM scenario, the aim is to design an agent representing a computer manufacturer, that competes with other agents for computer components (from the supplier side) and computer orders (from the customer side). These agents are also responsible for managing their limited production capacities, in order to deliver the products to customers before the due date they agreed in the order.

In more detail, a game includes six agents (competition entrants) that compete with one another to procure raw components and fulfil customer orders for assembled PCs. Each agent can produce 16 distinct computer types from four components: CPU, motherboard, memory and hard disk (*e.g.* a PC with a 2GHz IMD processor with 1GB memory and a 300GB hard drive). Consequently, on each of the 220 simulation days of the game, agents receive new request for quotes (RFQs) and actual orders (if offers they have previously sent win) from the customers. From the supplier side, an agent receives offers to deliver particular quantities and types of components at particular prices in response to the RFQs that it sent on the previous day. Thus, in each day of the game (lasting 15 seconds), the agent must decide on the following: (i) which new supplier RFQs to send and which supplier's offers to accept; (ii) which customer RFQs to respond to, and at what price; and (iii) how to schedule the production of PCs given the availability of components, the limited capacity of the factory and the delivery deadlines of pending orders. Now, an agent spends money on buying the components, paying for the storage of both components and PCs, paying penalties if it defaults on a promised delivery date and paying overdraft penalties if it is in debt to the bank. The agent earns money by selling PCs and receives interest from the bank if its balance is positive. The success of an agent is measured in terms of its profit (*i.e.*, its bank balance at the end of the game).

Against this background, we present the design and evaluation of our agent (SouthamptonSCM) that was the runner-up in the 2005 competition (out of 32 initial participants). The main contributions of this work are as follows. First, we develop techniques to enable the agent to adapt its price setting to the prevailing market situation, its own internal state (inventory level) and the time that has elapsed. At their core, these techniques employ fuzzy reasoning in order to allow the agent to adapt its prices daily so that it can fully exploit its production capacity, while still maximising its revenue by selling at appropriate prices. Second, we develop a mixed procurement strategy that enables the agent to buy components according to what is needed in the near future (by predicting customer demand) and also to order a small quantity of components far in advance (at a lower price) to meet the minimum daily usage. This strategy balances flexibility in dealing with the dynamically changing market, with the need to procure components at low cost whilst also minimising inventory stock. This is especially useful when the demand is increasing and the agents are thus competing for the components, since it is this case that causes component prices to increase. To evaluate the effectiveness of these developments, we analyse the performance of our agent in the actual competition and also in more systematic controlled experiments.

The remainder of the paper is organized as follows. Section 2 presents our agent. In section 3 we evaluate it. Finally, in section 4 we conclude and discuss future work.

2. SouthamptonSCM

SouthamptonSCM is composed of three sub-agents (see figure 1).[1] The *customer agent* receives RFQs from the customers and decides what offers to respond with. It also communicates with the factory agent to obtain the updated inventory levels and to send the

[1]Here we use the notion of sub-agents (instead of modules) because each of them can autonomously communicate with the suppliers and customers to get the RFQs, can send offers and obtain orders, and can decide how to respond to this information.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

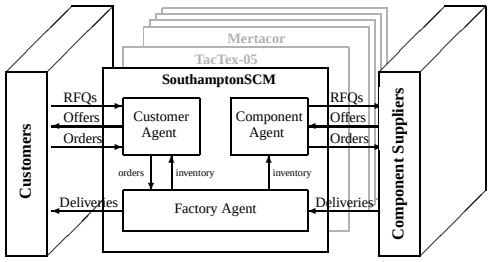


Figure 1: SouthamptonSCM agent architecture.

relevant customer PC orders. The *component agent* decides which RFQs and which orders to send to which suppliers. The *factory agent* receives the supplies delivered from the suppliers, decides based on the available resources (computer components and factory cycles) in what order the customer orders should be produced, and determines the schedules for delivering the finished PCs to the customers. We now deal, in turn, with each sub-agent.

2.1 The Customer Agent

The customer agent is the key component in our architecture (because we believe that offering the appropriate price at the right time is vital for success). If the price is too low, the agent will receive a low profit and if it is too high it will fail to win any orders (because customers always choose the lowest offer price among those they receive). Given this, the key challenges are to determine which customer RFQs to bid for and at what price. To achieve this, we use fuzzy reasoning to determine how to set prices according to the agent's inventory level, the market demand and the time into the game. Moreover, the parameters involved in the fuzzy rules can be adapted according to the quantity of the received and expected customer orders so as to maximise the factory utilisation.

The customer agent we used in last year's competition has previously been described in [3] and in several respects this year's design is similar. However, due in part to the changes in the rules of this year's competition and also because SouthamptonSCM 2005 is built upon last year's experiences, there are also a number of important differences. Most importantly:

- When prioritising customer RFQs, SouthamptonSCM 2005 considers the potential profit that each RFQ will generate (SouthamptonSCM 2004 did not consider the cost of the components). Moreover, when checking on the availability of the components required to assemble the requested finished product, unlike SouthamptonSCM 2004 that just considered the quantity of components held as inventory, the 2005 agent also considers pending component orders that are due to be delivered before the due date of the RFQ (see Section 2.1.1).

- SouthamptonSCM 2005 also calculates the offer prices of finished products differently. Although it uses the same basic technique as the 2004 agent, the 2005 agent has two rule bases for calculating prices: one for when it is close to the end of the game and one for other days in the game (see Section 2.1.2). The 2004 agent, also used two rule bases but these were: one for the end stage of the game and one for when it was expecting a large component delivery.[2]

[2]This rule was motivated by the use of a day 0 procurement strategy that ordered the majority of the components needed for the entire game on day 0. Since changes in the TAC SCM game specification for the 2005 competition removed the benefit of this procurement strategy, the customer agent had to be updated.

We now deal, in turn, with each of the these differences.

2.1.1 Choosing Customer RFQs.

The customer agent receives customer RFQs requesting a quantity of a particular PC for delivery on a specified day. When selecting which RFQs to respond to, SouthamptonSCM rates them according to the potential profit that they may bring and according to the inventory it holds. The latter inventory driven strategy offers customers PCs according to what is currently available and also what can be produced given the delivery date of pending components orders. This consideration of promised component deliveries is new and is included this year since changes to the supplier model mean that there are no longer significant delivery delays in these component orders[3] [1].

In more detail, suppose a customer RFQ is represented as a tuple $(i,\ q,\ p_{res},\ c_{penalty},\ d_{due})$, where $i \in \{1,\cdots,16\}$ is the type of PC the customer wants, $q > 0$ is the quantity, $p_{res} > 0$ is the reservation price (maximum it will pay), $c_{penalty} > 0$ is the fine paid if the computers are not delivered on time, and d_{due} is the desired delivery date. On each day, the customer agent receives a bundle of such RFQs and sorts them in the decreasing order of the profit margin of the type of PC requested:

$$p_{res} - p_{base} - \frac{c_{penalty}}{q},$$

where p_{base} is the cost of buying components (sum of the buying price for each component). The intuition is that the agent will first serve customers with high profit margins and low penalties. This is because the higher the p_{res}, the more profit will be made (compared to selling the same product to a customer with a low p_{res}). At the same time, the agent also wants to avoid getting high penalty orders because of the inherent uncertainties that exist in the game.

For each RFQ in the list, the agent first checks whether it can be supplied from its stock of finished PCs (see Section 2.3). If it can, the corresponding PC inventory is decreased. Otherwise, the agent checks whether it holds sufficient components within both its current inventory and its expected delivery queue (*i.e.* components that have been ordered, but have not yet arrived), and also whether it has a sufficient remaining production capacity to manufacture the required PCs. If it does, the agent decreases its available component inventory.

2.1.2 Fuzzy Rules for Calculating Offer Prices

For each type of PC, the agent uses a fuzzy reasoning inference mechanism which is based on the standard Sugeno controller [6] to calculate the offer price of components. Full details of this process are described in [3], however, here we present the rule bases as used in the 2005 agent. Now, due to the constraint that the TAC SCM game is played over a fixed number of days, we use two rule bases: one for the majority of the game days where the aim is to sell finished products at maximum profit, and one for the last stages of the game where the aim is to 'off-load' any remaining finished product and component inventory. Splitting the rule bases enables us to reduce their complexity and thus maintain their interpretability.

In more detail, in the first rule base, for each PC type the agent considers the current customer demand state and the inventory level of the components that make up that particular PC type. There are twelve such rules within this base and the following three are representative examples:

$\mathcal{R}_1$: if D is *high* and I is *low* then r_1 is *very-big*

[3]These changes were introduced to remove the incentive to adopt a day 0 procurement strategy.

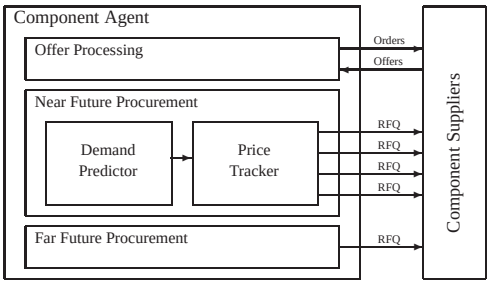


Figure 2: Overview of the component agent.

$\mathcal{R}_2$**:** if D is *medium* and I is *high* then r_2 is *very-medium*

$\mathcal{R}_3$**:** if D is *low* and I is *high* then r_3 is *small*

In these rules, the customer demand (D) is expressed in the fuzzy linguistic terms *high*, *medium*, and *low*, and the inventory level (I) in the terms *high*, *medium*, and *low*. The output of each rule is a fuzzy variable (r_1 to r_3 in these examples), and thus, the outputs of all the rules are combined by the Sugeno controller to give a single scalar result, r, that represents the *price adjustment factor* used to generate a reference price [2] for each type of PC. When r is high, the reference price approaches (or is even higher than) the highest transaction price recorded on the previous day (this information is made available to the agents through the daily market report), whilst when r is low, the reference price approaches the lowest transaction price. Given the reference price of each type of PC, an offer price is calculated by modifying the reference price by a factor related to the requested due date. The intuition here is that the sooner the due date, the higher the offered price. Thus, rule $\mathcal{R}_1$ captures the fact that if the customer demand for this kind of PC is high and the agent has a low inventory in stock, then the offer price should be very high since here there is an opportunity to maximise the agent's profit.

The second rule base is employed for the days near the end of the game (last 20 in this case) and it considers the current inventory level, the customer demand and also how close it is to the end of the game. There are 22 such rules that capture different combinations of situations of these three variables, and, three such rules in this rule base, are:

$\mathcal{R}_1'$**:** if D is *high* and I is *high* and E is far then r_1' is *big*

$\mathcal{R}_2'$**:** if D is *high* and I is *high* and E is *medium* then r_2' is *medium*

$\mathcal{R}_3'$**:** if D is *low* and I is *high* and E is *close* then r_3' is *very-small*

Here, the days to the end of the game (E) are expressed in the fuzzy linguistic terms: *far*, *medium*, and *close*. Thus, for example, rule $\mathcal{R}_3'$ captures the fact that if there is little demand for a particular type of PC, the agent has a high inventory, and there is a little time until the end of the game, then the price adjustment factor should be very small (thus ensuring a low offer price and hence reducing the risk of being left with inventory at the end of the game).

2.2 The Component Agent

The component agent is responsible for dealing with the component suppliers and aims to ensure that there are always sufficient components in stock to address the customers' changing demand for finished products. In doing so, it addresses a challenge that is common to all supply chains facing dynamically changing customer demand. That is, it must procure components at a low cost, whilst simultaneously maintaining a minimal component inventory in order to reduce the daily storage cost and also the possibility of being left with redundant stock if customer demand changes.

Now, the 2005 component agent uses a completely different strategy to that of last year. In 2004, it procured most of its components by placing a large order on day 0 (due to the fact that the suppliers have their full capacity available on day 0 and thus offer the lowest possible price [3]). However, in 2005, the game rules were changed in order to reduce the lottery effect in the supplier interactions, to discourage the use of a day 0 procurement strategy [7], and to deter agents from requesting large quantities of components without intending to order them [1]. This means that the component agent used in 2004 is no longer effective, and thus, we present our updated component agent here.

In addressing the procurement challenge this year, the agent faces a common dilemma concerning the lead time of component orders. More specifically, ordering components with a long lead time (*i.e.* for delivery a long time into the future) generally leads to low procurement costs (since the suppliers have adequate spare capacity over this long period). However, the agent is then exposed to the risk that the customer demand may change, leaving it with excessive component stock. Conversely, ordering components with a short lead time allows the agent to effectively track the changing customer demand and maintain minimum stock inventory. However, due to the short lead time, the suppliers generally have less flexibility in scheduling the production of these orders (since the order is competing with those from other agents for the limited production capacity available), and thus, procurement costs are high.

Now, to tackle this dilemma, SouthamptonSCM employs a mixed procurement strategy; dividing the remaining days of the game into two classes: near future and far future (see figure 2). For the days in the far future, the agent periodically orders a small fixed quantity of components, so as to ensure that a minimum quantity of stock is available in order to keep the factory running (see section 2.2.1 for details). For the days in the near future (which is 35 days in this case[4]), the agent makes a short term prediction of the future customer demand, and orders sufficient components to address this demand (see section 2.2.2 for details). In addition, within this near future, the agent also *tests* the market in order to track current prices. This price information is used to choose the appropriate lead time and reserve price of the RFQ sent to the suppliers. In more detail, since each agent can send a maximum of five RFQs for a particular component to each supplier each day, the agent uses four RFQs for the near future procurement (*i.e.* to actually order components with short lead time and also to track the component prices). The remaining RFQ is used to buy components for the far future.

2.2.1 Far Future Procurement

Of the two strategies, the far future procurement one is the simpler. For this, the agent assumes that in the far future, there will be a daily minimum need for 30 CPU and 60 other components (this ratio is required since there are four different types of CPU, but other components are only available in two different types), and thus checks whether there is sufficient current and pending component inventory to meet this need. If not, it submits an RFQ for a fixed amount to the relevant supplier, requesting delivery on the date at which the predicted inventory falls below the daily minimum need (up to a maximum of 120 days into the future).

2.2.2 Near Future Procurement

The near future procurement strategy is more complex. It con-

[4]In our experimental evaluation, we found that a larger number than 35 led to significantly more redundant components in stock due to the difficulty of predicting the fluctuating customer demand, whilst a smaller amount was not able to keep the stock above a minimum usable threshold.

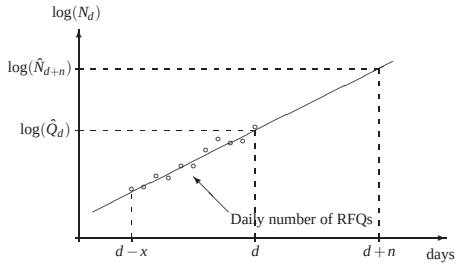


Figure 3: Linear regression process by which Southmapton-SCM predicts the future customer demand.

sists of two elements, a daily demand predictor that predicts the future demand of components, and a market tracker that generates the RFQs to be sent to the suppliers, both to order actual components required and to test the market to discern the most profitable order lead time and set appropriate reserve price.

Demand Predictor

As described above, the agent buys components for the near future based on a prediction of customer demand. Now, according to the game specification, the number of RFQs that each agent receives from the customers is described by three independent random walks; one for each market segment (finished product PCs are classified into three such segments: high, mid and low range). In more detail, the number of RFQs that an agent receives, within a single market segment, on day d is denoted by N_d, and is drawn from a Poisson distribution whose expected value is given by the parameter, Q_d. Thus, for each market segment:

$$N_d = \text{Poisson}\left(Q_d\right) \tag{1}$$

Again, according the game specification, the value of Q_d is updated each each day between minimum and maximum limits according to a multiplicative trend parameter, τ_d, and thus:

$$Q_{d+1} = \min\left(Q_{\max}, \max\left(Q_{\min}, \tau_d Q_d\right)\right) \tag{2}$$

where the trend, τ_d, is itself updated each day, again between minimum and maximum limits, according to a random walk:

$$\tau_{d+1} = \max\left(\tau_{\min}, \min\left(\tau_{\max}, \tau_d + \text{random}\left(-0.01, 0.01\right)\right)\right) \tag{3}$$

Now, since τ_d varies relatively slowly (and can thus be assumed to be constant over short periods of time), the agent can estimate the values of Q_d and τ_d (these estimated values are denoted by $\hat{Q}_d$ and $\hat{\tau}_d$) by recording, for the previous x days, the number of RFQs received within each market segment (*i.e.* $N_{d-x} \ldots N_d$), and then performing a linear regression of the logarithm of these values against d (we take the logarithm since the trend is multiplicative). Having found these values, the agent can then extrapolate, and predict the number of RFQs that will be received on day $d+n$, and thus:

$$\hat{N}_{d+n} = \min\left(Q_{\max}, \max\left(Q_{\min}, \left(\hat{\tau}_d\right)^n \hat{Q}_d\right)\right) \tag{4}$$

This process is shown diagrammatically in figure 3. In trials, we found that by using data from the previous 10 days, and predicting a value for 10 days into the future (*i.e.* $x = 10$ and $n = 10$), we obtained a good estimate for the average demand within the near future. This process gave significantly better results than the simple alternative of using the number of RFQs received on the current day, as an estimate for future days.

Having predicted the number of RFQs that will be received, within each market segment, on each day within the near future, the agent then calculates the expected daily usage of each component type (D_{id}). This is calculated by weighting the maximum number of PCs that can be manufactured on any day, D_{total},[5] by a factor that incorporates both the expected number of RFQs that will be received within each market segment (*i.e.* $\hat{N}^{low}_{d+n}$, $\hat{N}^{medium}_{d+n}$ and $\hat{N}^{high}_{d+n}$ as described above) and a weighting that describes how many of the PC types within any market segment actually contain the specific component being considered (for any component this weighting is described by r^{low}_{id}, r^{medium}_{id} and r^{high}_{id} and, for example, for component id of 200 these values are 0.4, 0.5 and 0.6 respectively). Thus:

$$D_{id} = \frac{D_{total}\left[r^{low}_{id}\hat{N}^{low}_{d+n} + r^{medium}_{id}\hat{N}^{medium}_{d+n} + r^{high}_{id}\hat{N}^{high}_{d+n}\right]}{\hat{N}^{low}_{d+n} + \hat{N}^{medium}_{d+n} + \hat{N}^{high}_{d+n}} \tag{5}$$

Given D_{id}, the current available stock of each component, and the expected deliveries from the suppliers from previous orders, the agent checks whether stock levels in the near future will fall below a minimum threshold. This threshold represents a buffer stock level, and guided by the principle that towards the end of the game the inventory level should decrease, it starts from 0 on day 0, increases gradually to a fixed number of 300 CPU and 600 other components over the first 30 days, and subsequently decreases to zero again over the last 10 days. If stock levels are predicted to fall below the buffer level within the near future, the agent identifies the quantity required (with a maximum order size of 80 components) and the latest date that this quantity should arrive by. This information is passed to the price tracker, which actually generates the RFQs that are sent to the suppliers.

Price Tracker

The price tracker acts to maintain an estimate of the current market price of the components. Due to the behaviour of the competing agents, this market price depends on the due date with which components are requested. For example, if the competing agents are ordering components with very short lead times, then the supplier will have little spare capacity, and thus, the corresponding offer prices that the agent receives will be greater than those of orders with long lead times. Every day the the price tracker generates four RFQs which are sent to each supplier. The due dates of these RFQs are distributed over three regions within the near future (*i.e.* less than 10, 10 to 20, and more than 20 days into the future) and the offer price that the agent receives in response to these RFQs enables it to track the changing market price of the component and also to identify the order lead times that will result in the minimum procurement costs.

Now, in more detail, the four RFQs generated by the price tracker fall into two different classes: those that actually order the components requested by the demand predictor, and those that have zero quantity (and are simply used to track the market price). On any day, up to four of these RFQs will be used to actually order components (depending on the requirements of the demand predictor), and within the constraints of the delivery date requested by the demand predictor, the lead time of these RFQs will be determined by the results of the market testing. This market testing is also used to set an appropriate reserve price for the RFQ.[6]

[5] In this case, $D_{total} = 364$, since, given the factory production capacity of 2000 cycles per day and an average production cost for a PC of 5.5 cycles, approximately $2000/5.5 = 364$ PCs can be manufactured per day.

[6] Note, the setting of this reserve price is particularly important in the case of our agent. This is because the suppliers maintain a record of the reputation of each agent that is dependent on the ratio of offers that are sent compared to the actual number of orders that

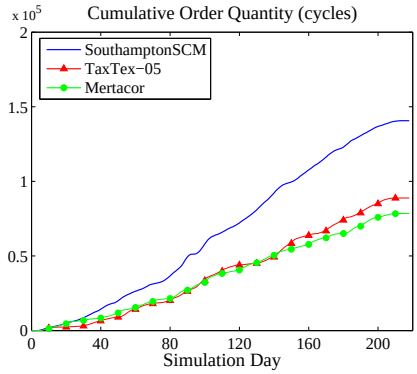


Figure 4: Cumulative order quantities in game tac4-4253.

2.3 The Factory Agent

One of the main challenges for the factory agent is scheduling what to produce and when to produce it (*i.e.*, how to allocate supply resources and factory time). The strategy we use for SouthamptonSCM 2005 agent is broadly similar with that of last year's agent. This strategy involves manufacturing PCs according to customer orders and satisfying orders with an earlier delivery date (see [3] for more detail). Now, since the computers stored in the factory will be charged storage cost, each order will be delivered as soon as it is filled. The agent builds the PCs according to the customers' orders it has obtained (which has the advantage of ensuring that the factory always produces the needed computers on time). However, if on any day, there are still free factory assembling cycles available, and the numbers of finished PCs in stock are below a certain threshold, then the agent produces additional PCs of each kind uniformly (subject to the availability of components) in order to maximise the factory utilisation. It is critical that this threshold is set appropriately; a high threshold will lead to excessive finished PC inventory, which may be hard to sell if demand is low. In 2005, this threshold is set to 80, and this is shown to work well in both low and high demand markets.

3. EVALUATION

Our evaluation is composed of three components: (i) the results from the 2005 competition; (ii) our post-hoc analysis of some typical games in the actual competition; and (iii) a systematic range of controlled experiments.

3.1 Competition Results

TAC SCM consists of a preliminary round (predominantly used for practice and fine tuning) followed by a seeding round that determines the groupings for the final rounds. The top 24 agents from the seeding round are organised into 4 groups that represent each of the four quarter-finals. The top three teams of quarter-finals 1 and 3 are entered into semi-final 1, and similarly, the first 3 teams from quarter-finals 2 and 4 are entered into semi-final 2. Finally, the first 3 teams in both semi-finals are entered into the final round.

In the 2005 competition, there were 32 entries and SouthamptonSCM was placed seventh among all the participants in the seeding round and entered group B for the quarter-final. Here, it had the second highest score, in its semi-final it had the highest score, and in the final, it was the runner-up (to TacTex-05 [5]).[7]

3.2 Competition Game Analysis

In this section, we analyse the performance of SouthamptonSCM in terms of the efficiency of both its buying and selling strategies. That is, we consider the effectiveness of the customer pricing model and the mixed component procurement strategy. Thus we select a particular game from the final (game tac4-4253) and analyse it in more detail.[8]

3.2.1 Analysing the Selling Strategy

To complement and better understand the competition result and to evaluate the effectiveness of our pricing model (since we attribute our success in selling PCs to the pricing technique we employ), we conducted a post hoc analysis of a game from the final. In order to reduce the complexity of this analysis, we consider just the top three performing agents within this game (*i.e.* in this case SouthamptonSCM, TacTex-05 and Mertacor (who also happen to be the top 3 agents in the overall competition)). Now, an essential factor in the performance of the trading agent is the ability to respond to customer RFQs with competitive offer prices. Thus, we analyse the game data and we consider just those RFQs for which all three of the above agents responded with offers. We then consider which of the three agents actually won these orders (noting that the order is always assigned to the agent that offers the lowest price). In figure 4 we show the resulting order quantities (expressed as factory production cycles averaged over all products), that were won by each of the three agents. Significantly, the SouthamptonSCM agent wins a far larger proportion of the orders than the other two agents. This is consistent with the overall game analysis, where the orders these three agents delivered in the game are: 7331 for SouthamptonSCM, 6742 for TacTex-05, and 5588 for Mertacor.

Now, there is little advantage to win these orders, if the offer price was excessively low. Thus, in figure 5, we compare the average offer prices for those orders that were won by each of the three agents. Specifically, in figure 5a we show the average offer prices of the orders that were actually won by SouthamptonSCM, in figure 5b those won by TacTex-05, and in figure 5c those won by Mertacor. In each case, as expected, we see that the agent that won the orders actually submitted offers with the lowest average price. However, despite winning the majority of the orders, the prices offered by SouthamptonSCM were not significantly lower than those of the other agents. Indeed, the opposite is true. The margin by which SouthamptonSCM undercut the other agents is actually smaller than the margin with which the other agents undercut SouthamptonSCM. For example, in this game, on average SouthamptonSCM undercut Mertacor's prices by 3.0% and TaxTex-05's prices by 4.1%. Conversely, on average, TacTex-05 undercut SouthamptonSCM's prices by 6.5% and Mertacor undercut the prices of SouthamptonSCM by 5.6%.

Thus, the selling strategy of our agent is able to accurately predict the market price of the finished products, and is able to undercut the competing agents by a small margin. In so doing, our agent wins a greater proportion of the orders than the other agents, but

are placed. Agents with higher reputation are given priority treatment and receive lower offer prices. Thus SouthamptonSCM maintains a perfect reputation record by adopting the policy of accepting all offers that it receives (this is dealt with by the *Offer Processing* in figure 2). Thus, it is essential that the agent sets the reserve price of RFQs to an appropriate value, to avoid being obliged to accept an excessively expensive offer.

[7]TacTex-05 is believed to have won by learning between games, and improving its buying strategy over the course of the final.

[8]In order to analyse a representative game, we select randomly from those games in the final that did not experience particurly low customer demand where all players ended the game year in debt. As such, we choose the first game of the final on server tac4 (*i.e.* game tac4-4253).

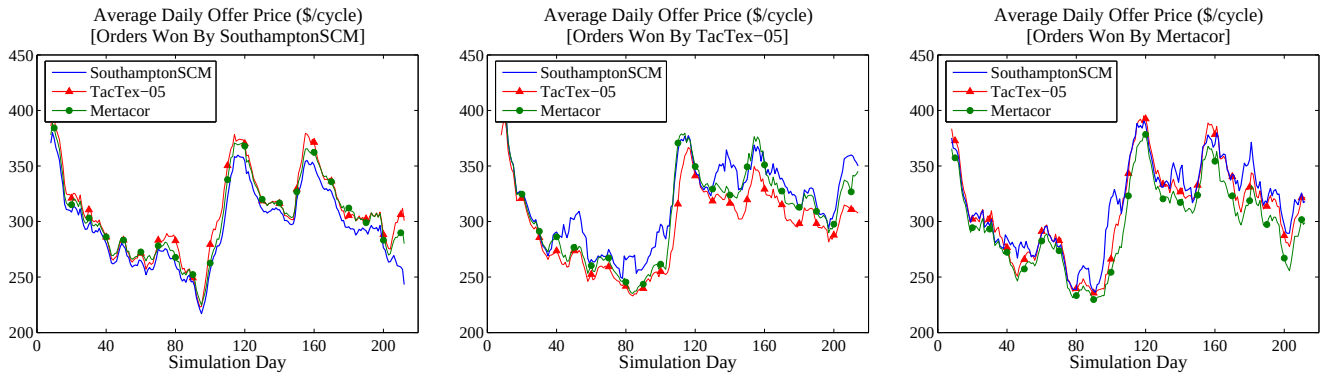


Figure 5: Daily offer prices for orders won by (a) SouthamptonSCM, (b) TaxTex-05 and (c) Mertacor in game tac4-4253.

does not unnecessarily reduce the price of its offers. Since the ultimate profitability of the agent depends on both the price and the quantity, SouthamptonSCM thus has the highest revenue.

3.2.2 Analysing the Buying Strategy

To investigate the effectiveness of its mixed procurement strategy, we compare, on a daily basis, the component ordering policies adopted by the three top performing agents (*i.e.* SouthamptonSCM, TacTex-05 and Mertacor). Now, in order to make this comparison, we can not average over all components as we did in the previous section. The large range in the base price of the components makes such averages meaningless. Thus, we select one component for this comparison, and here we use the IMD 2GHz CPU, since this is not the most expensive component (the higher performing 5GHz CPU has a higher base price), and yet, it largely determines the cost of the finished product PC. Whilst space does not allow us to present all the results, we note that the qualitative results that we present are representative for all the other components.

Now, in figure 6a, we show the daily average unit cost of the CPUs that were procured by each of the agents. As can be seen, SouthamptonSCM paid a similar price to TacTex-05 in the first half of the game. However, after that, SouthamptonSCM procures components at a much cheaper price. In figure 6b, we show the cumulative quantity of CPUs that were procured by each of the agents. Now, in this game, the demand for the IMD 2GHz CPU increased in the second half of the game, and thus we observe that all three agents purchase CPUs at a greater rate in the second half of the game (*i.e.* the gradient of the cumulative plot increases). Indeed, SouthamptonSCM and TacTex-05 end up having obtained an identical quantity of CPU, with Mertacor buying significantly less. However, since, on average, SouthamptonSCM is able to procure these CPUs at a lower price, when we compare the cumulative cost of this procurement, as shown in figure 6c, we see that SouthamptonSCM has paid less for these CPUs in total than TecTex-05 although they both have obtained identical quantity of CPU.[9]

We can see the reason for this difference in procurement costs by considering the lead time of these component orders. Thus, figures 6d, 6e and 6f, show a histogram of the lead time of component orders that arrive during each simulation day (in this figure larger orders are shown with darker shading). In figure 6d, we can see the result of the mixed procurement strategy that SouthamptonSCM adopts. Here, it is clear that each case sees a combination of orders that were placed with a long lead time (up to to 120 days during the second half of the game) and a short lead time of 5-20 days (corresponding to the near and far future procurement strategies). Note that TacTex-05 uses a predominantly short lead time strategy, whilst Mertacor uses a long lead time strategy.

Thus, during the first half of the game, orders with a short lead time result in procurement at a low cost. Here we see that TaxTec-05 procures components at less cost than Mertacor. However, in the second half of the game, the opposite is true and orders with a short lead time are more expensive. Here in this half of the game, SouthamptonSCM procures components at less cost than Mertacor. However, overall, the mixed procurement strategy of SouthamptonSCM allows it to achieve the best result in both halves of the game; in the first half it achieves similar low prices to TacTex-05, whilst in the second half it achieves prices significantly lower than both the other agents. Thus, we believe that the mixed procurement strategy incorporating both long and short lead times with market price tracking, enables SouthamptonSCM to adapt to the behaviour of the competing agents and buy components at a cheap price.

3.3 Controlled Experiments

To evaluate the performance of our agent in a more systematic fashion than is possible within the actual competition, we ran a series of controlled experiments. In previous work we have already analysed our pricing model through controlled experiments and showed that the way SouthamptonSCM sets its offering prices is significantly better than the benchmark strategies we considered [3]. Thus, here we focus on evaluating the buying strategy.

As mentioned earlier, we attribute the success of our agent to the mixed procurement strategy that it uses to purchase components at minimum cost. Thus, here we analyse how the buying behaviour works when compared to two other common alternatives. These two strategies are identical to SouthamptonSCM, except for the method they use to procure components (*i.e.* the way in which they schedule factory production and offer finished products to the customers is unchanged). The three types of agents we use are:

- **Mixed agent (MIX-agent)**. This agent uses our mixed procurement strategy (described in Section 2.2). In this case, the daily minimum need for CPU in the far future is set to 30 and the corresponding value for other components, is set to 60.
- **Long-term planning agent (LTP-agent)**. Compared with the mixed strategy, the LTP-agent still uses one RFQ each

[9]The difference in costs in figure 6 is small, but reflects the competition's competitive nature (note that 6b and 6c are cumulative plots). The difference represents a $6-7\%$ saving in material costs, whilst the difference in overall margin between SouthamptonSCM and TacTex-05 was only 1%.

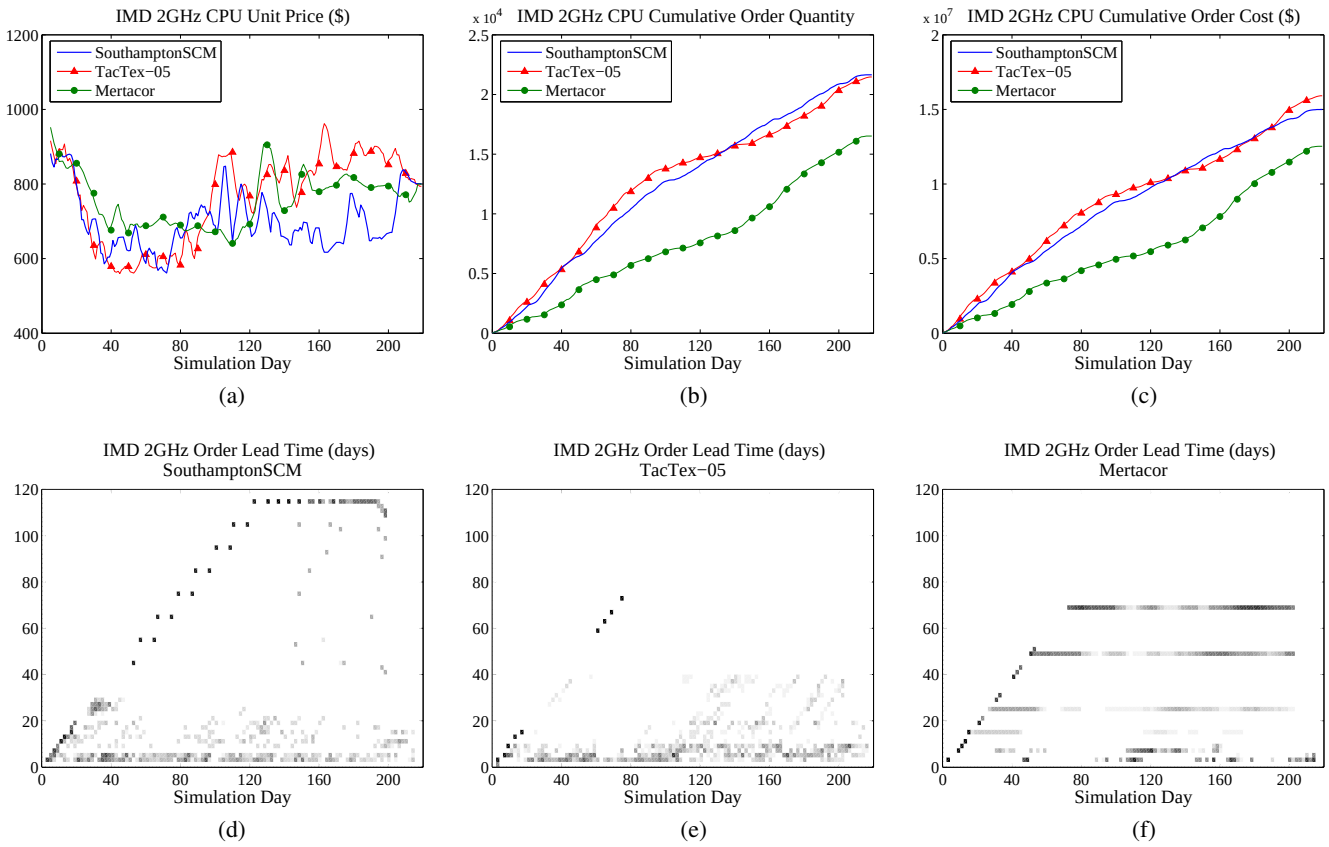


Figure 6: Comparison of component order prices, quantity and daily lead times for IMD 2GHz CPU orders in game tac4-4253.

day for far future procurement, but the quantity that is requested is greater. Thus this agent attempts to guarantee that it can obtain the components cheaply (by ordering with long lead times), but runs the risk that some components may ultimately turn out to be unusable (due to a downturn in customer demand). Specifically, the minimum need for CPU in the far future is set to 60 and the corresponding value for other components, is set to 120.

- **Short-term planning agent (STP-agent)**. This agent does not buy any components for the far future, and thus, only uses the near future procurement strategy. This strategy is able to predict and adapt to changes in customer demand. However, it runs the risk of paying a high price for some components when the supplier has little spare capacity.

Besides these three kinds of agents, the other competing participants within the game are the dummy agents provided by the organisers. These use a naïve build-to-order strategy that procures components with a short lead time (typically several days). Thus this agent risks paying high component prices, particularly so since it also sets the reserve price of RFQs that it submits to 0, and thus, to maintain its reputation, it may have to accept high offer prices.

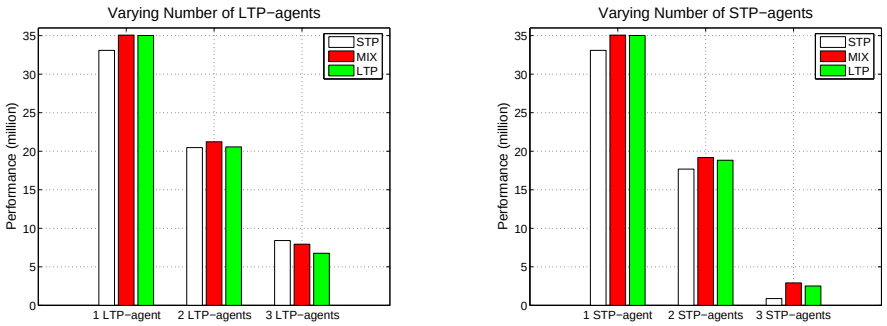
Given this background, five groups of experiments were conducted to examine the performance of each kind of agent in various situations. The number of MIX-agent, LTP-agent and STP-agent were: in setting A (1, 1, 1), in setting B (1, 2, 1), in setting C (1, 1, 2), in setting D (1, 1, 3), and in setting E (1, 3, 1). In each case, the remaining agents are the dummy agent provided by the organiser. The average revenue of each kind of agent in each experiment is then plotted (see figure 7).

Now, in figure 7a, the results for setting A, B and E are presented.[10] Here from A to B to E, we increase the number of LTP-agents and decrease the number of dummy agents. In setting A, MIX- and LTP-agents perform significantly better than the STP-agent,[11] but we cannot differentiate statistically which is better between MIX- and LTP-agents. In setting B, we cannot differentiate statistically which agent is better among these three type of agents. In setting E, the STP-agent is significantly better than the LTP-agent and MIX-agent is slightly better than the LTP-agent. But we cannot differentiate statistically which agent is better between STP- and MIX-agents. From these three experiments, we can conclude that with an increasing number of LTP-agents, the performance of the STP- and MIX-agents are getting better, while that of the LTP-agent is relatively worse. This is because more agents are requesting components for the far future, leaving more spare production capacities for the suppliers in the near future. Thus the component price for the near future will drop which means the agents that purchase many more components in the near future will benefit from this price and thus have a low cost.

Similarly, in figure 7b, the result for settings A, C and D are presented. Here from A to C to D, we increase the number of STP-agents and decrease the number of dummy agents. In settings A

[10]The performance for the dummy agents are not shown in the figure since they are much worse than these three type of agents and incorporating them would make it difficult to see the difference among the three.

[11]In this case and in the statistics below, significance is computed by a Student's t-test ($p < 0.15$).



(a) Experimental results for settings A, B and E. (b) Experimental results for settings A, C and D.

Figure 7: Comparison of performance in various settings.

and C, MIX- and LTP-agents are significantly better than the STP-agent. But we cannot differentiate statistically which agent is better between MIX- and LTP-agent. This is because there are only two agents that purchase components for far away which is not likely to cause the price increase of the components in the far future. But for the STP-agent, there are some dummy agents which order components for a due date in the next several days, thus more competition in the near future leads to an the increase in the component price and higher overall procurement costs. In setting D, we obtain the same result as in setting C, but with a much smaller p value of the Student's t-test. This means the performance of STP-agents are actually getting worse when there are more agents like itself. However, in all the cases we investigated, the MIX-agent is the most stable one as it performs consistently in the presence of various numbers of STP- and LTP-agents. Overall, it obtains the highest average performance (17.82) and the lowest variance (11.68).

Moreover, as more agents use the same broad strategy of sensing the market and ordering intelligently for the near and far future, the performance of all the agents are negatively affected, (*i.e.*, the performance of all the agents is getting worse from experiment A through B to E, and from A through C to D). This happens because as the agents within the game become increasingly homogeneous, there is greater competition for components at a given time (because there are several agents ordering components for broadly the same due date for the far future), and thus the suppliers have less spare capacity, and component costs increase.

4. CONCLUSIONS

This paper provides a number of insights into building agents for supply chain management applications. Specifically, it details the design, implementation and evaluation of SouthamptonSCM; an agent that successfully participated in the 2005 TAC SCM. This agent employs fuzzy reasoning to determine how to set prices and uses a mixed component procurement strategy that balances long and short term orders. Through analysing an actual competition game from the 2005 final we found that this strategy enables SouthamptonSCM to buy more components at lower prices than the other most successful agents in the game. In a series of controlled experiment, where we varied the number of short-term and long-term-planning agents in the game, we found that the mixed procurement strategy was the most stable in all the cases that we considered.

Now, as well as being successful with the TAC SCM competition, several aspects of our agent design and strategy are applicable in a wider context. Firstly, the general structure of the component agent is to employ a mixed strategy where orders in the far future cover the minimum baseline quantities needed in low demand markets, whilst orders in the near future handle current changing demand. This mixture of baseline and opportunistic purchasing behaviour is a common strategy in this domain and the technology we develop for achieving this can be readily transferred. Second, we believe our pricing model technology will also be useful in real SCM applications where just undercutting competitors' prices can significantly improve profitability. Specifically, to apply our model in other domains, the designers of the rule base would need to adapt the fuzzy rules to reflect the factors that are most relevant. Now we believe that customer demand and inventory level are highly likely to be critical factors for almost all cases and thus these rules can remain unaltered. By using different rule bases, different factors can easily be incorporated (as we did here, in order to handle the additional need to reduce inventory towards the end of the game).

Our future work in this area focuses on the component agent. We would like to improve it so that it can adapt the quantity for far future and near future procurement automatically between the rounds of the games according to the procurement behaviours employed by the opponents. This can be achieved based on our controlled experiment results, where if there are more LTP-agents in the game, the agent will adapt its behaviour to the direction of STP-agent. Similarly, if there are more STP-agents, the best response is to change the buying behaviour to the direction of the LTP-agent. This knowledge of opponnets' behaviour can be obtained from the game history and thus this adaptation can be used in repeated games within the same participants.

5. REFERENCES

[1] J. Collins, R. Arunachalam, et al. The supply chain management game for the 2005 trading agent competition. Technical Report CMU-ISRI-04-139, School of Computer Science, Carnegie Mellon University, December 2004.

[2] M. He, H. F. Leung, and N. R. Jennings. A fuzzy logic based bidding strategy for autonomous agents in continuous double auctions. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1345–1363, 2003.

[3] M. He, A. Rogers, E. David, and N. R. Jennings. Designing and evaluating an adaptive trading agent for supply chain management applications. In *Proc. IJCAI Workshop on Trading Agent Design and Analysis*, pages 35–42, Edinburgh, Scotland, 2005.

[4] K. Kumar. Technology for supporting supply-chain management. *Comms of the ACM*, 44(6):58–61, 2001.

[5] D. Pardoe and P. Stone. Predictive planning for supply chain management. *Proc. International conference on automated planning and scheduling*, to appear.

[6] M. Sugeno. An introductory survey of fuzzy control. *Information Sciences*, 36:59–83, 1985.

[7] M. Wellman, J. Estelle, S. Singh, et al. Strategic interactions in a supply chain game. *Computational Intelligence*, 21(1):1–26, 2005.

Reducing Costly Information Acquisition in Auctions

Kate Larson
School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
klarson@cs.uwaterloo.ca

ABSTRACT

Most research on auctions assumes that potential bidders have private information about their willingness to pay for the item being auctioned, and that they use this information strategically when formulating their bids. In reality, bidders often have to go through a costly information-gathering process in order to learn their valuation for the item being auctioned. Recent attempts at modelling this phenomena has brought to light complex strategic behavior arising from information-gathering, and has shown that traditional approaches to auction and mechanism design are not able to overcome it. In this paper, we show that if the auction designer has some information about the agents' information-gathering processes, then it is possible to create an auction where, in equilibrium, agents have incentive to only gather information on their own valuation problems and to reveal the results truthfully to the auctioneer. Additionally, simulation results show that, from a system-level perspective, the overall cost of information acquisition is substantially lower in this new auction when it is compared to a classic auction mechanism.

Categories and Subject Descriptors: 1.2.11 Multiagent Systems
General Terms: Design, Theory
Keywords: Mechanism Design, Game Theory, Resource-bounded Agents

1. INTRODUCTION

Most research on auctions assumes that the problem facing bidders is how to bid given their private preferences for the item(s) being auctioned. In reality, however, bidders often have to go through a costly information-gathering process in order to determine what their actual preferences are. This process may involve such things as asking questions about a product in order to learn whether it is of high or low quality, solving optimization problems in order to ensure that the bidder bids on the minimum amount of materials needed to complete some job, or conducting research on a firm in order to learn whether its assets complement the assets currently held by the potential bidder.

Recently, researchers have started studying how computational and informational constraints influence both the auctioneer and the bidders. One direction of research has focused on what happens if the auctioneer does not have infinite computational powers. In many interesting auction domains, such as combinatorial auctions, the auctioneer is required to solve (possibly multiple) $\mathcal{NP}$-hard problems. Determining how to replace it with approximation algorithms, while still maintaining the desirable game-theoretic properties, as well as characterizing the domains of truthful social choice functions, has been a vibrant research area [5, 11, 13, 10, 14].

Researchers have also started studying computational and informational constraints faced by bidders. Most of the auction literature has assumed that the bidders know how they value the items being auctioned. However, in many settings bidders do not have adequate information in order to formulate their valuations and instead must learn them by computing or acquiring information (at some cost). One model that has been studied allows a bidder the choice between participating in the auction without knowing its true valuations or paying a fee to learn them. Questions asked using this model include what sort of incentives are required for bidders to acquire information about their valuations [1], and how does information acquisition depend on the rules of the auction [3, 15, 16].

It has also been noted that a bidder's decision as to whether to compute or gather information about it's valuations can depend on the preferences of others [18]. We have proposed explicitly modeling the information-gathering actions of bidders along with the decisions they make when deciding how to use their information-gathering resources. We placed this *deliberative-agent* model into a game theoretic framework, and have analyzed classic auctions in order to gain an understanding of the impact that computational and information-gathering constraints have on agents' strategic behavior [7]. This brought to light new forms of strategizing on the part of the bidders where they actively gathered information on their competitors, and provided a possible explanation for behavior seen in practice [17].

In recent work we studied the problem of designing auctions specifically for bidders who must gather information [9] in order to determine their actual valuations and formulate their bids. We proposed three intuitive properties for auctions. We argued that the auction should not have to know

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

the details of the bidders information-gathering processes (*preference-formation independence*), bidders should not have incentive to gather information about other bidders (*deliberation-proof*), and that the auction should support some level of honesty among the bidders (*non-misleading*). However, we showed that it is impossible to design an (interesting) auction that has these three properties.[1] The contribution of this paper is that we show by that relaxing one of the properties, preference-formation independence, and allowing the auction to use some minimal amount of information about the bidders, it is possible to avoid strategic-deliberation and misleading behavior.

The rest of the paper is organized as follows. In the next section we describe our deliberative-bidder model, and discuss the problems faced in designing auctions for such bidders. In Section 3 we describe an optimal-search procedure and discuss how it can be incorporated into an auction. We then show that this auction provides the appropriate incentives for deliberative bidders. In particular it reduces the strategic burden placed on these bidders as they decide how to use their (limited) deliberation resources. Finally, Section 5 reports on experimental results which illustrate the feasibility of running such a procedure, as well as the effects it has on reducing the total cost of information gathering.

2. DELIBERATIVE BIDDING AGENTS

In this section we provide the background and motivation for the rest of the paper. We start by presenting our model of a deliberative agent. We discuss the implications of having deliberative agents participate in auctions and, in particular, we highlight the complex strategic behavior that may arise when bidders are deliberative. We then discuss some of the challenges in designing auctions for such agents.

In the rest of this paper we assume that the reader has a basic understanding of elementary game theory and auction theory. While we do explain key concepts as they are needed, a more thorough treatment on this topic can be found in microeconomic texts or books specializing on auctions [12, 6]. We pause now to comment on the terminology used in this paper. We use the terms "bidder" and "agent" interchangeably. That is, every agent is a bidding agent. We also use the terms "deliberative" and "information-gathering" interchangeably.

2.1 Deliberative agents

In this paper we assume that bidders are *deliberative*. A deliberative bidder is one who must compute or gather information in order to determine how much it values the item(s) being auctioned, has restrictions on its computing or information-gathering capabilities, and who carefully considers how to use its available resources given its restrictions.

We assume that a deliberative agent has a set of deliberation resources, which we will simply refer to as the agent's resources. We denote the resources of agent i by R_i. An agent is able to apply its resources to any preference-determination problem it wishes. If there are m possible problems, then we let $(r_1, \ldots, r_m) \in R_i^m$ denote the situation where the agent has devoted r_j resources to problem j. In particular, the agent is allowed to deliberate on its own preference-formation problems, as well as the problems of any other agent.

We model deliberation-resource limitations through cost functions. The cost function of agent i is $\text{cost}_i : R_i^m \rightarrow \mathbb{R}^+$. The only restriction placed on the cost functions are that they must be additive and non-decreasing.

A deliberative agent i is endowed with a multi-set of algorithms $\mathcal{A}_i = \{A_i^j\}$ where A_i^j is the algorithm agent i can use on problem j. We use the term *algorithm* in its broadest sense; algorithms are step-by-step procedures for solving some problem. In particular, we include information gathering processes in our set of algorithms. The algorithms of a deliberative agent have the anytime property; they can be stopped at any point and are guaranteed to return a solution, but if additional resources are allocated to the problem then a better solution (or at least no worse a solution) will be returned.

A deliberative agent carefully decides how to allocate its resources on algorithms given its cost function. To help with this process, deliberative agents are equipped with a set of *performance profiles*, $\mathcal{PP}_i = \{P_i^j\}$, one performance profile for each algorithm. A performance profile has two components. First, it describes how allocating resources to a problem changes the output of the algorithm. In particular, it describes, for any resource allocation to a problem, what possible solutions the algorithm may return, conditional on any and all features of the algorithm and problem instance which are deemed to be of importance to the agent. This is coupled with a procedural component, which, given the descriptive part of the performance profile, returns a deliberation policy that describes how the agent should optimally allocate deliberation resources. A performance profile can be as simple as a probability distribution over possible valuations coupled with a rule that states whether the agent should deliberate or not, or it may use a more sophisticated procedure that allows agents to condition their deliberation policies on current results [8].

To summarize, a deliberative agent is defined by

$$\langle R_i, \text{cost}_i(\cdot), \mathcal{A}_i, \mathcal{PP}_i \rangle$$

where R_i is the set of deliberation resources of agent i, $\text{cost}_i : R_i^m \rightarrow \mathbb{R}$ is a cost function which limits the amount of resources the agent can use, $\mathcal{A}_i$ is the multi-set of algorithms available to the agent, and $\mathcal{PP}_i$ is the associated set of performance profiles.

2.2 Auctions and Deliberative Bidders

A deliberative bidder needs to decide how and whether to gather information on valuations and then how to use the information it has in order to bid. Figure 1 illustrates this process. Motivating the bidders is the desire to maximize their utility. In particular, the utility of a deliberative bidding agent depends on its valuation, whether or not it has won the item that was up for auction, and the price that it pays if it was the winner. That is, if an agent i had used resources $r = (r_1, \ldots, r_m)$, had determined valuation $v_i(r_i)$ had won the item, and had to pay price p, then its (expected) utility would be

$$u_i = v_i(r_i) - p - \text{cost}_i(r).$$

If the agent had not won the auction then its utility is

$$u_i = -\text{cost}_i(r).$$

[1] Auctions where the outcome is completely independent of the bidding strategies of the agents trivially satisfy the three properties.

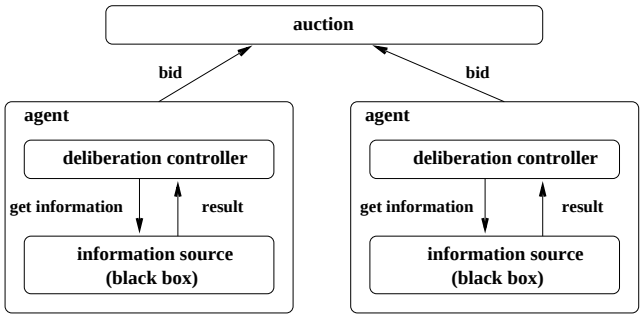


Figure 1: An auction with two deliberative bidders. Each bidder must gather information in order to determine how much it values the item that is being auctioned. A bidder may also have incentive to gather information on other participants in the auction.

Note that the bidder may use its deliberation resources on problems which do not directly affect its own valuation. However, this will still influence its utility since it incurs a cost for the total amount of resource usage.

A strategy for a deliberative bidder is a policy which specifies what actions (deliberative and bidding) to execute at every stage in the auction. We define a *history* at stage t, $H(t) \in \mathcal{H}(t)$, as the set which includes all actions (both deliberative and bidding) that the bidder itself has taken up to stage t, the results of the deliberation actions, as well as all actions other bidders may have taken (these may not have been observed by the agent). A (deliberation) *strategy* is a mapping from the set of histories to the set of actions (deliberative or bidding) for each stage in the game. That is, $S_i = (\sigma_i^t)_{t=0}^{\infty}$ and

$$\sigma_i^t : \mathcal{H}(t) \mapsto A_i$$

where A_i is the set of actions available to bidder i.

To clarify this definition we present a simple example. In a direct auction, bidders submit their bid directly to the auctioneer at some deadline T. The strategies of a deliberative bidder have the following form; $S_i = (\sigma_i^t)_{t=0}^{\infty}$ where

$$\sigma_i^t(H_i(t)) = \begin{cases} d_i^j & \text{if } t < T \text{ or } t > T \\ \hat{v} & \text{if } t = T \text{ and } \hat{v} \in \mathbb{R} \end{cases}$$

where $\hat{v}$ is the bid submitted to the auctioneer, and d_i^j is a deliberative action.

In this new, enlarged, strategy space we look for equilibria, which we call *deliberation equilibria*. When we studied the deliberation equilibria in standard auctions, we noted that the equilibrium strategies of the bidders could be very complex. In particular, deliberative bidders have incentive to deliberate on the valuation problems of other bidders, and to base their own deliberation actions on the results [7]. We call this behavior *strategic deliberation*.

We believe that strategic deliberation is a undesirable phenomenon. When bidders strategically deliberate they use their own costly information-gathering resources on valuation problems that do not directly relate to their own preferences, thus wasting these resources. Additionally, the strategic burden placed on the bidders is potentially overwhelming. They must take into consideration the bidding actions of competitors, the deliberation actions of competitors, as well as the results of their competitors deliberation actions, in order to determine how to best deliberate and bid for themselves.

Recently, we studied the problem of designing auctions explicitly for deliberative bidders [9]. To this end we proposed three desirable properties for such auctions. First, we argued that a well-designed auction should reduce the strategic burden of the bidders. In particular, a well-designed auction should not provide any incentive for bidders to strategically deliberate. That is, auctions should be *deliberation-proof*. Second, we proposed that an auction should encourage bidders to take actions which are consistent with their actual valuations. That is, bidders should not try to willfully mislead others concerning their valuations. We called this property *non-misleading*. Finally, we argued that an auction should not be involved in the information-gathering or deliberative process of the bidders. In particular, the auction should not depend on the actual form of the cost functions, anytime algorithms or performance profiles of the bidders. We called this property *preference-formation independence*.

While each of these properties is desirable in isolation, unfortunately it is impossible to design an (interesting) auction with all three [9]. In the rest of this paper we relax the preference-formation independent property and allow the auction to use some information about the information-gathering processes of the bidders. We show that by relaxing this one property it is possible to have an auction where the other two properties are achieved.

3. AN OPTIMAL-SEARCH AUCTION

The preference-formation independence property is very strong since it means that the auction is not able to use any information about the information-gathering processes of the bidders. In this section we slightly relax this property and allow the auction designer to use some information about the cost functions and performance profiles of the bidders. In Section 4 we show that this is enough to get a deliberation-proof and non-misleading auction.

The key insight is that if the auction designer has information about the bidders' cost functions and performance profiles, then it can use this information to sort the bidders. The auction can then process the bidders in order, getting each one in turn to gather information and bid until some criterion has been met. By using an optimal search procedure from the operations research literature [19], coupled with carefully designed reservation prices, it is possible to create an auction where the most promising bidders (i.e. the ones most likely to have a high valuation and thus win the auction) are asked to participate early in the process, and, secondly, where bidders are provided with enough information that they no longer have incentive to gather information about others.

This is not the first time that someone has proposed using a search procedure in an auction. In particular, Burguet [2] and Cremer, Spiegel and Zheng [4] have studied this idea and have proposed both auctions and revelation mechanisms. While the auction in this paper is based on the Cremer-Spiegel-Zheng auction, our bidder model, motivation, and the properties we prove differ substantially. Both Burguet and Cremer *et al.* assume that the bidders can only gather information on their own valuations, as opposed to our assumption that bidders are free to gather information

on any problem they wish. Second, their goal is to design an auction which maximizes the revenue of the seller and so are interested in extracting the expected surplus from the bidders by imposing appropriate admission fees. Instead, we focus on the strategic behavior of the bidders and show that the auction has desirable properties from their perspective.

We now describe the assumptions that have been made in this paper. First, we assume that there are n deliberative bidders, and that the performance profiles and the cost functions of the bidders are known to the auction designer and to each other. Second, we assume that the cost functions and the performance profiles of the bidders are independent. Both these assumptions are required for the results.

To simplify the explanation of the auction, as well as the notation, we will make certain assumptions about the performance profiles of the agents. In particular, we assume that while a bidder i does not know its valuation *a priori*, it does know that it comes from some interval $[0, \overline{v}_i]$ with probability distribution P_i which is equivalent to the descriptive component of the bidder's performance profile. We argue that this assumption is not unduly restrictive. First, deliberative agents with performance profiles like this still have incentive to strategically deliberate in standard auctions. Second, the proposed auction will still work in settings where bidders have more sophisticated performance profiles. The only repercussion is that sorting the bidders, as will be described next, becomes more complex.

In the rest of this section we describe the optimal-search auction. We first give an overview of the optimal search procedure used by the auctioneer, and then describe how the auctioneer uses it when setting up an auction. In the next section (Section 4) we prove that this has the properties we desire.

3.1 Optimal Search

In the operations research literature a search problem is often formulated in the following way. Assume that there are n boxes (bidders) to open (ask to deliberate), and each box (bidder) i has some random value v_i which is drawn from distribution (performance profile) $P_i(v)$ with density f_i. If the searcher opens a box (queries a bidder) i then it incurs a cost of cost_i. The payoff to the searcher is the maximum value found up to the point when it stops. The question is, then, in what order should the boxes (bidders) be searched (asked to deliberate) and when should the process stop?

Weitzman proved that this problem can be solved optimally using a very simple procedure [19]. First, each box (or bidder) is assigned a *cutoff value* K_i where

$$K_i = \int_0^{K_i} K_i f_i(v)dv + \int_{K_i}^{\overline{v}_i} v f_i(v)dv - \text{cost}_i.$$

If the searcher had already opened a box with reward K_i then it would be ambivalent between opening box i or keeping reward K_i. If the best reward found so far is less than K_i then the searcher is best off opening box i.

Once the cutoff values have been computed, the optimal search procedure is completely characterized by the following rules:

1. (**Selection Rule**) If a box is to be opened, it should be that closed box with highest reservation price.
2. (**Stopping Rule**) Terminate search whenever the maximum sampled reward exceeds the reservation price of every closed box.

The cutoff values and the search procedure have some interesting properties which make them particularly useful for our application. The cutoff values are not the expected valuations of the bidders and, in fact, the optimal search procedure may well search a bidder with a low expected value before a bidder with a higher expected value. As Weitzman noted "Other things being equal, it is optimal to sample first from distributions that are more spread out or riskier in hopes of striking it rich early and ending the costly search" [19]. Interestingly, one of the main incentives driving bidders to strategically deliberate is very similar. Bidders are better off paying some small amount to learn whether or not a competitor has a high valuation for an item (and thus would be likely to win the auction) and only then deciding whether it is worth while to gather information on its own valuation problem [7]. Some other appealing properties of the cutoff values and the search procedure are that as the cost of learning a valuation increases, the cutoff value decreases, and as the search progresses the valuations needed to stop the search decrease.

We note that the cutoff value calculation is equivalent to finding a fixed-point, and may be a computational challenge for the searcher. However, in many settings, finding the fixed point is relatively straightforward. For example, if v_i is drawn from the uniform distribution over $[0, 1]$, and the cost of learning v_i is 0.25, then the cutoff value is

$$\begin{aligned} K_i &= \int_0^{K_i} K_i dv + \int_{K_i}^{1} v dv - 0.25 \\ &= K_1^2/2 + 0.25. \end{aligned}$$

Solving for K_i, we get $K_i \approx 0.293$.

3.2 The Auction

The optimal-search auction progresses through a series of stages. First, using its knowledge of the performance profiles and cost functions of each bidder, it computes the cutoff values. It labels the bidders in decreasing order of cutoff values so that $K_1 > K_2 > \ldots K_n$. It then asks each bidder in turn to gather information on its valuation problem and reveal the result. If, after asking the first t bidders to deliberate and reveal their results, the highest valuation is greater than K_{t+1} then the auction is halted, the highest bidder wins the item, and the auction has followed the optimal search procedure. However, it is not possible to simply get each bidder to truthfully reveal their valuations when asked. Instead, the auctioneer uses reserve prices and a sequence of second-price auctions to get the bidders to truthfully reveal whether or not their deliberations have resulted in valuations which are larger than a specific cutoff value.

In stage 1, the auctioneer tells bidder 1 that it should gather information on its valuation problem. The auctioneer then makes a take-it-or-leave offer of price p_1^1 to the agent. If the offer is accepted then the auction stops, bidder 1 gets the item and pays p_1^1. If the offer is rejected then the auction continues to the next stage. If the auction reaches stage t then bidder t is asked to deliberate on its valuation problem. Once it has done so then bidders 1 to t participate in a second-price auction with bidder and stage specific reserve prices. If bidder i submits a bid which is greater than its reserve price p_i^t then the auction stops. If only one bidder submitted a bid which was greater than its reserve price then

it is given the item and it pays its reserve price. If there are multiple bidders who submitted bids that were higher than their reserve prices then the highest bidder i wins the item and pays $p = \max[p_i^t, b_{-i}]$ where $b_{-i} = \max\{b_j | b_j > p_j^t$ and $j \neq i\}$. If the auction reaches the final state, n, then all bidders have been asked to gather information on their valuations. All bidders then participate in a second-price auction with no reserve prices.

The auctioneer must be careful when it comes to setting the reserve prices. If it is not done properly then bidders may have incentive to lie about their valuation after they have learned them. If the reserve prices are too low then a bidder may want to submit a bid which is higher than its true valuation in order to stop the auction early so as to avoid competition from others. If the reserve prices are set too high then a bidder may submit a bid which is lower than its true valuation in the hopes that as the auction progresses the reserve prices will drop. Cremer *et al* showed that it is possible to set the reserve prices so that in (Bayes-Nash) equilibrium the bidders bid truthfully [4]. This is done by working backwards through the stages of the auction.

If the auction reaches the last stage, then all bidders know their valuation. Since they participate in a second-price auction, each bidder is best off bidding truthfully. In the second last stage the first $n-1$ bidders know their valuations. Assume all of these bidders but i bid truthfully. Our goal is to set the reserve price of bidder i, p_i^{n-1} so that bidder i will only submit a bid greater than p_i^{n-1} if its valuation is greater than or equal to K_n. If $v_i = K_n$ then the only way the bidder can win the auction is if all bidders who know their valuations submitted a bid less than K_n. That is $v_{-i} < K_n$. If bidder i bid truthfully, the auction would stop, it would win, and it would pay its reserve price. Its expected payoff is

$$(K_n - p_i^{n-1})\Pi_j P_j(K_n) \quad (1)$$

where $1 \leq j < n$, $j \neq i$. If bidder i lied about its valuation at stage $n-1$ then the auction would continue to the final stage where all bidders are informed and all participate in a second-price auction. Bidder i will only win if all bidders, including bidder n, have valuations which are less than $v_i = K_n$. Since, by assumption, all other bidders are telling the truth about their valuations, agent i knows that the first $n-1$ bidders (minus itself) must have valuations which are less than K_n. Thus, its payoff if it waits for the last round is

$$\Pi_{1\leq j\leq n-1} P_j(K_n) \int_0^{K_n} (K_n - v_{-i}^n) d\frac{\Pi_{1\leq j\leq n} P_j(v_{-i}^n)}{\Pi_{1\leq j<n-1} P_j(K_N)} \quad (2)$$

where $j \neq i$ and v_{-i}^n is the highest valuation of the bidders from 1 to n, not including bidder i. Bidder i is ambivalent between declaring its true valuation in round $n-1$ and waiting for the second-price auction in the next round whenever (1)=(2). If the reserve price p_i^{n-1} is set such that (1)=(2) then bidder i will truthfully reveal its valuation, given that all other bidders are also doing the same.

The appropriate reserve prices for earlier rounds are determined in a similar fashion. If in stage t bidder i has valuation $v_i = K_{t+1}$, and assuming that all bidders j, $1 \leq j \leq t$, $j \neq i$ reveal their valuations truthfully, then, if bidder j bids truthfully, the auction will stop and its payoff will be

$$(K_{t+1} - p_i^t)\Pi_j P_j(K_{t+1}) \quad (3)$$

where $1 \leq j < t$, $j \neq i$. If the bidder waits until the next stage and then reveals its true valuation, its expected payoff is

$$(K_{t+1} - p_i^{t+1})\Pi_j P_i(K_{t+2}) + \quad (4)$$
$$\Pi_j P_j(K_{t+1}) \int_{K_{t+2}}^{K_{t+1}} (K_{t+1} - v_{-i}^{t+1}) d\frac{\Pi_j P_j(v_{-i}^{t+1})}{\Pi_j P_j(K_{t+1})}$$

where $1 \leq j < t+1$ and $j \neq i$, and where bidder i will pay p_i^{t+1} if $v_{-i}^{t+1} < K_{t+2}$ and v_{-i}^{t+1} if $v_{-i}^{t+1} > K_{t+2}$. If p_i^t is set such that (3)=(4) then bidder i is best off bidding truthfully in stage t, assuming that all other possible bidders also bid truthfully.

4. PROPERTIES

The optimal-search auction described in the previous section has some properties which make it appealing. First, there exists a Bayes-Nash equilibrium where the bidders truthfully reveal their valuations at the right time. Secondly, and of particular interest, it is structured so that bidders have no incentive to strategically deliberate. That is, the optimal-search auction is deliberation-proof and non-misleading.

Proposition 1 *The optimal-search auction is non-misleading. There exists a Bayes-Nash equilibrium where bidders truthfully reveal their valuations when asked by the auctioneer.*

Proof: (Sketch) The full proof is quite technical and follows the proof described in Cremer *et al* [4]. Thus, we only provide an overview of the proof. The main idea is that the reserve prices used by the auctioneer have been carefully chosen so that a Bayes-Nash equilibrium, where the bidders truthfully reveal the results of their deliberation, exists. If the auction does not stop before all bidders have been asked to deliberate on their own valuation problems, then a second-price auction is run. At this point, all the bidders know their valuations and so are best off bidding truthfully. At stage $t < n$, using the equilibrium hypothesis that all other bidders who have learned their valuations truthfully reveal them, the reserve prices for bidder i were carefully designed so that it is in it's best interest to truthfully reveal the valuation it obtained via deliberation. This is because in stage t bidder i is participating in a second-price auction with $t-1$ other bidders and so it's bid does not change the price it will pay if it wins in stage t. Additionally, the reserve prices are such that if $v_i \geq K_{t+1}$ then bidder i is best off revealing this and stopping the auction in stage t, while if $v_i < K_{t+1}$ then the bidder is best off letting the auction continue. In either situation, truth-telling is bidder i's best response, given its beliefs about the potential valuations of all other bidders, conditional on the fact that the auction has reached stage t and the hypothesis that all other bidders truthfully reveal their valuation once they have learned it.

□

Proposition 2 *The optimal-search auction is deliberation-proof. Deliberative bidders have no incentive to use their deliberation resources on the valuation problems of others.*

Before we present the proof of this proposition, we provide some insight into why bidders strategically deliberate in auctions. Strategic deliberation is caused when there is some

form of asymmetry between the bidders in terms of the cost functions or the performance profiles. Strategic deliberation often occurs when one bidder gathers information on a competitor in order to check whether the competitor has a high valuation or not. Based on this information, the deliberating bidder will then decide whether or not it is worthwhile to deliberate on its own valuation problem. Interestingly, the optimal search procedure uses a similar motivation for sorting agents. In particular, it queries agents in such an order so that agents that other bidders may want to gather information on (i.e those with potentially high valuations and low cost) are asked to deliberate and reveal their results early in the process, thus removing the need for other agents to gather information on them.

Proof: (Sketch) If the auction stops before a bidder is asked to deliberate, then the bidder has no incentive to deliberate on any problem and thus will not strategically deliberate. If a bidder already knows its valuation then it has no incentive to learn the valuations of other bidders in the optimal-search auction, since it will participate in second-price auctions. Therefore, the only situation of interest is when the auction asks a bidder to gather information and it has not yet done so on its own valuation problem.

Another useful observation is that if $\text{cost}_i < \text{cost}_j$ then bidder i has no incentive to deliberate on the valuation problem of bidder j. That is, if it is more expensive to deliberate on another bidder's valuation problem compared to ones own, then one will not deliberate on the other's problem.

Case 1: Assume that $K_i > K_j$, and assume that the auction has reached stage i. Since the performance profiles of all bidders are common knowledge, bidder i can determine $E[v_i]$ and $E[v_j]$. If $E[v_i] \leq E[v_j]$ then it must be the case that $K_i < K_j$ since $\text{cost}_i > \text{cost}_j$. This is a contradiction, and so it must be the case that $E[v_j] < E[v_i]$.

If $E[v_i] > K_j$, then in expectation the auction will stop before it reaches bidder j and thus the valuation of bidder j is irrelevant. Therefore, bidder i has no incentive to incur a cost from learning it. Assume that $E[v_i] < K_j$. Let $t = \max\{t | K_t < E[v_i]\}$. That is, in expectation, stage t is the earliest stage in which bidder i could influence the outcome of the auction. If $K_t > E[v_j]$ then we are in a similar situation as before and so for the same reason bidder i has no incentive to deliberate on the valuation problem of bidder j. If $K_t < E[v_j] < E[v_i]$ then it is possible that v_j will effect what price bidder i may pay, but in expectation it will not effect the allocation decision of the auction. Since strategic-deliberation is driven by allocation uncertainty as opposed to price uncertainty, bidder i has no incentive to deliberate on player j.

Case 2: Now assume that $K_j > K_i$. If the auction never reaches stage i then bidder i has no incentive to deliberate on any valuation, and thus certainly not on the problem of bidder j. Assume that the auction does reach stage i. This can only occur if $v_j < K_i$. Since, the auction is implementing an optimal search process, we know that it must be the case that

$$v_j - \sum_{1<l<i} \text{cost}_l < E[v_i] - \text{cost}_i - \sum_{1<l<i} \text{cost}_l.$$

Therefore, by simple algebra,

$$v_j < E[v_i] - \text{cost}_i$$

Clearly bidder i now has no incentive to deliberate on bidder j's problem. In expectation bidder i has a higher valuation than bidder j and thus will be allocated the item before bidder j would. Secondly, even if bidder j influenced the price that bidder i had to pay, learning the actual valuation of bidder j will not change the deliberation strategy of bidder i.

□

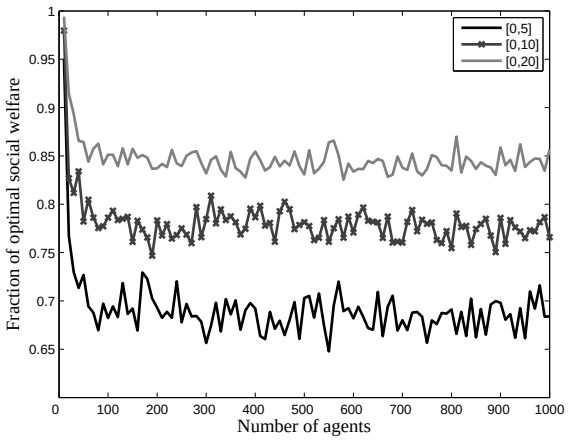


Figure 2: Fraction of the optimal allocation the search auction achieves. In this example all agents have their cost functions fixed to 1.0.

5. EXPERIMENTAL RESULTS

We investigated how the optimal-search auction worked in practice. In particular, we were interested in the economic efficiency of this approach, and whether it significantly reduced the amount of deliberation among the agents. To this end, we conducted two sets of simulations. The first set of simulations compared the allocation made by the search auction to the optimal allocation (made by a second-price auction). The second set of simulations compared the total cost of deliberation of all agents in the search auction with that of a second-price auction.

We set the parameters of the two sets of experiments the same way. We divided the bidders into six different classes, and then ran second-price auctions and optimal-search auctions in each class as we varied the number of bidders.

Each bidder had some valuation v_i which was drawn independently from the uniform distribution over $[0, V]$, where V was either 5, 10 or 20. For each of these valuation intervals, we set the cost functions in one of two ways. We first set the cost function of all agents in each class to be cost = 1.0. This meant that the cutoff values used in the optimal-search auction were the same for all bidders who's value had been drawn from the same interval. This put the optimal-search auction at a disadvantage since it had to run with minimal information available to it. We varied the cost functions for the bidders in other experiments. In particular, for any bidder who had a valuation drawn from the interval $[0, V]$, we randomly drew a cost constant cost_i from the uniform distribution over $[0, V/2]$. This meant that with high probability each bidder had a different cutoff value allowing the optimal-search auction to order the bidders to its advantage.

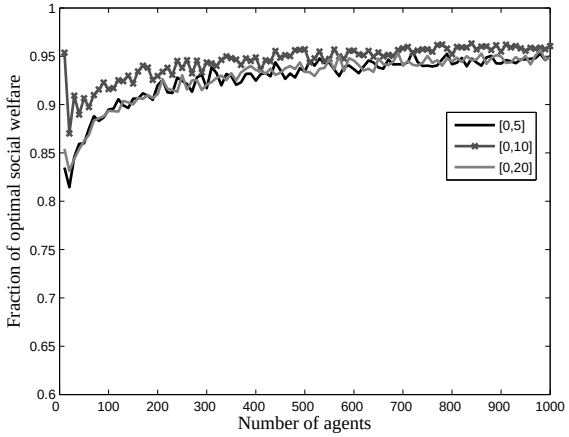


Figure 3: Fraction of the optimal allocation the search auction achieves. In this example, cost functions are drawn uniformly and independently from an interval related to their valuations.

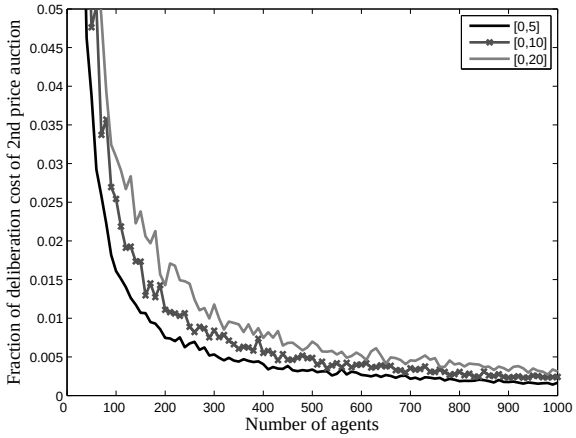


Figure 4: Cost incurred by all agents in the search auction compared to cost incurred in a second price auction. As the number of agents increase, the fraction becomes vanishingly small. Cost functions were fixed so that they were all equal to 1.0. This means that all agents (in each experiment) had the same cutoff value.

For each of the six parameter configurations we ran a series of second-price auctions and optimal-search auctions. We compared the efficiency of the optimal-search auction with that of the efficient second-price auction in order to determine what the efficiency loss was. We also compared the total cost of deliberation done by all bidders in the second-price auction with that done in the optimal-search auction in order to determine whether there was a substantial overall savings with the optimal-search auction. In each experiment we varied the number of bidders from 2 to 1000, and repeated each experiment 100 times. We present the average of our results.

In our first set of experiments we measured the efficiency of the search auction with that of the second-price auction. The second-price auction is guaranteed to allocate the item to the agent who has the highest valuation, while the search-auction allocates the item to the agent who has the highest valuation amongst the agents who have been asked to deliberate before the auction is stopped. It is possible that the search-auction is stopped before the agent with the actual highest valuation is asked to deliberate. Figures 2 and 3 show our findings.

In Figure 2, the search auction is at a disadvantage since all the cutoff values for the agents in each experiment are the same. However, we still observe that the search auction is within at least 65% of optimal (the allocation of the second-price auction) for all parameter settings. The allocation does depend on the interval from which the valuations are being drawn, with values from $[0, 5]$ leading to a lower allocation than the others. If the auction had more than 100 bidders, then the allocation did not seem to be affected by the number of bidders (this was expected). In Figure 3 the search auction was able to make better allocations when it was able to distinguish between the agents based on their cutoff values. Once we had at least 200 bidders, the allocation made by the search auction was always within 90% of optimal.

In our second set of experiments we studied whether or not the search auction reduced the total amount of deliberation done by all agents significantly, compared to what would happen in a second-price auction. In a second-price auction, given our setup, each agent would have incentive to gather information on their own valuation problem, while in the search auction only a subset of the agents will likely be asked to gather information (and thus incur a cost). Figures 4 and 5 show our results. Even if the search auction has little information by which to sort the agents (Figure 4) the overall savings in deliberation cost from a system-wide perspective is substantial. This is even more marked in the experiments where agents had different cutoff values, allowing the search auction to query the most promising agents first (Figure 5).

To conclude, our experiments show that the search auction does quite well when it comes to allocating the item. It also greatly reduces the amount of deliberation done by all agents in the auction.

6. CONCLUSION

A common assumption in auction research is that bidders have private information about their willingness to pay for the item being auctioned and that they use this information strategically when formulating their bids. In reality, bidders often have to undergo a costly information-gathering process in order to determine their valuations for the item being auctioned. Recent attempts at modelling this phenomenon has brought to light complex strategic behavior that had previously been overlooked. In particular, bidders may either have incentive to actively gather information on others or may have incentive to "fool" other bidders about their results. Earlier work proved that if the auction designer had no information about the bidders' information-gathering processes then this strategic behavior on the part of the agents was impossible to avoid.

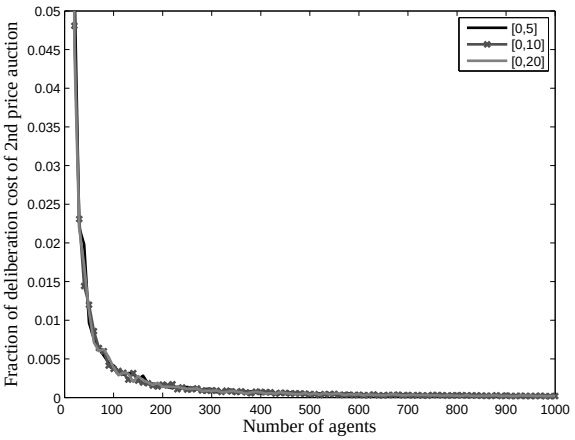


Figure 5: Cost incurred by all agents in the search auction compared to cost incurred in a second price auction. As the number of agents increase, the fraction becomes vanishingly small. Cost functions were drawn uniformly and independently from intervals related to their valuations.

In this paper we showed that by providing the auction designer with some minimal amount of information about the information-gathering processes of the bidders, it is possible to design an auction such that the bidders only gather information on their own valuation problems, and, when asked, report the results truthfully to the auctioneer. The insight is that by carefully ordering the agents using an optimal search procedure, and then asking them sequentially to gather information and reveal their results, it is possible to remove the uncertainty that drives the complex strategic behavior of deliberative agents in standard auctions. We additionally showed, via simulation results, that the savings in terms of overall information-gathering costs is significant.

There are several directions in which this work can be taken. This paper focussed solely on single-item auctions, and so generalizing the approach to multi-item auctions is an obvious next step. Second, the search auction relied on the assumption that the performance profiles and cost functions of the agents are known to everyone. While it seems likely that the auction also provides the correct incentives so that bidders will truthfully reveal their performance profiles and cost functions, this still needs further analysis. Finally, it is our belief that optimal search procedures from the operations research literature may find use in other settings, for example in situations where the total amount of communication has to be minimized.

7. REFERENCES

[1] D. Bergemann and J. Välimäki. Information acquisition and efficient mechanism design. *Econometrica*, 70:1007–1034, 2002.

[2] R. Burguet. Optimal repeated purchases when sellers are learning about costs. *Journal of Economic Theory*, 68:440–455, 1996.

[3] O. Compte and P. Jehiel. Auctions and information acquisition: Sealed-bid or dynamic formats?, 2001. Working Paper.

[4] J. Cremer, Y. Spiegel, and C. Zheng. Optimal selling mechanisms with costly information acquisition. Working paper, August 2003.

[5] N. Kfir-Dahav, D. Monderer, and M. Tennenholtz. Mechanism design for resource bounded agents. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS)*, pages 309–315, Boston, MA, 2000.

[6] V. Krishna. *Auction Theory*. Academic Press, 2002.

[7] K. Larson and T. Sandholm. Costly valuation computation in auctions. In *Theoretical Aspects of Rationality and Knowledge (TARK VIII)*, pages 169–182, Siena, Italy, July 2001.

[8] K. Larson and T. Sandholm. Using performance profile trees to improve deliberation control. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, San Jose, CA, USA, July 2004.

[9] K. Larson and T. Sandholm. Mechanism design and deliberative agents. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05)*, Utrecht, Netherlands, 2005.

[10] R. Lavi, A. Mu'Alem, and N. Nisan. Towards a characterization of truthful combinatorial auctions. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, 2003.

[11] D. Lehmann, L. I. O'Callaghan, and Y. Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002. Early version appeared in ACMEC-99.

[12] A. Mas-Colell, M. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[13] N. Nisan and A. Ronen. Computationally feasible VCG mechanisms. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 242–252, Minneapolis, MN, 2000.

[14] D. Parkes. *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*. PhD thesis, University of Pennsylvania, 2001.

[15] D. Parkes. Auction design with costly preference elicitation. *Annals of Mathematics and AI*, 44:269–302, 2004.

[16] N. Perisco. Information acquisition in auctions. *Econometrica*, 68(1):135–148, January 2000.

[17] M. Rothkopf. On misusing auctions to value stranded assets. *The Electricity Journal*, 10(10):10–17, December 1997.

[18] T. Sandholm. Issues in computational Vickrey auctions. *International Journal of Electronic Commerce*, 4(3):107–129, 2000. Special Issue on Applying Intelligent Agents for Electronic Commerce. A short, early version appeared at the Second International Conference on Multi–Agent Systems (ICMAS), pages 299–306, 1996.

[19] M. Weitzman. Optimal search for the best alternative. *Econometrica*, 47(3):641–654, 1979.

Computationally Efficient and Revenue Optimized Auctioneer's Strategy for Expanding Auctions

Onn Shehory
IBM Haifa Research Lab
Haifa 31905, Israel
onn@il.ibm.com

Eran Dror
Technion - Israel Institute of Technology
Haifa 32000, Israel

ABSTRACT

An expanding auction—a special type of an ascending-price open-cry auction—allows the auctioneer to dynamically add (identical) items to the single item offered initially. Expanding auctions are becoming an increasingly popular business-to-consumer auction mechanism. The auctioneer's revenue from an expanding auction depends, in large part, on its schedule for increasing the number of units offered. As we show, the naïve strategies commonly used for increasing the number of items offered in contemporary e-commerce implementations of expanding auctions are sub-optimal. In this study we provide a strategy that, given some assumptions about buyers' behaviors, maximizes the expected revenues of the auctioneer in an expanding auction. We model the auction process as a state graph in which nodes are auction states and edges are transitions. With this model, finding the optimal strategy is equivalent to solving a search problem on the state graph. We prove that the search problem to be solved, although seemingly exponentially complex, is actually linearly bounded. Based on this result, we introduce an informed decision strategy that optimizes the auctioneer's revenue.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent agents and multiagent systems.

Keywords

Expanding auction, auctioneer's revenue.

1. INTRODUCTION

Auctions are a popular electronic commerce tool [7]. Multi-unit auctions are also in use, however they are not as widely deployed as single-unit auctions are. This is despite the availability of an array of theoretical multi-unit auction solutions [5, 8]. We believe that the reasons for the limited use of multi-unit auctions are practical limitations of existing solutions such as high computational costs, difficulty of optimizing auctioneer's revenues [9], attack vulnerability, etc. The expanding multi-unit auction (initially studied in [11], though in limited detail), which is in use in several business-to-consumer (B2C) sites, is not an exception in this respect. Agents might be able to resolve some of these problems [10, 12], either on behalf of auctioneers or on behalf of buyers, however they should be provided with appropriate strategies for such decisions. The auctioneer's strategies for placing additional items up for sale in the expanding auction that are currently implemented in e-commerce sites do not optimize its revenues. In this study we overcome this limitation. We provide a novel strategy for the auctioneer to decide whether it should increase the number of items sold, or keep the number unchanged. The strategy is computationally feasible in real-world auctions, and it optimizes the auctioneers' expected revenues. This work is the first to provide such a strategy for an expanding auction.

Various mechanisms were developed to support the sale of multiple items [5, 7]. Some of these are sealed-bid auctions, e.g., Generalized Vickrey Auction (GVA) [3, 13]. GVA ensures that bidders expose their true valuation functions, which are defined over the possible allocations of the goods. GVA calculates the allocation that maximizes the sum of the bidders' utilities hence maximizing the auctioneer's revenue. The main disadvantage of this mechanism is that it is too complicated for practitioners, and in particular for consumers, to understand [2].

Ascending multi-unit auctions [2, 4] provide a reliable process of price discovery. This is important in auctions where a common value component is present, as is the case for commodities sold to consumers. An ascending bid multi-unit auction may be realized by enhancing the demand schedules approach. This method may be considered as a multi-round version of the multi-unit sealed bid mechanism [4]. Associated activity rules ensure that every bidder will contribute to price discovery as long as the auction process proceeds.

A better open format mechanism for allocating multiple units is the ascending clock mechanism [2]. Compared to the ascending bid multi-unit format this design has many advantages. Bidders have to submit only a single quantity in each round instead of a demand schedule, which makes the auction easier to implement and more comprehensible. Collusions are avoided by this mechanism, yet a uniform price is set in a similar manner to the uniform price auctions discussed earlier. This protocol yields revenues similar to the expected outcome with GVA. The main disadvantage here is that the number of units sold must be smaller than the number of units requested (as demand is greater than supply). Therefore this mechanism is more suitable to limited supply

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8–12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005…$5.00.

goods like communication spectrum or treasury bills, but not for commodities.

The expanding auction format is not well studied in terms of its theoretical properties. It is an open ascending price auction, and thus it has similarities to other such protocols. However, it has a simple, comprehensible user interface. Further, it is simple to implement. And as we show in this study, optimizing the auctioneer's expected revenues can be performed in linear time.

2. EXPANDING AUCTIONS

One of the reasons for multi-unit auctions' lack of popularity as B2C Internet auctions may be the likelihood of premature termination of the auction execution process. Premature termination may occur when the number of participants in the auction does not exceed the number of units offered (assuming each bidder desires one unit of the good). In single unit auctions such premature termination has a minor chance of occurring because the participation of a single bidder is usually unlikely. In multi-unit auctions the auctioneer may set a reservation price to prevent the loss expected from premature termination. However, this approach may deter bidders from the auction.

An emerging new auction type used in B2C e-commerce sites, which addresses the problem of premature auction termination, is the expanding multi-unit auction[1] (in short: the expanding auction). In an expanding auction, initially one item of a good is offered to the bidders. The bidders are free to raise their bids whereas the auctioneer is free to raise the number of units offered by one each time a new bid is received, with no upper limit on the number of units to be offered. The auction terminates when no more bids are placed. The winner list consists of the n highest bids offered where n is the number of units offered upon auction termination. The first bidder to enter the winner list is the first one to leave it. Similarly, with the Japanese auction [6] the auctioneer sets a bid rate used to gradually raise the price. Bidders are not allowed to raise their bids at a rate other than the one set by the auctioneer. It is assumed that each bidder desires one unit of the good and is risk neutral, therefore his/her utility function is given by $U_i=V_i-p$, where p is the price paid and V_i is bidder i's valuation of the good.

The expanding auction increases the auctioneer's control over the execution process. In a typical multi-unit auction the auctioneer has no control over how the execution will proceed once the initial parameters have been set. In contrast, in the expanding auction, the number of units offered may be changed dynamically. This allows the auctioneer to terminate the execution by gradually raising the number of units offered, or to continue the process to a certain extent by preserving the number of units offered. The control granted enables the auctioneer to set the number of units offered according to the number of bidders participating in the auction, making it easier to avoid the premature termination phenomenon observed in typical multi-unit auction mechanisms.

[1] We found no reference in the literature for the mechanism described here as the 'Expanding multi-unit auction' mechanism.

The most important aspect in the implementation of an expanding auction is the decision under which conditions to raise the number of units offered and under which to preserve it unchanged. This decision has a significant effect on auction execution, and more importantly on the auctioneer's revenues.

A simple but naive strategy for addressing this issue, used by B2C sites, is the step function strategy. For example, a step function strategy is used by an auction site named OlSale. (details are in the OlSale statute [1]). The strategy is based on a step function set by the auctioneer, composed of <price level, number of units> pairs that determine the number of units to be offered when the current bid reaches a certain level. Figure 1 depicts the step function strategy. When the current bid reaches p_1 the auctioneer offers one more unit, raising the number of units offered to 2. When the price reaches p_2 the auctioneer offers an additional unit raising the number of units to 3, and so on.

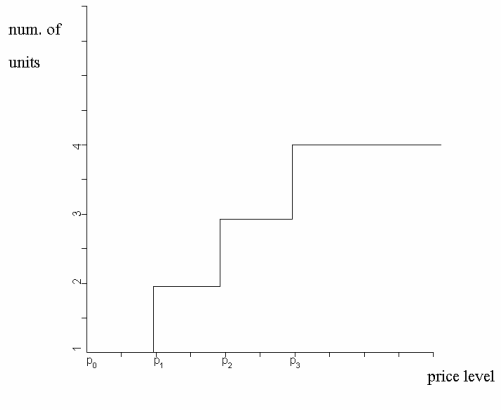


Figure 1. A step function policy

The step function strategy is simple to implement, however it is not trivial to configure its parameters in order to maximize the auctioneer's revenue. The same step function may yield higher revenues under certain conditions and lower revenues in others. This is illustrated in Example 1.

Example 1. Let (b_1,b_2,b_3) be 3 bidders with maximal bidding values (100,1000,2000), correspondingly. The bidders participate in an expanding auction in which the minimum bid rate is 10 and the step function is as given in Figure 1. Suppose that p_1=100; b_2 and b_3 may win the auction with bids equal to (100,110) respectively. If p_1=1000, b_2 and b_3 may win the auction with bids equal to (1000,1010), resulting in higher revenues obtained by the auctioneer compared to p_1=100. In contrast, when the bidding values are (100,110,120), p_1=100 results in higher revenues obtained by the auctioneer. With p_1=100, b_2 and b_3 will win the auction with bids equal to (100,110), respectively, whereas with p_1=1000 only b_3 will win the good at a price of 120.

The goal of this study is to overcome the weakness of the naïve step function policy presented above. We devise a strategy that, by taking into account the expected maximal bidding values, outperforms the step function strategy. It is important to note that this study assumes the new strategy it offers will maintain its optimality as long as the behavior of the bidder population does not undergo a fundamental change. Clearly, in a population of rational bidders, this assumption

might not hold. Yet, typical human participants in B2C sites are not rational bidders. Therefore, the suggested new strategy should be valuable for contemporary expanding auction sites. Optimal bidder strategies and their effect on the suggested auctioneer's strategy are left for future research and are out of the scope of this paper.

3. THE INFORMED DECISION STRATEGY

We model the expanding auction by a state graph. The nodes of the graph represent the states of the auction at a given point in time, and the edges represent transitions between states. Each state denoted by S has the following properties:

- The price a bidder must pay to join the winner list, denoted by l_d, where d denotes the bid sequence. For instance, l_2 denotes the price a bidder must pay to join the winner list after 2 bids have already been submitted and accepted by the auctioneer. Note that all states that may be formed after the auctioneer has received exactly d bids share the same price level l_d.
- The number of units the auctioneer is willing to sell at price l_d, denoted by n_S.
- A list B_S of maximal bidding values[2] (MBV) of the bidders currently participating in the auction. B_S contains MBVs of bidders which are either in the winner list, or willing to pay more than l_d and may therefore join the winner list at a later stage.
- A list of MBVs of the bidders in the winner list, denoted by B_{sw}. Clearly, $B_{sw} \subseteq B_S$.

Two operators will define the transition between states. The operators are defining the auctioneer's decision on *how to proceed with the auction process* given a particular state. The *raise*() operator corresponds to the auctioneer's decision to raise the number of units by one whereas the *preserve*() operator corresponds to his/her decision to keep the number of units offered unchanged. The operators are activated on states and produce new states.

The winner list is a queue that consists of bids placed by bidders. The size of this queue is equal to the number of units offered by the auctioneer. Once a bidder enters the winner list he/she is positioned at the end of the list. The first bidder to leave the list is the first bidder that entered the list (FIFO). The transition between states may affect the size of the queue, that is, the size is increased by one once the auctioneer decides to raise the number of units offered, otherwise the size remains unchanged.

Example 2. This example illustrates the state tree representing possible executions of an expanding auction process (cf. Figure 2). It has the following properties:

- The initial number of units offered by the auctioneer denoted by n_0 is 1.
- The reservation price denoted by l_0 is 90.
- The minimum bid increment denoted by Δ is 10.
- The initial list of the MBVs of the participating bidders denoted by B_0 is (100,120,130).

To simplify the illustration, in this example (and only in this example) it is assumed that the auctioneer first receives bids originating from bidders that have lower MBVs than other participating bidders have. This means that if multiple bidders submit equal bids at the same price level l_d the auctioneer will accept the bid originating from the bidder with the lowest MBV among the MBVs of the submitting bidders. The assumption ensures a single representation of the state graph illustrated in Figure 2. It will not be assumed further that this assumption holds.

In Figure 2, a state S_i is represented as follows: S_i=(b_1,b_2,b_3|n_i). In this representation, b_i refers to the MBV of bidder i, and n_i is the number of items offered by the auctioneer. Within the list of MBVs, some may be emphasized in bold-face font. The emphasized MBVs correspond to the bidders which are in the current winner list. Note that since not all bidders participate in the auction at all states, in some states the MBV list includes only a subset of bidders.

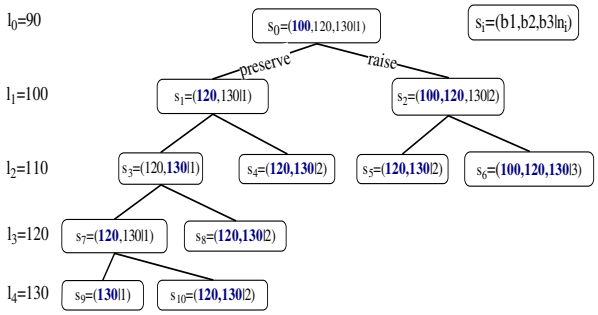


Figure 2. A state graph representing the dynamics of an auction process

The auction in Figure 2 starts with bidder b_1 obtaining the unit offered at 90$. In case that the auctioneer decides to raise the number of units offered, bidder b_2 joins the winner list by offering 100$ for the second unit. A transition from S_2 to its *raise* descendent S_6 results in b_3 joining the winner list with a bid of 110$. More interesting is the transition from S_2 to its *preserve* descendent, which results in the removal of b_1 from the winner list to be replaced by bidder b_3. Bidder b_1 also leaves the auction since he/she is not willing to pay more than 100$ for the good. The auction halts when all bidders in a given state are also in the winner list and therefore do not have the incentive to bid. These states are the terminating states (S_4 ,S_5 ,S_6 ,S_8 ,S_9 ,S_{10}).

It is important to observe that given the MBVs of bidders entering the auction we can infer the most beneficial strategy by forming and inspecting the state tree. In this example, raising the number of units offered repeatedly leads the process to state S_6 in which the auctioneer gains 100+100+90=300, thus outperforming all other terminating states.

In developing a decision strategy that is based on the state tree described, we face two fundamental problems. Firstly, the MBVs of bidders entering the auction are unknown. Secondly,

[2] The maximal bidding value of a bidder is the maximal bid he/she is willing to offer during the auction. This value may differ from the private value, depending on his/her strategy.

the search for the most beneficial strategy may be expensive in terms of complexity of the search problem to be solved.

To address the first problem, we assume that for each bidder b_i participating in the auction, a probability distribution function f_i of his/her maximal bidding value is given or can be learned from historical biddings. The probability distribution function f_i depicts the price bidder i is willing to pay for the good compared to its catalog price. The comparison to the catalog price is expressed in percentage points thus normalizing the data collected across different types of goods.

Using the statistical data gathered, a strategy that takes into account the bidding profile of the bidders participating in the auction can be devised. The informed decision strategy (in short, IDS) we propose is such a strategy. It is a simulation-based strategy that, given the current state of the auction, attempts to determine the auctioneer's action that would yield the best results. It does this by generating MBVs vectors of bidders participating in an auction and the current state of the auction. To find what the auctioneer's best choice for proceeding the auction is, a search problem that takes into account all possible scenarios has to be solved for each MBV vector generated. A solution should determine whether the auctioneer should increase the number of units sold by one, or preserve it unchanged. The knowledge applied to the simulation quantifies the value of each choice, thus improving the chance of making a good decision.

Given the auctioneer's reservation price l_0 and the maximal bidding value of the bidders participating l_{max}, we can establish a bound on the number of bid levels. The worst case is arrived at when only one unit is offered during the lifetime of the auction. Hence, the number of bid levels is bounded by $(l_{max} - l_0)/\Delta$. Since the branching factor of the tree is 2, the number of states in the state tree is bounded by $O(2^{(l_{\max}-l_0)/\Delta})$. Therefore, computing the complete state tree is too expensive in terms of the time it takes for the auctioneer to respond to bid offers submitted by the participating bidders. The responsiveness of the auction process will be downgraded significantly, causing bidders to leave the auction. In Section 4 we show that the complexity of the search problem can be reduced to linear time.

Provided that bidders' MBVs can be estimated and the complexity of the search problem can be reduced to linear time, the IDS may be implemented in real-world Internet auctions. We presume that an auctioneer implementing the IDS would like to control the amount of time allocated to computing the action needed to maximize his/her revenues. This will enable the auctioneer to guarantee an acceptable responsiveness level for new bids received.

The algorithm presented below describes the implementation of the IDS within the auction execution process. The time frame allocated by the auctioneer for computing the optimal strategy is denoted by t_p, and the time in which a new bid that raises the price of the good sold is received is denoted by t_0:

1. $t_0 \leftarrow now$, $nRaise \leftarrow 0$, $nPreserve \leftarrow 0$
2. Generate an MBV vector v_{sample} based on the probability distribution functions of participating bidders.
3. Using the IDS compute the optimal action to be taken based on vector v_{sample}. If raising the number of units sold by one is the best course of action then $nRaise \leftarrow nRaise+1$, else $nPreserve \leftarrow nPreserve+1$.
4. If $now < t_0 + t_p$ go to step 2.
5. If $nRaise > nPreserve$ raise the number of units sold by one, if $nRaise < nPreserve$ preserve the number of units sold. Otherwise choose one of the operators randomly.

Detailed explanations of step 3 above are in the proceeding sections.

4. CALCULATING THE STRATEGY IN LINEAR TIME

Given a state S and the required statistical data for deriving the MBVs of the bidders currently participating in an auction, it is possible to compute the optimal set of actions the auctioneer has to perform to maximize his/her revenue. As shown above, the complexity of this analysis is bounded by $O(2^{(l_{\max}-l_0)/\Delta})$. This complexity is infeasible for real-world applications of auction sites. In this section we show that the complexity of the search problem is actually linear in the maximal number of bid levels of the auction. We do this by utilizing the knowledge we have with regards to the properties of the problem domain we address. This knowledge enables us to prune a substantial part of the states in the graph. We show that the search problem can be bounded by $(l_{max} - l_0)/\Delta$, thus proving that the analysis can be completed in a linearly bounded computation time. This result enables the implementation of the IDS in real-world Internet auctions.

The proof starts by introducing a set of invariants that capture some basic properties of the state tree. Using these invariants, we initially prove that the number of states to be analyzed can be reduced to $O(((l_{max} - l_0)/\Delta)^2)$, thus providing a polynomial time bounded analysis procedure. Based on the first proof, we later prove that the number of states that have to be analyzed is linearly bounded by $O((l_{max} - l_0)/\Delta)$.

Invariant 1. Given a state S, a descendent state will be produced if there exists a bidder $b^* \in B_S$ which is not in the current winner list, and his/her MBV satisfies $p_{b^*} \geq l_d + \Delta$. When this condition holds, state S will have 2 descendent states; otherwise it will have no descendent states. Formally: $b^* = \exists b \in B_S \mid b \notin B_{sw}, p_b \geq l_d + \Delta$

Invariant 2. Given a state S, we denote the bidders' list that would result from the auctioneer's decision to raise the number of units offered by $B_{raise(s)}$. The list $B_{raise(s)}$ consists of the bidders belonging to S's winner list and an additional bidder b^* that joins S's winner list (see Invariant 1). Bidders that do not participate in S's winner list, but their MBVs satisfy $p_b > l_d + \Delta$, are included in the descendent state bidder list although they do not participate in its winner list. Formally:

$$B_{raise(s)} = \{(b \in B_S \mid (b \in B_{sw}) \vee (p_b > l_d + \Delta) \vee (b = b^*)\}$$

Invariant 3. Given a state S, we denote the bidders' list that would result from the auctioneer's decision to preserve the number of units offered by $B_{preserve(s)}$. When traversing from S to its preserve descendent state, a bidder $b' \in B_{sw}$ will be removed from S's winner list, substituted by another bidder b^*

that satisfies $b^* \in B_S$, $b^* \notin B_{sw}$. Bidder b' will remain in the descendent bidder list only if his/her MBV satisfies $p_{b'} > l_d + \Delta$. Provided that bidder b' satisfies this condition the bidder list of the descendent state is given by $B_{preserve(s)} = B_{raise(s)}$.
Otherwise the bidder list is given by:

$$B_{preserve(s)} = \{(b \in B_S \mid (b \in B_{sw}, b \neq b') \vee (p_b > l_d + \Delta) \vee b^*\}$$

Invariant 4. When traversing from state S to state *raise*(S), no bidder is dropped from the winner list. This results from the additional unit offered by the auctioneer. Hence, $B_{sw} \subset B_{raise(s),w}$.

Figure 3 illustrates a formal description of the state graph formed as a result of computing an auction execution. Each node S_i in the graph consists of its bidder list denoted by B_i. The initial number of units offered by the auctioneer is denoted by n_0 and the reservation price set initially by the auctioneer is denoted by l_0. The auctioneer's decision to raise the number of units is reflected in the graph by traversing to the right. The descendent state created is characterized by an additional unit offered to the participating bidders and a price level that is higher by Δ. In case that the auctioneer preserves the number of units the graph branches to the left forming a state that is characterized by the same number of units offered and a price level that is higher by Δ. Note that the state tree formed is not necessarily a complete tree but is shown as such to cover cases in which a complete tree is formed.

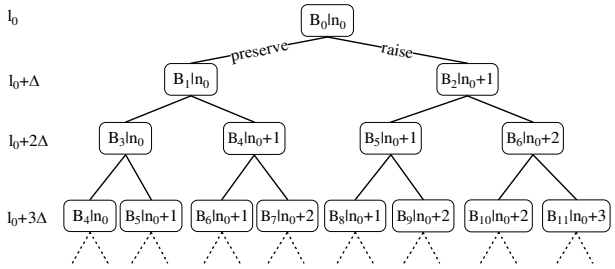


Figure 3. An auction execution state graph

For a given state S_i we define its preserve branch denoted by *preserve_branch*(S_i) as the set of states formed by iteratively traversing left starting at state S_i until we reach a terminating state. The following algorithm defines (and constructs) *preserve_branch*(S_i):

1. $S \leftarrow S_i$, *preserve_branch*(S_i) $\leftarrow S_i$
2. If deg(S) = 0 stop.
3. *preserve_branch*(S_i)$\leftarrow$*preserve_branch*(S_i)$\cup$ {*preserve*(S)}
4. $S \leftarrow$ {*preserve*(S)}
5. go to step 2

In a similar manner we define *raise_branch*(S_i). *raise_branch*(S_i) is formed by iteratively traversing right, starting at state S_i to its raise descendents. *raise_branch*(S_i) is constructed by the algorithm:

1. $S \leftarrow S_i$, *raise_branch*(S_i) $\leftarrow S_i$
2. If deg(S) = 0 stop
3. *raise_branch*(S_i)$\leftarrow$*raise_branch*(S_i)$\cup$ {*raise*(S)}
4. $S \leftarrow$ {*raise*(S)}
5. go to step 2

Following, we introduce three theorems. These show that the complexity of searching the auction's state tree can be reduced to polynomial and linear.

Theorem 1. Let T_0 be a state tree of an expanding auction whose root is S_0. Let S_{T_0} be the set of terminating states of T_0. Let $S_{T_0}^*$ be the subset of terminating states reached by traversing on *raise_branch*(S_i) for each state S_i in *preserve_branch*(S_0). We define $S_{T_0}^*$ formally as follows. Let P_{T_0} = *preserve_branch*(S_0) and $R_{T_0} = \cup$ *raise_branch*(s) $\mid s \in P_{T_0}$ then $S_{T_0}^* = \{ s \in R_{T_0} \cup P_{T_0} \mid \deg(s)=0\}$. This theorem presents two claims: (1) $S_{T_0} = S_{T_0}^*$. That is, $S_{T_0}^*$ covers all the terminating states in the state tree produced by the auction execution. (2) The number of states to be visited for computing $S_{T_0}^*$ is bounded by $O(((l_{max} - l_0)/\Delta)^2)$.

Proof (of claim 2)**.** Each traversal from state S_i to its terminating state in *raise_branch*(S_i) is bounded by $O((l_{max} - l_0)/\Delta)$ states (the depth of the tree). In the worst case the size of *preserve_branch*(S_0) is $(l_{max} - l_0)/\Delta$. It hence follows that number of states to be visited in *preserve_branch*(S_0) is bounded by $O(((l_{max} - l_0)/\Delta)^2)$ ■

From Theorem 1 it follows that one needs to search only $O(((l_{max} - l_0)/\Delta)^2)$ nodes to find the best auctioneer's action. For space reasons, the proof of Claim 1 is omitted. The definitions introduced in Theorem 1 and the approach to proving it are illustrated in Figure 4.

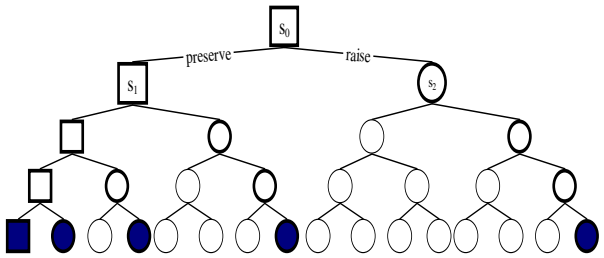


Figure 4. $S_{T_0}^*$,R_{T_0},P_{T_0} definitions: P_{T_0} states are rectangle shaped, R_{T_0} states are marked with a thick border style, $S_{T_0}^*$ are filled shapes.

Theorem 2. Let S_1, S_2 be two states such that S_2=*preserve*(S_1). Let S_{1r}, S_{2r} be the terminating states reached by traversing *raise_branch*(S_1) and *raise_branch*(S_2), respectively. Let Rev(S_{1r}) and Rev(S_{2r}) be the revenues collected by the auctioneer at states S_{1r} and S_{2r}, respectively. Given Rev(S_{1r}), Rev(S_{2r}) can be computed as following:

If $P_b \geq (depth(S_{1r})+1)\cdot\Delta + l_{s1}$ then $\text{Rev}(S_{2r}) = \text{Rev}(S_{1r}) + (depth(S_{1r}) + |B_{s_1 w}|)\cdot\Delta$. Otherwise, $\text{Rev}(S_{2r}) = \text{Rev}(S_{1r}) - (l_{s1} - \Delta\cdot(|B_{s_1 w}| - 1))$. Here, P_b denotes the MBV of the bidder b that leaves S_1 winner list when traversing from S_1 to S_2; l_{s1} denotes the price level of state S_1 and $depth(S_{1r})$ denotes the number of states produced while traversing from S_1 to S_{1r}. Figure 5 illustrates the states produced in case that $P_b < (depth(S_{1r})+1)\cdot\Delta + l_{s1}$. Figure 6 illustrates the state produced in case that $P_b \geq (depth(S_{1r})+1)\cdot\Delta + l_{s1}$.

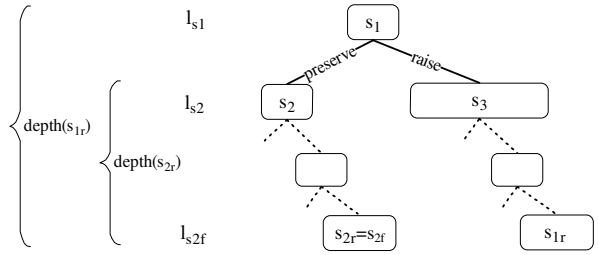


Figure 5. The auctioneer's revenue in case that $P_b < (depth(S_{1r})+1)\cdot\Delta+l_{s1}$

This theorem presents three claims. Claim (1) states that, in the case of $P_b < (depth(S_{1r})+1)\cdot\Delta+l_{s1}$, when traversing from S_1 to S_2 the auctioneer loses the revenue received from bidder b's participation in the winner list at state S_1. Therefore the auctioneer's revenue at state S_{2r} is smaller by P_b than its potential revenue at state S_{1r}.

Claim (2) states that, in the case of $P_b \geq (depth(S_{1r})+1)\cdot\Delta+l_{s1}$, bidder b submits an additional bid at state S_{2f} and thus increases the potential revenue at state S_{2r} by $(depth(S_{1r})+1)\cdot\Delta$ compared to the revenue obtained at state S_{1r}.

Claim (3) states that the number of states produced while traversing from S_2 to S_{2r} equals $depth(S_{1r})$ in case $P_b \geq (depth(S_{1r})+1)\cdot\Delta+l_{s1}$ or equals $depth(S_{1r})$-1 otherwise.

Proof: Traversing $raise\_branch(S_1)$ to a terminating state we discover $depth(S_{1r})$ new states. These states were produced as a result of $depth(S_{1r})$ bids received by the auctioneer. The same bids are received by the auctioneer while traversing from S_1 to S_2 and then along $raise\_branch(S_1)$. Observing state S_2 and $raise(S_1)$ it is evident that $B_{s2} \cup \{b\} = B_{raise(s1)}$ since while traversing from S_1 to S_2 bidder b, which leaves the winner list, might leave the bidder list as well. Assuming that $B_{s2} = B_{raise(s1)}$ we can conclude that since these states share the same price level l_{s2}, all $depth(S_{1r})$ bids received by the auctioneer while traversing $raise\_branch(S_1)$ will produce new states while traversing from S_1 to S_2 and then along $raise\_branch(S_2)$. Moreover, knowing that none of these bids belong to bidder b, which participates in S_1 winner list and therefore does not have the incentive to send bids to the auctioneer, we conclude that even if $B_{s2} = B_{raise(s1)} \setminus b$, all $depth(S_{1r})$ bids will produce new states while traversing from S_1 to S_2 and then along $raise\_branch(S_2)$ until we reach state S_{2f}. Observing states S_{1r}, S_{2f} shown Figures 5 and 6 we conclude that they both share the same price level since they are both at a distance of $depth(S_{1r})$ from S_1, and that $B_{s_{2f}w} \cup \{b\} = B_{s_{1r}w}$, since b has left the winner list while traversing from S_1 to S_2, and all bids that were accepted by the auctioneer while traversing from S_1 to S_{1r} were also accepted while traversing from S_1 to S_{2f}. It is important to observe that in case that $b \chi B_{s_{2f}w}$, b is the only bidder in state S_{2f} that is willing to raise the price level. Otherwise, there exists another bidder b' that is willing to do so; that is, $P_{b'} \geq l_{s2f} + \Delta$, and therefore the same bidder b' should participate in S_{1r} leading us to the conclusion that S_{1r} is not a terminating state.

In order for bidder b to participate in S_{2f}, the condition $P_b \geq depth(S_{1r})\cdot\Delta+l_{s1}$ must hold true. Further, bidder b will be willing to raise the price level if $P_b \geq (depth(S_{1r})+1)\cdot\Delta+l_{s1}$; this results in a new state in which all bidders participate in the winner list, and therefore it is a terminating state. That is, we reached S_{2r} (see Figure 6). The revenue obtained in this case equals $Rev(S_{1r})+(depth(S_{1r})+|B_{s_jw}|)\cdot\Delta$, since bidder b pays $l_{s1}+(depth(S_{1r})+1)\cdot\Delta$ instead of $l_{s1}-\Delta\cdot(|B_{s_jw}|-1)$ in state S_{1r}. The number of states produced while traversing from S_2 to S_{2r} consists of $depth(S_{1r})$-1 states produced while traversing from S_2 to S_{2f} and an additional state produced in this case, therefore $depth(S_{1r}) = depth(S_{2r})$.

If the condition above does not hold, bidder b will not participate in S_{2f} while other bidders participate in the winner list. Therefore, we reached the terminating state S_{2r}. Since bidder b does not participate in the winner list the auctioneer's revenue decreases by $l_{s1}-\Delta\cdot(|B_{s_jw}|-1)$ compared to S_{1r} revenue. The number of states produced while traversing from S_2 to S_{2r} consists of $depth(S_{1r})$-1 states produced while traversing from S_2 to S_{2f}, therefore $depth(S_{1r})$-1= $depth(S_{2r})$ ■

For space limitations, the proof of the theorem is presented only in part. Theorem 2 provides a better understanding of the auction execution process. Based on this understanding, we can refine, and precisely define, step 3 of the algorithm presented in Section 3. That step computes the optimal action to be taken based on the current state of the auction denoted by S_1 and the MBVs of the participating bidders given by vector v_{sample}.

Theorem 3. The number of states to be visited for computing *IDS* is bounded by $O((l_{max} - l_0)/\Delta)$. The proof is expressed by the algorithm presented below. ■

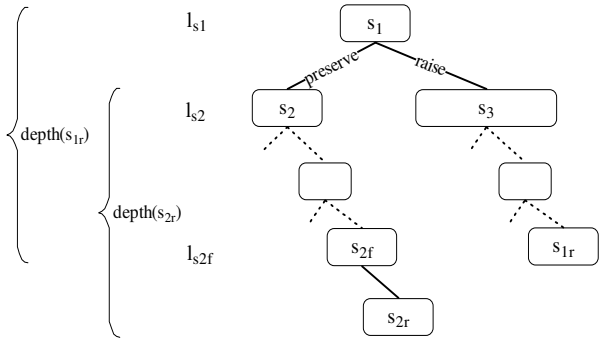


Figure 6. The auctioneer's revenues in case that $P_b \geq (depth(S_{1r})+1)\cdot\Delta+l_{s1}$

Figure 7 illustrates the details of step 3 of the algorithm. Starting from the current state of the auction, denoted by S_1, compute $depth(S_{1r})$ and $rev(S_{1r})$. This is done by computing each state produced along S_1 raise branch until S_{1r} is reached. Then, compute $depth(y_r)$ and $rev(y_r)$, where y_r is the terminating state reached by traversing from S_1 to its preserve descendent and then traversing along its descendent raise branch. The computation of $depth(y_r)$ and $rev(y_r)$ is immediate (as shown in Theorem 2). In case that the revenue expected by reaching y_r is higher then the one expected by reaching x_r, traversing to $y=preserve\_desc(x)$[3] is preferable. That is, it is better to preserve the number of units offered. Otherwise, unless y is a terminating state, it is necessary to check whether y can lead to a better revenue than $rev(x_r)$ could. This means

[3] $preserve(x)$ appears as $preserve\_desc(x)$ in Figure 7.

that it is necessary to compute the optimal strategy as if y is the current state (x←y). Since we already know the value of $depth(y_r)$ and $rev(y_r)$, there is no need to perform the first step of the algorithm, which is time-consuming. The number of states computed along the execution of the algorithm is composed of $(l_{max}\text{-}l_0)/\Delta$ states computed at the initial phase of the algorithm and $(l_{max}\text{-}\ l_0)/\Delta$ states computed until reaching a terminating state. Summing up the number of states we compute during the algorithm we find out that this number is linear bounded by $O((l_{max}\text{-}l_0)/\Delta)$. This result proves Theorem 3.

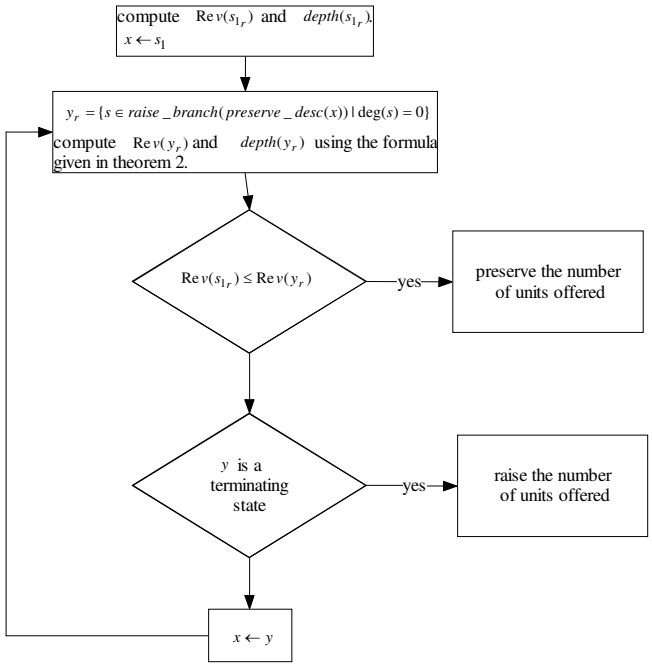


Figure 7. Illustration of step 3 of the informed decision strategy algorithm.

5. EXPERIMENTAL EVALUATION

The theoretical model presented above was tested experimentally to assess its added value compared to other techniques for running auction sites in the context of conducting expanding auctions.

Our simulation captured the dynamics of the expanding auction mechanism discussed above. Two strategies have been tested with such auctions:

- The informed decision strategy we propose.
- A post optimal decision strategy – this strategy is equivalent to the IDS except that here instead of sampling a MBV vector based on statistical knowledge gathered, the strategy considers the real MBV generated for each bidder participating in the auction. This strategy is superior compared to the IDS and serves as a measure for evaluating the quality of the solution offered by the IDS.

Both strategies have been tested in two types of environments: a homogenous environment in which bidders participating are sharing the same probability distribution function, and; heterogeneous environment in which the probability distribution function of each bidder is distinct.

For each setting, the parameters of the experiment were set to allow producing statistically significant results. For example, for an experiment in which the bidders' MBV distribution has a low variance to expectation ratio, 1000 trials were sufficient to arrive at statistical significance. Testing the same feature in the case of a high variance vs. expectation ratio required 4000 trials to produce stable results.

We have conducted a series of experiments in which all bidders originated from a homogenous distribution function; the variance/expectation ratio of the distribution function was set to 0.25. The number of bidders participating in the auction was set to 100. The results show that the IDS gains 70-80 percent of the post-optimal strategy in terms of revenue. Increasing the variance/expectation ratio shows degradation in the results compared to the post-optimal strategy. This is expected as the uncertainty of the sample MBV vectors produced increases.

The experiments with homogeneous distribution functions served as a reference. We have also performed experiments where we mixed homogenous and heterogeneous bidders. The results obtained when conducting experiments in mixed environments were highly dependent on the variance/expectation ratio, the portion of bidder types participating in the auction and the bidder selection distribution function. Interestingly, for experiments where the mutual variance/expectation ratio was set to 0.25, the portion of bidder types was 50% of each, and the bidder selection distribution function was uniform, the results can be computed from results of other experiments. In particular, one can average the results of similar experiments performed on fully homogenous and fully heterogeneous bidder populations.

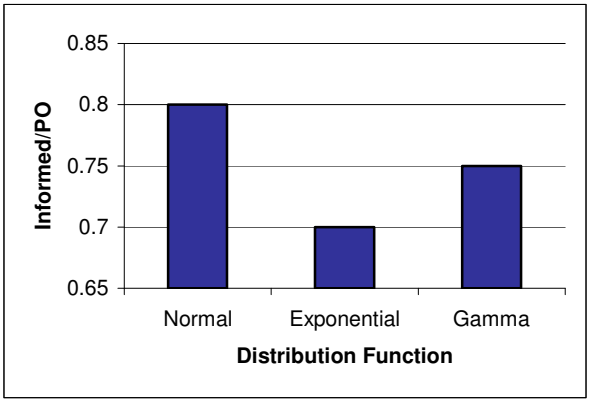


Figure 8. Informed Decision Strategy vs. Post-optimal strategy (variance/expectation ratio=0.25).

In a set of experiments, using settings as stated above, we have performed experiments in which normally distributed bidders, exponentially distributed bidders and Gamma distributed bidders participate in the same auction. The expected revenue arrived at was 0.7-0.8 of the post-optimal results (cf. Figure 8).

Note that we did not perform experiments that compare our strategy to the naive step strategy currently used in web implementations of the expanding auction. Such experiments could have further exemplified the advantage of our strategy. However, the reason for not performing such experiments is that, as we have shown by example, the step function can lead to results far from optimal. In contrast, because our strategy selects the best possible action given the knowledge available, it would always outperform the step strategy.

6. CONCLUSION

The expanding auction mechanism discussed in this paper is different from other mechanisms, such as the ascending clock mechanism or the GVA mechanism, in the sense that it does not require that the auctioneer set the number of units he/she is willing sell before the auction starts. This property makes the expanding multi-unit auction mechanism suitable for B2C commodities sales, where the auctioneer might prefer flexibility in the number of units offered to the participants.

The study performed here introduces a practical solution to the management of expanding multi-unit auctions. The informed decision strategy proposed was proven, theoretically and empirically, to address computation and time constraints that must be met by real life auctions. It was shown that the optimal strategy for searching the problem space—seemingly exponentially bounded—can be solved in linear time. The naïve step function strategy used in B2C expanding auction sites provides a sub-optimal mechanism to establish the price levels in which the auctioneer should raise the number of units offered. The revenues obtained by our new strategy outperform this naïve strategy.

Empirical results have shown that under diverse probability distribution functions the informed decision strategy we introduced produces significant gains. The auctioneer's expected revenues constitute 70%-80% of the post-optimal revenues. This is an excellent result given the probabilistic nature of the information used by our strategy. Future research needs to examine the effect of rational bidder strategies on our result. Regardless, IDS is very useful by itself, as bidders in contemporary dynamic sales sites hardly implement rational bidding.

7. REFERENCES

[1] www.olsale.co.il/olsale/rules.aspx?type=1

[2] L. Ausubel, "An Efficient Ascending-Bid Auction for Multiple Objects", Working Paper 97-06, U. Maryland, 1997.

[3] L. Ausubel, "A Generalized Vickrey Auction", Working Paper, University of Maryland, 1999.

[4] P. Cramton. Ascending Auctions, European Economic Review 42:3-5, 745-756, 1998.

[5] A. Goldberg, J. Hartline, and A. Wright. Competitive Auctions and Digital Goods. In Proc. 12th Symp. On Discrete Algorithms, 735–744. ACM/SIAM, 2001.

[6] P. Klemperer. Auction Theory: A Guide to the Literature. Journal of Economic Surveys, Vol. 13(3) pp. 227-286, 1999.

[7] E. Maskin, J. Riley, "Open and Sealed-Bid Auctions: Auction Theory with Private Values," American Economic Review, 75, 150-155, 1985.

[8] P. Milgrom, R. Weber, "A Theory of Auctions and Competitive Bidding", Econometrica, 50, 1089-1122, 1982.

[9] R. Myerson, "Optimal Auction Design," Mathematics of Operations Research, 6, 58-73, 1981.

[10] S. Park, E. Durfee, W. Birmingham, "An Adaptive Agent Bidding Strategy Based on Stochastic Modeling", Proc. of the Third International Conference on Autonomous Agents, 1999.

[11] O. Shehory, E. Dror, "Optimizing Auctioneer's Revenues in Expanding Multi-Unit Auctions", Proc. AAMAS 2005.

[12] H. Varian, "Economic Mechanism Design for Computerized Agents", Working Paper, U. California, 1995.

[13] W. Vickrey, Counterspeculation, Auctions and Competitive Sealed Tenders. J. of Finance, 16, 8-37, 1961.

A Robust Combinatorial Auction Mechanism against Shill Bidders

Tokuro Matsuo
Graduate School of Engineering, Nagoya Institute of Technology
tmatsuo@ics.nitech.ac.jp

Takayuki Ito*
CCS, Sloan School of Management, Massachusetts Institute of Technology
takayuki@mit.edu

Robert W. Day
School of Business in University of Connecticut
Bob.Day@business.uconn.edu

Toramatsu Shintani
Graduate School of Engineering, Nagoya Institute of Technology
tora@ics.nitech.ac.jp

ABSTRACT

This paper presents a method for discovering and detecting shill bids in combinatorial auctions. The Vickrey-Clarke-Groves Mechanism is one of the most important combinatorial auctions because it can satisfy the strategy-proof property, individual rationality, and Pareto efficiency, that is, it is the only mechanism that simultaneously satisfies these properties. As Yokoo et al. pointed out, false-name bids and shill bids pose an emerging problem for auctions, since on the Internet it is easy to establish different e-mail addresses and accounts for auction sites. Yokoo et al. proved that VCG cannot satisfy the false-name-proof property, and they also proved that no auction protocol can satisfy all three of the above properties and the false-name proof property simultaneously. Their approach concentrates on designing a new mechanism that has desirable properties, but this is quite complicated. As a new approach against shill-bids, in this paper, we design a mechanism that utilizes VCG and an algorithm for finding potential shill bids. Our mechanism is quite simple compared with Yokoo's approaches [11][12][13]. Our mechanism can judge whether there might be a shill bid from the results of the VCG procedure. We prove a theorem stating that shill bidders cannot increase their utilities unless all shill bidders win in the auction. Based on this theorem, our proposed mechanism compares the agents' utilities in a conventional auction with those in an auction where a shill bidder does not join in the auction. When these agents' utilities are different between the above cases, such agents might be shill bidders. Then, our mechanism allocates items to the shill bidders as a group from the set of items obtained through successful bids by the agent in the conventional auction. This process prevents shill bidders from increasing unfair profits. Furthermore, even though shill bidders participate in the auction, the seller's profit does not decrease using our proposed method. Thus, our mechanism detects shill bids when it only detects the possibility of shill bids. Our proposed method has the following three key advantages. First, we propose a method to detect shill bidders by comparison between bidders utilities. Our method is superior than existing complex mechanisms in the point of view of generalization and wide-use, because our auction mechanism employs only VCG. Second, even though there are shill bidders in an auction, incentive compatibility property is preserved using our mechanism. Finally, the schemer, in our mechanism, does never have incentive to make shill bidders. The schemer's utility does not increase in our mechanism even though a schemer make shill bidders. Namely, not to make shill bidders is dominant strategy for the schemer.

Categories and Subject Descriptors

I.2.11 [**Multiagent Systems**]: Design, Theory, and Electronic Commerce

Keywords

Computational Mechanism Design, Combinatorial Auctions, Shill Bids, Vickrey-Clarke-Groves Mechanism, Incentive Compatibility and Pareto Efficiency.

1. INTRODUCTION

This paper presents a method for detecting shill bids and for the allocation of items in combinatorial auctions. Auction theory has received much attention from computer scientists and economic scientists in recent years [7]. One reason for this interest is the fact that Internet auctions such as Yahoo Auction and eBay have developed very quickly and widely. Also, auctions in B2B trades are increasing rapidly. Moreover, there is growing interest in using auction mechanisms to solve distributed resource allocation problems in the field of AI and multi-agent systems. Combinatorial auctions have been studied very widely as one of most important auction formats. In a combinatorial auction, bidders can make bids on multiple bundles of different items[9].

The Vickrey-Clarke-Groves (VCG) Mechanism is one of the combinatorial auction mechanisms that are strategy-proof, i.e., the dominant strategy is for bidders to declare their true evaluation value for a good, and its allocation is Pareto efficient. Many scientists in the field of auction

*He is a Visiting Scholar from Nagoya Institute of Technology, Japan, and also a Visiting Scholar in Harvard University, USA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

theory have focused on VCG because of its strategy-proof property. The details of VCG are described in Section 2.

As Yokoo et al. pointed out, false-name bids and shill bids are an emerging problem for auctions, since on the Internet it is easy to establish different e-mail addresses and accounts for auction sites [13]. Bidders who make false-name bids and shill bids can benefit if the auction is not robust against false-name bids. A method to prevent false-name bids and shill bids is a major issue that auction theorists need to resolve.

Yokoo et al. proved that VCG cannot satisfy the false-name-proof property [13]. Moreover, they proved that there is no auction protocol that can satisfy all of the three major properties, i.e., false-name-proof property, Pareto efficiency, and strategy-proof property. They have developed several other complicated auction protocols that can satisfy at least the false-name-proof property. Their approach is to find alternative auction protocols and mechanisms. However, satisfying the false-name proof property prevents them from developing Pareto efficient and strategy-proof protocols.

On the contrary, in this paper we propose a new approach to detecting shill bidders as well as an allocation method in combinatorial auctions. Even though there exist shill bidders, our method can employ the VCG mechanism without using complicated mechanisms such as the protocol proposed by Yokoo et al. Our mechanism can determine whether there might be a shill bid from the results of the procedure of a VCG mechanism.

The mechanism proposed in this paper consists mainly of the following two steps. First, our mechanism judges the possibility of shill bidders based on the result of an auction. Auctions where some bidders bid their valuations as zero are attempted, and other bidders' utilities are calculated in this situation. The utility of each bidder is compared between a conventional VCG and the altered outcome. When there are bidders whose utilities change with the situation, our mechanism judges that such bidders might be shill bidders. Then, items are allocated to successful bidders. However, in this step, the following method of allocation is used, the items are allocated to bidders who are not shill bidders. On the other hand, some items are allocated as a set to a group of shill bidders and bidders who might be shill bidders, for a merged valuation function. This prevents shill bidders from making unfair profits and decreasing total revenue for sellers. Namely, in our approach, we build a mechanism to find shill bids in order to avoid them in VCG. This differs from the approach of Yokoo et al., which builds mechanisms to avoid shill bids. In Yokoo's approach, they have been trying to design a different mechanism that can satisfy desired properties (e.g. false-name proofness). However, the resulting mechanisms have been complicated due to the difficulty of this approach. On the other hand, our mechanism detects shill bidders and regards them as a single bidder. In our mechanism, VCG is employed without any complicated modifications. This gives our mechanism more generic usefulness since VCG is a well known protocol compared with Yokoo's proposed protocols.

A shill bid is defined as two or more bids created by a single person, who can unfairly gain a benefit from creating such bids. The straightforward way to find shill bids is to find a bidder whose utility becomes negative when his/her bids and those of another bidder are merged. The merging method is described in Section 3. However, this straightforward method requires an exponential amount of computing power, since we need to check all possible combinations of bidders. Thus, we use a theorem involving the characteristics of shill bids and propose a new method for detecting shill bidders. The theorem shows that a group of shill bidders cannot increase their utilities when a shill bidder cannot be a successful bidder. Using this method, the computational costs are reduced dramatically compared with the merge investment method. Comparing bidders utilities between the normal VCG and the above method, the group of shill bidders is detected. Our mechanism allows a set of items to be allocated to a shill bidders group for merged investments. In addition to detecting shill bidders, the VCG's properties are preserved. Further, a seller's revenue never decreases, even though shill bidders bid in the auction.

Our proposed method has the following properties and advantages. First, we propose a method to detect shill bidders by comparison between bidders utilities. Our method is superior than existing complex mechanisms in the point of view of generalization and wide-use, because our auction mechanism employs only VCG. Second, even though there are shill bidders in an auction, incentive compatibility property is preserved using our mechanism but efficiency is not preserved in some cases. Finally, the schemer, in our mechanism, does never have incentive to make shill bidders. The schemer's utility does not increase in our mechanism even though a schemer make shill bidders. Namely, not to make shill bidders is dominant strategy for the schemer.

The rest of this paper consists of the following six parts. In Section 2, we give preliminaries, explaining several terms and concepts of auctions. In Section 3, we explain shill bids and show the characteristics of shill bidding in VCG. In Section 4, the method of detecting shill bidders and allocating items used in this paper is introduced. In Section 5, discussions are presented to show the properties of our mechanism. Finally, we give our concluding remarks.

2. PRELIMINARIES

2.1 Model

Here, we describe the model and definitions needed for the present work. The auction participants of trading consist of a seller and bidders. The seller prepares multiple items for sale, and bidders place bids for bundles they want to purchase.

- In an auction, we define that a set of bidders/agents is $N = \{1, 2, \ldots, i, \ldots, n\}$ and a set of items is $A = \{a_1, a_2, \ldots, a_k, \ldots, a_m\}$.
- $v_i^{a_k}$ is bidder i's value, at which the ith bidder bids for the kth item ($1 \leq i \leq n$, $1 \leq k \leq m$).
- $v_i(B_i^{a_k,a_l})$ is bidder i's value, at which the ith bidder bids for the bundle including the kth and lth items ($1 \leq i \leq n$, $1 \leq k, l \leq m$). This notation is used when a bidder evaluates more than two items.
- $p_i^{a_k}$ is the payment when agent i can purchase an item a_k. When bidder i purchases the set of bundles of items, the payment is shown as $p_i(B_i^{a_k,a_l})$.
- G_i is the bundle of items allocated to agent i.

- The set of choices is $G = \{(G_1, \ldots, G_n) : G_i \cap G_j = \emptyset, G_i \subseteq G\}$.

ASSUMPTION 1 (QUASI-LINEAR UTILITY). *Agent i's utility u_i is defined as the difference between the value v_i of the allocated good and the monetary transfer p_i for the allocated good. $u_i = v_i - p_i$. Such a utility is called a quasi-linear utility.*

ASSUMPTION 2 (PRIVATE VALUE). *In this paper, we employ the assumption that the characteristics of items are private value for all bidders. Each agent's type is independent of the types of the other bidders.*

ASSUMPTION 3 (SINGLE-MINDED VALUATION). *We assume single-minded valuation, that is, bidders never bid as substitute values for each item.*

Namely, in this paper, when the number of items in a bundle increases, the total value for the bundle decreases. This means that free disposal is assumed.

Each bidder i has preferences for the subset $G_i \subseteq G$ of goods. Formally, each bidder has type θ_i, that is, in a type's set Θ. Based on the type, we show that the bidder's utility is $v_i(G_i, \theta_i) - p_i^{G_i}$ when the bidder purchases item G_i for $p_i^{G_i}$. Note that $(v_i(G_i, \theta_i))$ is bidder i's value for bundle $G_i \subseteq G$.

2.2 VCG: Vickrey-Clarke-Groves Mechanism

The Vickrey-Clarke-Groves mechanism [2][4][10] has some key properties, that is, it's strategy-proof and Pareto efficient if there exists no false-name bid. We say an auction protocol is Pareto efficient when the sum of all participants' utilities (including that of the auctioneer), i.e., the social surplus, is maximized. If the number of goods is one, in a Pareto efficient allocation, the good is awarded to the bidder having the highest evaluation value corresponding the quality of the good.

In the VCG, first each agent tells his/her value $v_i(G_i, \theta_i)$ to the seller. We omit the "type" notation and simply write $v_i(G_i, \theta_i) = v_i(G_i)$. The efficient allocation is calculated as an allocation to maximize the total value:

$$G^* = arg \max_{G=(G_1,\ldots,G_n)} \sum_{i \in N} v_i(G_i).$$

The auctioneer informs the payment amount to the bidders. Agent i's payment p_i is defined as follows.

$$p_i = \sum_{i \neq j} v_j(G^*_{\sim i}) - \sum_{i \neq j} v_j(G^*).$$

Here, $G^*_{\sim i}$ is the allocation that maximizes the sum of all agents' values other than that of agent i. This is the allocation in which the total value is maximum when all agents, other than agent i, bid their values:

$$G^*_{\sim i} = \arg \max_{G \setminus G_i} \sum_{N-i} v_j(G_j).$$

3. SHILL BIDS

3.1 Weakness against shill bids

In auction research, some papers have reported the influence of false name bids in combinatorial auctions, such as VCG [13]. These are called "shill bids." Here, we show an example of shill bids.

Assume there are two bidders and two items. Each agent bids for a bundle, that is, $\{a_1, a_2, (a_1, a_2)\}$.

Agent 1's value $v_1(B_1^{a_1,a_2}) : \{\$6, \$6, \$12\}$
Agent 2's value $v_2(B_2^{a_1,a_2}) : \{\$0, \$0, \$8\}$

In this case, both items are allocated to agent 1 for 8 dollars. Agent 1's utility is calculated as $12 - 8 = 4$.

If agent 1 creates a false agent 3, his/her utility increases.

Agent 1's value $v_1(B_1^{a_1,a_2}) : \{\$6, \$0, \$6\}$
Agent 2's value $v_2(B_2^{a_1,a_2}) : \{\$0, \$0, \$8\}$
Agent 3's value $v_3(B_3^{a_1,a_2}) : \{\$0, \$6, \$6\}$

Agent 1 can purchase item a_1 and agent 3 can buy item a_2. Each agent's payment amount is $8 - 6 = 2$, and each agent's utility is calculated as $6 - 2 = 4$. Namely, agent 1's utility is 8 dollars (because agent 3 is the same as agent 1).

Yokoo et al. examined the effect of false-name bids on combinatorial auction protocols [13]. False-name bids are bids submitted by a single bidder using multiple identifiers such as multiple e-mail addresses. They showed a formal model of combinatorial auction protocols in which false-name bids are possible. The obtained results can be summarized as follows: (1) the Vickrey-Clarke-Groves (VCG) mechanism, which is strategy-proof and Pareto efficient when there exists no false-name bid, is not false-name proof; (2) there exists no false-name-proof combinatorial auction protocol that satisfies Pareto efficiency; (3) one sufficient condition where the VCG mechanism is false-name-proof is identified, i.e., the concavity of a surplus function over bidders.

3.2 Schemer's utility

In this paper, a "schemer" is an auction participant who contrives shill-bids and creates "shill bidders". When shill bidders are not successful bidders, the schemer's utility never increases. Namely, the schemer cannot increase his/her utility by using a "decoy" bidder. The schemer's utility does not increase unless all shill bidders and the schemer become successful bidders. The following theorem shows the characteristics of shill-bids.

THEOREM 1 (SHILL-BIDDERS MUST BE WINNERS). *An agent who creates shill bidders cannot increase his/her utility unless a shill bidder wins.*

[Proof] We prove that bidder agent i's utility u_i does not increase when the agent's shill bidder does not win in an auction. When bidder i does not create any shill bidders, bidder i's payment p_i can be illustrated as $p_i = \sum_{i \neq j} v_j(G^*_{\sim i}) - \sum_{i \neq j} v_j(G^*)$. When the bidder agent creates shill bidders, bidder i's payment p'_i is $p'_i = \sum_{i \neq j} v_j(G'^*_{\sim i}) - \sum_{i \neq j} v_j(G'^*)$.

Here, we show $p'_i \geq p_i$, assuming that bidder i's shill bidders do not win in the auction. Also, the set of allocations in the auction does not change. Namely, $G' = G \not\ni s$ holds. The difference between p'_i and p_i is shown as follows. $p'_i - p_i = \sum_{i \neq j} v_j(G'_{\sim i}) - \sum_{i \neq j} v_j(G') - (\sum_{i \neq j} v_j(G_{\sim i}) - \sum_{i \neq j} v_j(G)) = \sum_{i \neq j} v_j(G'_{\sim i}) - \sum_{i \neq j} v_j(G_{\sim i})$.

The objective function of the winner determination problem solved over a strictly larger set of bids (that is, one with additional shill bids) cannot decrease. By adding bids the winner determination value never goes down. Thus, shill bidders' utilities do not increase when the shill bidder does not win.

□

3.3 Characteristics of shill bids

Shill bids are characterized by the fact that a successful shill bidder's payment amount is decided based on the other shill bidders and the schemer. Namely, the schemers get unfair profits from the shill bidders' bids because the shill bidders' valuations are used in calculating the schemer's payment amount.

Here, we give the following two lemmas characterized by shill bidding.

LEMMA 1 (SHILL BIDDER'S UTILITY). *A shill bidder's utility cannot be independent of the values bid by other shill bidders.*

[Proof] Essentially, the lemma can be proven based on the definition of payment in the VCG mechanism. We give an example of this situation. Using the example of the section 2, we consider the situation where bidder 3's valuation changes. The shill bidders group includes bidders 1 and 3. Each bidder bids his/her value as follows.

$v_1(B_1^{a_1,a_2}) : \{\$6, \$0, \$6\}$
$v_2(B_2^{a_1,a_2}) : \{\$0, \$0, \$8\}$
$v_3(B_3^{a_1,a_2}) : \{\$0, \$4, \$4\}$

Bidder 1's payment amount can be calculated as $8 - 4 = 4$ for item 1 and bidder 3's payment amount is $8 - 6 = 2$ for item 3. Thus, the schemer's (bidder 1's) utility can be computed as \$6. In section 2, the schemer's utility is \$8. On the other hand, in this situation, the schemer's utility has a different value. Namely, when shill bidders' valuations for items change, the bidders' payoff in the same shill-bid group changes. A shill bidder's payment is not independent of the other shill bidders' or the schemer's value. □

LEMMA 2 (SHILL BIDDER'S PAYMENT). *Each shill bidder's payment amount is calculated based on the valuations of the other shill bidders in the shill-bid group.*

[Proof] This lemma can be proven easily based on the definition of VCG. In VCG, a successful bidder's payment is shown as $p_i = \sum_{i \neq j} v_j(G^*_{\sim i}) - \sum_{i \neq j} v_j(G^*)$. The $\sum_{i \neq j} v_j(G^*_{\sim i})$ in the right side of the equation means the sum of the winners' total valuations except for bidder i's valuation. This first member of the equation includes all other shill bidders' values. This means that each shill bidder's payment amount is calculated based on the other shill bidders' valuations in the same shill-bid group, as described above. Thus, a shill bid can be characterized by its payment amount being interdependent with other shill bidders' valuations. □

In VCG, bidder i's payment amount is based on the bidders' values in the bidder set $N \setminus i$ in equation as shown in section 2.2. On the another hand, bidder i's valuation is used in calculating the other shill bidders' payment amounts. Let us consider the situation where a schemer does not create shill bidders. In this situation, the schemer's payment increases compared with the situation where there are shill bidders, since there is no shill bidder who can influence to the schemer's payment. Intuitively, we can detect shill bidders by merging investments among winners. We can determine the possibility of shill bidders by comparing the utilities between a calculation using an individual agent's value and one using the merged values. Figure 1 shows a conception of the merged value method. Winners' utilities under normal VCG are reduced than the utilities under the merged investment. Thus, the merged investment method is one of appropriate methods for detecting shill bidders.

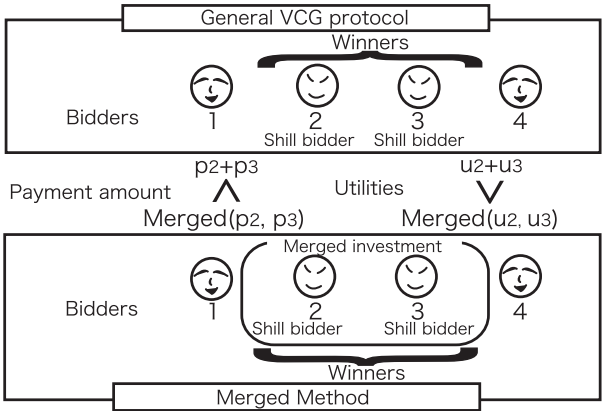


Figure 1: Merged Investment

4. DETECTING SHILL BIDDERS AND ALLOCATING ITEMS

4.1 Merged investment

Shill bidders are detected based on a comparison between bidders' utilities. Concretely, first, each bidder's value is merged with other bidders' values. An auction is conducted under the condition of merged investment. The merged bidder's utility is computed in this situation and compared with the sum of each bidder's utility in general VCG. If the bidder's utility is reduced in the merged case, such a bidder may be guilty of shill bidding. Thus, we construct an artificial situation where a shill bidder cannot influence the other shill bidders through collusion, and then we can detect the shill bidders in VCG. The next section describes our method of detecting shill bidders in detail.

A feature of shill bids is that the scheming agent can increase his/her utility when he/she bids the values divided into multiple bids. We show a straightforward method to determine the possibility of shill bidding by using the winners' values in a combinatorial auction. Shill bidders are created by separating the bids of an original agent. Namely, the original agent can increase utility by dividing his/her bidding actions. When the original bidder and his/her shill bidders are winners of an auction, we can determine how much the agent increases the utility by comparing the utilities of divided bidding with those of merged investment bidding. Intuitively, we apply a winner-based algorithm, which is based on a comparison of utilities between bidding in real auctions and bidding based on our method. Below, we describe our proposed algorithm in detail.

Input: Bids on bundles from each participant.
Output: True if there is a potential shill bid.
False if there is no potential shill bid.

Function Detect a Shill bid
begin
Determine winners and calculate
payments based on VCG.
Create a power set S for a set of participants.
for each $s \in S$
Merge participants' values in s

by merge function $f(s)$.
Determining winners and calculating payments
based on merged values by VCG.
$u_{f(s)}$:= the utility of s after merging.
u_{sum_s} := sum of the utilities in s before merging.
if $u_{f(s)} < u_{sum_s}$
return True
else
return False
end.

In the above method, the number of merged investments is calculated as $2^n - n - 1$ when the number of winner is n, namely the computational cost is $O(2^n)$. Therefore, a huge computational cost is required to judge whether there are shill bidders in the auction. Consequently, we propose a new method for detecting shill bidders. Based on [Theorem 1], a schemer's utility decreases if a shill bidder does not win in the auction, since the shill-bid scheme fails. Assume that there are shill bidders i and j. Bidder i's utility decreases if bidder j does not join the auction because shill bidder j's valuation is used in calculating i's payment. Using this method, the number of computations is just $n - 1$ if all bidders win in the auction.

4.2 Method of detecting shill bidders

Based on [Theorem 1], shill bidders cannot increase their utilities unless their confederates win the auction. Then, how can we detect shill bidders by using this feature? We tentatively try each auction where winners are excluded, one by one. Figure 2 shows a conception of our new proposed method. If there are shill bidders, such bidders' utilities are reduced by the above procedure through comparison between a normal auction and the trial auctions. Thus, shill bidders are detected by comparing the differences in their utilities.

Here, we give an outline of our proposed mechanism. First, an allocation is calculated and successful bidders' utilities are computed in VCG. Second, bidders' utilities are calculated after attempting auctions in which each winner is excluded one by one. Third, each bidder's utility is compared between the first step and the second step. The bidders whose utilities change are classified into a shill-bidder group. Then, the auction is retried among innocent bidders and the shill-bidder group. The bundle is allocated as a set to the shill-bidder group. The reason why we allocate items as a set in this method is that a shill-bidder group sometimes includes false-shill bidders. False-shill bidders, who bid unexpectedly in the manner of shill bidding, are innocent. In this paper, we focus on preventing shill bidding and fraud. We permit the mechanism to allocate items to the shill-bidder group to cope with shill bidding, but we never employ the traditional auction method in which items are allocated to each bidder. Thus, our mechanism cannot only detect shill bidders but also preserve the advantages of VCG.

Here, we concretely explain the procedure of our mechanism.

[**Algorithm**] (**Step 1**) Assume that n bidders join the auction and that the winners are determined based on VCG. Items are allocated to the successful bidders in a combination such that the total of all bidders' valuations is maximum. $G^* = arg\max_{G=(G_1,\ldots,G_n)} \sum_{i \in N} v_i(G_i)$. The successful bidder i's payment amount p_i is calculated based on

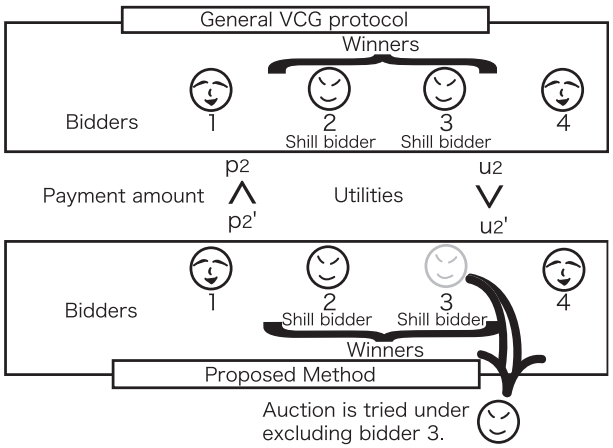


Figure 2: Proposed Method

the following equation. $p_i = \sum_{i \neq j} v_j(G^*_{\sim i}) - \sum_{i \neq j} v_j(G^*)$. $G^*_{\sim i}$ is the combination where all bidders' valuations are maximized except for bidder i's valuation. This allocation is defined as follows. $G^*_{\sim i} = \arg\max_{G \setminus G_i} \sum_{N-i} v_j(G_j)$.

(**Step 2**) Winners' utilities are calculated. We simply define that a set of successful bidders is shown as $W = \{1, 2, \ldots, \iota, \ldots, \theta\}$. Bidder ι's surplus is u_ι.

(**Step 3**) An auction in which a bidder θ does not participate is tried. In this auction, the allocation is described as $G^*_{\sim\theta} = arg\max_{G=(G_1,\ldots,G_n)} \sum_{N-\theta} v_i(G_i)$. The set of winners is $W = \{1, 2, \ldots, \iota, \ldots, \theta - 1\}$. In this allocation, each bidder's utility is compared with his/her utility at the allocation in (**Step 1**).

(**Step 4**) The difference in the winners utilities, $u_\iota - u_\iota^{\sim\theta}$, is calculated. $u_\iota^{\sim\theta}$ is winner ι's utility in an auction that bidder θ does not join. A set of winners $\{w_{\sim\iota}\} \in W$ is detected under $u_\iota - u_\iota^{\sim\theta} > 0$.

(**Step 5**) In (**Step 4**), a winners group $(\{w_{\sim\iota} \setminus \theta\} \in W) = (\{w_{\sim\theta} \setminus \iota\} \in W)$, whose utilities are reduced, is established. Items are allocated to the winners except for bidders in the above winners group (shill bidders), that is, $W \setminus w$.

(**Step 6**) Winners w's values are merged. The payment amount is calculated in the merged investment. For this payment, items are allocated as a set to the shill bidders.

4.3 Example

In this section, an example of our proposed mechanism is shown in the case where five bidders participate in an auction for two items. In this situation, assume that there are a schemer and a shill bidder created by the schemer. Bidders bid their valuations for each item and bundle as follows.

$v_1(B_1^{a_1,a_2})$: {\$8, \$7, \$15}
$v_2(B_2^{a_1,a_2})$: {\$7, \$8, \$15}
$v_3(B_3^{a_1,a_2})$: {\$12, \$0, \$12}
$v_4(B_4^{a_1,a_2})$: {\$0, \$12, \$12}
$v_5(B_5^{a_1,a_2})$: {\$0, \$0, \$20}

In this round, the successful bidders are 3 and 4. Each bidder's payment amount is 8. Namely, u_3 and u_4 are both 4 dollars.

Here, an auction where bidder 3 does not participate is tried based on our proposed mechanism to detect shill bidders. In this case, the winners are bidder 5 or bidders 1 and

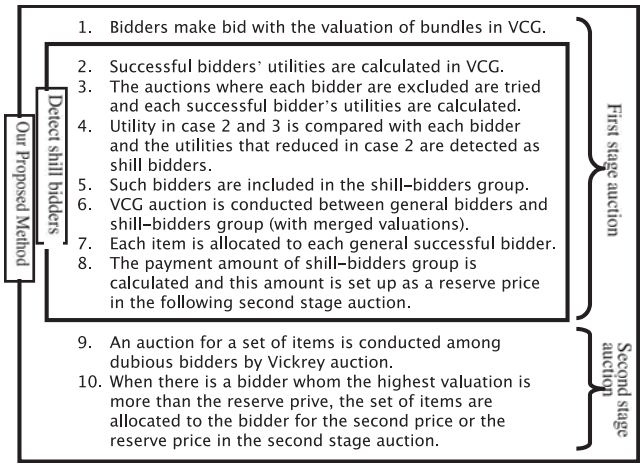


Figure 3: Proposed mechanism

4. Bidder 4's payment amount is 12 dollars. u_4 is 0. Bidder 4's utility is reduced compared with the normal auction.

Similarly, there is the same result when bidder 4 is excluded from a set of bidders. In this case, the winners are bidder 5 or bidders 2 and 3. Bidder 3's payment amount is 12 dollars. u_3 is 0. Bidder 3's utility is reduced compared with the normal auction. Bidders 3's and 4's utilities decrease after the integration of valuations.

In an allocation, bidders 3's and 4's valuations are merged by a certain merge function F. For example, we assume that $v_{f(i,i+1,\ldots,j)}$ is the following maximum function among agents' values for each item. Merged value $v_{f(i,i+1,\ldots,j)}$ is shown as $(\max_{i,i+1,\ldots,j}\{v_i^{a_1}\}, \max_{i,i+1,\ldots,j}\{v_i^{a_2}\},\ldots,$ $\max_{i,i+1,\ldots,j}\{v_i(B_i^{a_1,\ldots,a_m}\})$ for agent i's value $(v_i^{a_1}, v_i^{a_2},\ldots,$ $v_i(B_i^{a_1,\ldots,a_m}))$ in the maximum selection method. In this example, the merged valuation $v_{F(3,4)}$ for bidders 3 and 4 is $\{12, 12, 24\}$. Using this merged valuation, the auction is retried. The integrated bidders group's payment amount $p_{F(3,4)}$ is calculated as $20-0=20$. Both items are allocated to the merged bidders for \$20 as a set.

4.4 Robust auction mechanism against shill bidders

Figure 3 shows an outline of our proposed mechanism. The mechanism is distinguished between two stages. In the first stage, VCG auction is conducted. and shill bidders are detected as shown in the above subsections. Items are allocated to general bidders who are no possibility of shill bidder. After that, the ramained set of items are allocated to shill-bidders group. Here, reserve price is calculated as a result of first stage auction. In the second stage, each dubious bidder as a shill agent bids to a set of items like a Vickrey auction. A bidder who bids the highest valuation makes a successful bid when the valuation is more than the reserve price.

In our mechanism, there are two situation concerned with efficiency. When the set of items is allocated successfully to the bidder for the bidder's reserve value or less, the allocation is Pareto efficient. On the other hand, the false-name proof and strategy-proof property is preserved based on this method.

Here, we give the concrete allocation method. The method employs like the single item auction that is, each bidder bids only one bid value to the set of items. The set of items is allocated based on the second highest value. Thus, the method is essentially same as the Vickrey auction.

We assume that there are bidders $w = \{i'_1, \ldots, i'_\iota, \ldots, i'_n\}$ in shill bidders group. Bidders' valuations are shown as $\{v_{i'_1}, \ldots, v_{i'_\iota}, \ldots, v_{i'_n}\}$. Bidders' payments included in the shill bidders group is shown as $\{p_{i'_1}, \ldots, p_{i'_\iota}, \ldots, p_{i'_n}\}$. Bidders' utilities is shown as $\{u_{i'_1}, \ldots, u_{i'_\iota}, \ldots, u_{i'_n}\}$. We decide a reserve price based on the price (such as the end of subsection 4.3) allocated to the shill bidders group at the first stage. The set of items is shown as $A' \subset A$.

[Algorithm] (**Step 1**) Bidders who are included in the shill bidders group bid their valuations $v_{i'_\iota}$ for the set of items A'.

(**Step 2**) We assume simply higher order of bidders valuations is shown as $\{v_{i'_1} > v_{i'_2} > \ldots > v_{i'_\iota} > \ldots > v_{i'_n}\}$. If the bidder's valuation $v_{i'_2}$ is not more than the reserve price, items are not allocated to the winner. Otherwise, (**Step 3**) is employed.

(**Step 3**) If the bidder's valuation $v_{i'_2}$ is more than the reserve price, the winner of this auction is decided based on the second highest price which is bid by a bidder in the shill bidders group. Namely, bidder i'_1's payment amount $p_{i'_1}$ is $v_{i'_2}$.

5. DISCUSSION

5.1 Properties of protocol

Here, we give a discussion on the properties and validity of our mechanism. Our mechanism preserves the desirable advantages of VCG, that is, Pareto efficiency, individual rationality, and incentive compatibility. First, Pareto efficiency is proven self-evidently because items are allocated in a combination where participants' total utilities are maximum in VCG. The individual rationality property can be shown simply by the definition of VCG. Here, we give the following theorem to prove the incentive compatibility property of our mechanism.

THEOREM 2 (STRATEGY-PROOF PROPERTY). *Incentive compatibility is preserved in our mechanism.*

[Proof] In literature [13], Yokoo et al. proposed a flamework for mechanism design called PORF protocol, that is, designed mechanism is satisfied incentive compatibility property. Mechanisms based on PORF's characteristics are incentive compatible mechanism and strategy-proof mechanism. Our proposed mechanism has the PORF's feacture as shown in the following explanations.

First, we give some definitions for explanations. Bidder i is a ith bidder who participates in an auction. Shill bidders (including innocent bidders in the shill bidders group) are shown as i'. Shill bidders group is shown as w, that is, i' is included in w. In our proposed method, all bidders' payments and shill bidders' payments included in shill bidders group are not depend on their own bid valuations. Thus, our mechanism is satisfied the strategy-proof property.

Second, the bidders i's payment amount is calculated based on VCG. On another hand, the payment amount for shill bidder i' is decided based on the reserve price $r(= p_w)$ in the second stage. The reservation value r is independent with the valuations included in the sholl bidders group.

Finally, an allocation feasibility for bidder i is reserved due to same as VCG. On the other hand, the allocation for shill bidder i' is reserved because items are allocated to a bidder who is in the shill bidders group.

As shown the above issues, our proposed mechanism is satisfied the PORF flamework. Even though a bidder's valuation who is not in the shill bidders gruop can give effects to shill bidders group's payment amount, the payment amount is not reduced by the valuation. □

In our proposed method, a schemer can not increase their utilities if he/she makes shill bidders compared with the situation where the schemer does not make shill bidders. This situation is same as normal VCG auction because shill bidders' values are merged. Considering situations in the real auctions, to make shill bidders puts some costs for the schemer. Thus, if the schemer does not make shill bidders, the schemer's strategy becomes dominant strategy.

When the set of items is allocated successfully, the allocation is efficient. On the other, if the bidder's valuation $v_{i'_2}$ is not more than the reserve price, the allocation is not efficient because items are remained unsold. The efficiency is related the number of bidders in shill-bidders group and the rate of false-shill bidders in the group. The efficiency is shown as a decreasing function with the rate of false-shill bidders in the group. Here, we give a proposition that a schemer never has an incentive to make shill bidders in our mechanism. Schemer's dominant strategy is not to make shill bidders.

ASSUMPTION 4 (STRATEGY SPACE). *We assume the following two strategies, that is, a schcemer makes shill bidders and does not make shill bidders.*

PROPOSITION 1 (DOMINANT STRATEGY). *Schemer's dominant strategy is not to make shill bidders in our mechanism.*

[Proof] After detecting shill bidders, second-price auction is conducted based on bidders' valuations and reserve price for the set of items. In this auction, if there is a winner, his/her payment amount is the reserve price r' or more than the price because the second-price should be more than the reserve price. Namely, the following holds. $p_{i'_1} = v_{i'_2} \geq r'$. The reserve price is essentially same as the payment amount in which bidders do not bid by shill bidding.

First, we consider a situation where shill bidders group is organized by only bidders made by the schemer. If the schemer does not make shill bidders, he/she can purchase items for r'. On the other hand, If the schemer make shill bidders, he/she purchase items for r' or more than the reserve value. Even though a shill bidder can make successful bid and his/her valuation is $v_{i'_1}$, his/her utility $u_{i'_1}$ is calculated as $v_{i'_1} - v_{i'_2}$. On the other hand, when the schemer does not make shill bidders, his/her utility $u_{i'_1}$ is calculated as $v_{i'_1} - r'$. Clearly, the following holds. $v_{i'_1} - v_{i'_2} < v_{i'_1} - r'$. Thus, the schemer, in this case, does never have incentive to make shill bidders.

Second, we consider a situation where shill bidders group is organized by innocent bidders (false-shill bidders) and shill bidders made by the schemer. In this situation, there are following two cases, (1) the case in which the schemer wins the second auction and (2) the case in which the schemer loses the second auction. $p_{i'_1} = v_{i'_2} \geq r'$ holds concerned with the schemer's payment amount. The schemer's utility $u_{i'_1}$ is calculated as $v_{i'_1} - v_{i'_2}$. On the other hand, when the schemer does not make shill bidders, his/her utility $u_{i'_1}$ is calculated as $v_{i'_1} - r'$. Clearly, the following holds. $v_{i'_1} - v_{i'_2} < v_{i'_1} - r'$. In the latter case, the schemer's utility is 0 because he/she can not purchase items. If the schemer does not make shill bidders, he/she can make a successful bid in the first stage auction(VCG). Clearly, the schemer's utility is $v_{i'_1} - r' (> 0)$.

Thus, the schemer, in this case, does never have incentive to make shill bidders. In any case, the schemer's utility does not increase in our mechanism. Namely, not to make shill bidders is dominant strategy for the schemer. □

5.2 Multiple schemes

Our mechanism can cope with multiple schemers in an auction. If there are multiple schemers, our mechanism detects multiple shill-bidder groups created by the schemers.

In our mechanism, some bidders may casually submit bid values that seem like unnatural shill bidders' valuations. Such bidders are included in the shill-bidder group. However, the goal of our protocol is to detect and prevent shill bidding. To solve this problem, our mechanism allocates items to a shill-bidder group as a set of items. When the shill-bidder group consists of all shill bidders, the result of the mechanism is the same as an auction in which the schemer does not commit unfair practices. Even if the shill-bidder group includes an innocent bidder, we can provide many types of methods of allocation and calculation of payment amount. In the shill-bidder group, items are allocated based on the method in which each bidder's payment does not depend on his/her valuation.

6. RELATED WORK

Milgrom analyzed the shill-biddable feature in VCG [7]. Bidders in VCG can profitably use shill bidders, intentionally increasing competition in order to generate a lower price. Thus, the Vickrey auction provides opportunities and incentives for collusion among low-value, losing bidders. This feature is a result of the monotonic increase problem. However, this work does not provide a method of detecting shill bidding in combinatorial auctions.

Yokoo et al. reported the effect of false-name bids in combinatorial auctions [13]. To solve the problem, Yokoo, Sakurai and Matsubara proposed novel auction protocols that are robust against false-name bids [11]. One such protocol is called the Leveled Division Set (LDS) protocol, which is a modification of the VCG. It utilizes the reservation prices of auctioned goods for making decisions on whether to sell goods in a bundle or separately. Furthermore, they also proposed an Iterative Reducing (IR) protocol that is robust against false-name bids in multi-unit auctions [12]. The IR protocol is easier to use than the LDS, since the combination of bundles is automatically determined in a flexible manner according to the declared values of agents. They concentrated on designing mechanisms that could perform as alternatives to VCG. Due to our fundamentally different purpose, we do not simply adopt off-the-shelf methods for mechanism design.

Ausubel and Milgrom referred and characterized shill bidding [1]. They showed a theorem in which the substitute condition is sufficient for shill bidding and loser collusion to be unprofitable. Further, if the set of bidder values is otherwise sufficient inclusive, the condition is also necessary for the theorem. However, in our proposed method, we do not

only show the features of shill bidding, but we also give a concrete mechanism for detecting shill bidding.

Some researchers [3][5][6][8][9] have proposed methods for computing and calculating the efficient solution in combinatorial auctions. These analyses contributed to the pursuit of a computational algorithm for winner determination in combinatorial auctions, but they did not deal with shill bidding and thus are fundamentally different approaches from our work. However, some of these algorithms can be incorporated in our work. Combinatorial auctions have a computationally hard problem in which the number of combinations increases when the number of participants/items increases in an auction, since agents can bid their values as a set of bundled items.

Sandholm [8] proposed a fast winner determination algorithm for combinatorial auctions. Sandholm et al. [9] also showed how different features of a combinatorial market affect the complexity of determining the winners. These works studied auctions, reverse auctions, and exchanges, with one or multiple units of each item, with and without free disposal. We theoretically analyzed the complexity of finding a feasible, approximate, or efficient solution.

Fujishima et al. proposed two algorithms to mitigate the computational complexity of combinatorial auctions [3]. Their Combinatorial Auction Structured Search (CASS) algorithm determines efficient allocations very quickly and also provides good "any-time" performance. Their second algorithm, called Virtual Simultaneous Auction (VSA), is based on a simulation technique. CASS considers fewer partial allocations than the brute force method because it structures the search space to avoid considering allocations containing conflicting bids. It also caches the results of partial searches and prunes the search tree. On the other hand, VSA generates a virtual simultaneous auction from the bids submitted in a real combinatorial auction and then carries out simulation in the virtual auction to find a good allocation of goods for the real auction. In our work, to determine efficient allocations quickly in each VCG, we employ the CASS method. However, Fujishima's paper does not focus on shill bids.

Leyton-Brown et al. proposed an algorithm for computing the optimal winning bids in a multiple-unit combinatorial auction [6]. This paper describes the general problem in which each good may have multiple units and each bid specifies an unrestricted number of units desired for each good. The paper proves the correctness of our branch-and-bound algorithm based on a dynamic programming procedure. Lehmann et al. proposed a greedy optimization method for computing solutions of combinatorial auctions [5]. However, the VCG payment scheme does not provide for a truth-revealing mechanism. Therefore, they introduced another scheme that guarantees truthfulness for a restricted class of auction participants.

7. CONCLUSIONS

This paper proposed a method for detecting shill bids in combinatorial auctions. Our mechanism can judge whether there might be a shill bid from the results of the VCG procedure. We proved a theorem stating that shill bidders cannot increase their utilities unless all shill bidders win in the auction. Based on this theorem, our proposed mechanism compares the agents' utilities in a conventional auction with those in an auction where a shill bidder does not join in the auction. This process prevents shill bidders from increasing unfair profits.

Our mechanism has the three advantages. (1) Our mechanism is more superior than existing complex mechanisms in the point of view of generalization, because our mechanism employs only VCG. (2) Even though shill bidders exist in an auction, incentive compatibility property is preserved using our mechanism. (3) The schemer does never have incentive to make shill bidders in our mechanism. Namely, not to make shill bidders is dominant strategy for the schemer.

8. REFERENCES

[1] L. M. Ausubel and P. Milgrom. Ascending auctions with package bidding. *Frontiers of Theoretical Economics*, 1(1), 2002.

[2] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[3] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proc. of the 16th International Joint Conference on Artificial Intelligence (IJCAI99)*, pages 548–553, 1999.

[4] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[5] D. Lehmann, L. I. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.

[6] K. Leyton-Brown, M. Tennenholtz, and Y. Shoham. An algorithm for multi-unit combinatorial auctions. In *Proc. of 17th National Conference on Artificial Intelligence (AAAI2000)*, 2000.

[7] P. Milgrom. *Putting Auction Theory to Work*. Cambridge University Press, 2004.

[8] T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Proc. of the 16th International Joint Conference on Artificial Intelligence(IJCAI'99)*, pages 542–547, 1999.

[9] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *Proc. of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS02)*, pages 69–76, 2002.

[10] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, XVI:8–37, 1961.

[11] M. Yokoo, Y. Sakurai, and S. Matsubara. Bundle design in robust combinatorial auction protocol against false-name bids. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-2001)*, pages 1095–1101, 2001.

[12] M. Yokoo, Y. Sakurai, and S. Matsubara. Robust multi-unit auction protocol against false-name bids. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-2001)*, pages 1089–1094, 2001.

[13] M. Yokoo, Y. Sakurai, and S. Matsubara. The effect of false-name bids in combinatorial auctions: New fraud in Internet auctions. *Games and Economic Behavior*, 46(1):174–188, 2004.

Mertacor: A Successful Autonomous Trading Agent

Panos Toulis*†, Dionisis Kehagias†‡, Pericles A. Mitkas†‡,
†Aristotle University of Thessaloniki, Dept. of Electrical and Computer Engineering,
54124, Thessaloniki, Greece, +30-2310996390
‡Centre for Research and Technology Hellas, Informatics and Telematics Institute,
57001, Thermi, Greece, +30-2310464160
mertacor@olympus.ee.auth.gr

ABSTRACT
In this paper we present the internal architecture and bidding mechanisms designed for Mertacor, a successful trading agent, which ended up first in the Classic Trading Agent Competition (TAC) of 2005. TAC provides a realistic benchmarking environment in which different trading agents compete with each other in order to best satisfy their clients' preferences and maximize their total profit. The "travel game" scenario of TAC involves three types of auctions; a) continuous one-sided that sell flight tickets, b) ascending multi-unit auctions for booking hotel rooms, and c) continuous double auctions for entertainment tickets. For each one of these types, we describe the techniques deployed by Mertacor. In flight auctions prices are updated according to a random walk process, thus the accurate prediction of the next update is not feasible. A key element of agent behavior in these auctions is its ability to accurately deduce the specific time in an auction, at which bidding will be profitable. In order to deal with the uncertainty due to price fluctuations in flight auctions and to provide our agent with an efficient decision mechanism, we have designed the *z-heuristic* framework. The goal of *z*-heuristic is to figure out when the price assumes its minimum value and recommend bidding at that moment. In the case of hotel auctions, Mertacor used fuzzy reasoning in conjunction with rule-based reasoning in order to predict the closing prices of hotel rooms, when historical data from past auctions are available. In order to bid efficiently in entertainment auctions, we have designed a bidding strategy whose goal is to preserve a pre-specified long-term profit. We finally present and discuss the results of agent benchmarking in the TAC Classic game.

Categories and Subject Descriptors
I.2.11 [**Computing Methodologies**]: Artificial Intelligence-Intelligent Agents

General Terms
Algorithms, Performance, Design, Economics, Experimentation.

Keywords
Agent-mediated E-commerce, Trading Agents, Auctions.

*This author is a student

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

1. INTRODUCTION
Autonomous agents that participate in electronic marketplaces represent a new paradigm of automated transactions, in which computer programs are faced with the challenge to effectively increase their revenue without human intervention. The successful performance of agents in trading scenarios, measured by the profit or savings they materialize, may introduces a great potential for agent-mediated e-commerce in general and electronic auctions in particular.

This paper presents Mertacor, a trading agent that ended up first in the finals of international Trading Agent Competition for 2005 (TAC-05). The latter provides a competitive environment where a number of autonomous trading agents compete to each other in order to procure travel packages on behalf of a number of customers in online simultaneous and interrelated auctions of different types. In this paper we describe Mertacor's architecture and internal structure, as well as the details of the bidding algorithms designed to deal with the different types of auctions in the context of TAC.

The TAC (http://www.sics.se/tac) provides a competitive trading environment in which, each participating agent operates with the goal of assembling travel packages on behalf of eight clients. Each package refers to a 5-day period travel and consists of a round-trip flight, a hotel reservation and tickets for three different entertainment events. The clients have separate preferences over the arrival and departure dates, the type of hotel and entertainment events they wish to visit, which are randomly assigned to each client at the beginning of the game. The objective of each agent is to maximize the total satisfaction of its clients. In the TAC simulation environment, all three kinds of commodities (flights, hotels and entertainment tickets) are sold in simultaneous online interrelated auctions of three different types, running over a game, which lasts for 9 minutes. TAC has been designed so that no optimal strategy that always wins may exist. On the contrary, the agents that participate in the competition need to deploy tradeoffs between the offered goods in order to best satisfy their clients' preferences. The TAC supported auctions are described in the following.

There is only one Airline Company that sells tickets in single seller continuous one-sided auctions, which close at the end of the game. Each auction sells tickets for a particular day and direction, whereas an unlimited number of seats are available. The first agent that submits a bid in an auction buys the auctioned ticket for its current price and the auction clears. Once a bid is submitted it cannot be withdrawn. Prices in flight auctions are updated according to a random walk process. The initial price is a uniformly distributed variable in the range [250,400]. Each price

update is then perturbed every 10 seconds by a value uniformly drawn from the following intervals:

- $[-10, x(t)]$ if $x(t) > 0$
- $[x(t), 10]$ if $x(t) < 0$
- $[-10,10]$ if $x(t) = 0$.

The bound $x(t)$ is calculated by the following equation:

$$x(t)=10+(t/9{:}00)(x\text{-}10)\,, \quad (1)$$

where x is a hidden random variable drawn uniformly on [-10, 30] and 9:00 denotes the auction duration (9 minutes).

There are two hotels in which clients can stay between the arrival and departure dates: "Tampa Towers" (TT) and "Shoreline Shanties" (SS). TT is more comfortable, clean and preferable than SS, thus it is expected to be more expensive. Hotel rooms are traded in standard ascending multi-unit 16th price English auctions, which close at randomly determined times in the last 8 minutes of the game. In each auction 16 rooms are offered for each combination of hotel and night. A hotel auction clears and matches bids only once, when it closes. On clearing, hotel rooms are allocated to the 16 highest offers with all competitors paying the price of the lowest winning bid. Price quotes are only generated once per minute, on the minute.

Entertainment tickets are traded in continuous double auctions (CDA), which are held between the participants during the game. Each agent holds a randomly chosen number of tickets from the beginning of the game and may operate as either a seller or a buyer. Trade may occur at any time during the game and participants are allowed to continuously update their prices, i.e. buyers may update their offered prices and sellers their ask prices. Entertainment ticket auctions clear continuously. On clearing, bids match immediately. A bid that does not completely match remains standing in the auction.

The score that the agent receives at the end of each game is calculated as the utility minus the expenditure costs. The utility function in its general form is:

$$Utility = 1000 - travelPenalty + hotelBonus + FunBonus,$$

Apart from tackling with the utility optimization problem, TAC participants need to also deal with many uncertainty factors introduced by the different nature of each auction types and the interrelations that hold between them. For example, the agents need to acquire flight tickets, hotel rooms and entertainment tickets so that are all consistent with the preferred arrival and departure dates on behalf of their clients. Agents should also take compound decisions, including how much and when to bid. In order for an agent to operate efficiently it has to monitor its environment and collect sufficient information about the market conditions and the behavior of its opponents.

Since the beginning of the competition in 2000, the TAC problem attracted many participants from different countries and organizations. ATTac-2000 agent [7], [8] made the first key contribution to this challenging area. The intricacies of the game were clarified and attacked in a systematic way. ATTac-2000 was the first agent who won the TAC. The notion of *marginal utility* was then recognized to play an important role in the TAC game framework. In fact it has been proven that bidding marginal values in sequential auctions with deterministic prices is an optimal strategy and a fairly satisfactory one in the TAC environment [2]..

Since the first TAC, teams concentrated their efforts mainly on developing with effective ways of price prediction. A survey of agent prediction methodologies deployed in TAC-02 may be found in [10]. Due to space limitations we only refer here to those approaches that most influenced our work. A novel price prediction method was designed for agent SouthamptonTAC [3] by the application of fuzzy techniques. Walverine [1] on the other hand provided an analytical approach relied on the principles of a competitive economy. Another approach was proposed by whitebear [9], which simply but also interestingly enough, used average prices for prediction. LearnAgents [6] introduced an alternative multiagent architecture as opposed to most competitors' single agent approaches.

Although all of the above efforts generated good results, they had not traced thoroughly the details of the flight ticket price perturbation process. Having realized that the hidden variable x is the most significant parameter in flight auctions, Mertacor's flight bidding component provides a formula that calculates the probability of the fact that x lies on a particular continuous interval. This approach is innovative in the context of TAC flight auctions.

In this paper we describe the implementation details of Mertacor and illustrate the key contributions of our work. In particular, we first present the internal architecture of our agent, in Section 2, which proved to be very stable since the beginning of the competition. An additional feature of the described architecture is that it easily allows rapid changes in agent's internal bidding mechanisms due to its high degree of modularity.

The next key contribution of this paper is the description of a novel decision framework, called *z-heuristic*, designed in order to deal with the problem of optimal bidding in flight auctions.

Section 4 describes the mechanism we have deployed in order to deal with price prediction in hotel auctions. In particular, we adopted a previously known fuzzy approach, originally developed for agent SouthamptonTAC [3], which was a competent in TAC-01 and TAC-02. Further to this approach, we have also slightly modified its functionality, by adding a new set of rules that improve the results of the fuzzy reasoning module.

Another contribution of this paper, described in Section 5, is the introduction of a new bidding strategy for continuous double auctions, which are realized in the form of entertainment auctions in the context of TAC.

Section 6 provides an outline of the benchmarking results taken from TAC-05, as well as independent experiments, while Section 7 concludes the paper and outlines future work.

2. MERTACOR ARCHITECTURE

Mertacor's architecture, which is shown in Figure 1, adopts a typical modular design that supports synchronous intercommunication. Three main units (Flight, Hotel and Entertainment Auction Unit) undertake the responsibility to concurrently handle information related to the three different types of auctions running on the TAC server. Typically, this information includes the prices of the goods that are auctioned off. This configuration adheres to the typical abstract agent architecture, where the three units play the role of Mertacor's sensor components. An additional sensor unit, which also receives game information pertaining user preferences and running auction ids is the *Game Info* component.

All input information perceived by the sensor components of Mertacor goes to the *Bidder* module, which deploys the three bidding mechanisms developed for each of the aforementioned auction types into three separate modules, one for each type of

auction. Once *Bidder* calculates the new bids it sends them back to the TAC server. Thus, *Bidder* implements the main "effector" components of Mertacor. Since there is one direct communication channel to the TAC server, which is served through a TCP connection, it is critical to allow all three parts of Bidder to synchronously send information to the TAC server. For this reason, the three components of *Bidder* are implemented as threads. For instance, the *Hotel Bidder* component instantiates a thread process that places bids in hotel auctions every minute. *Bidder* also provides central control services that are activated when special conditions of emergency arise (e.g. a bid is rejected by the server).

Mertacor relies on the execution of a Linear Programming (LP) model. This is implemented as a separate module used not only to provide the optimal goods allocation but also the marginal value of the traded goods and the shadow prices of the model's constraints [5]. In what follows we provide thorough descriptions of the internal bidding mechanisms of Mertacor with respect to the three separate modules.

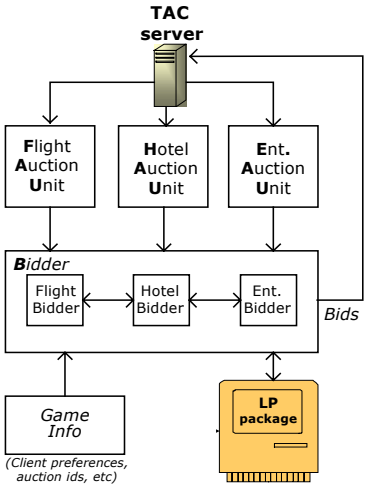


Figure 1. Overview of Mertacor's architecture

3. BIDDING IN FLIGHT AUCTIONS

According to TAC flight auction rules, price perturbations at time t are related to the value of the random variable $x(t)$, which is given by Equation (1). The ranges, from which perturbations are drawn, are determined by the sign of the bound $x(t)$. The main objective of our analysis is to figure out the time at which $x(t)$ changes sign. This time guarantees a profitable buying. This holds because $x(t)$ is a linear function depending on time and on the hidden variable x. Excluding extreme cases, this means that $x(t)$ either decreases or increases, according to x. Moreover, prices are updated either from D_1 =[-10, $x(t)$] or D_2=[$x(t)$, 10]. Since prices are initially updated from D_1, if $x(t)$ increases in a timely fashion, it is easy to show that prices will most likely increase. On the other hand, if $x(t)$ decreases, it is likely for it to change sign. In this case, the interval D_1 decreases, thus prices decrease accordingly. When $x(t)$ changes sign, prices are updated from D_2. Since $x(t)$ changes linearly, at this point it is negative and close to zero. Hence, prices will start rising again. Thus, bidding at this point will potentially result in a profitable transaction.

The goal of the decision framework we describe in the remaining Section, is to make an estimation about the real value of the x parameter and therefore about the change of the variable $x(t)$. For this purpose, we exploit the price updates that are revealed to agents every 10 seconds. These updates provide us with *low bounds,* i.e. with real numbers denoted as B_i that satisfy the inequality $x \geq B_i$.

Knowing the point where $x(t)$ changes sign is very useful. By clustering these bounds into a vector and calculating the probability of appearance of this specific vector we can construct an interval where the hidden parameter lies, with any pre-specified probability

3.1 The *z-heuristic* Decision Framework

In this section we present the z-heuristic, a framework used for estimating the time at which the prices of flight tickets reach their minimum value. For simplicity we introduce the y variable as follows:

$$y = x - 10 ,$$

where x is the hidden variable appeared in Equation (1). Since x is a random variable uniformly drawn on [-10, 30], it is clear that y is also a random variable uniformly distributed in [-20, 20] Each flight auction lasts for 9 minutes and posts a price update every 10 seconds. Therefore, the total number of price updates during a flight auction is $N=54$. Thus, Equation (1) can be rewritten as:

$$x_i = 10 + i \cdot \frac{y}{N}, i \in [1, N] \quad (2)$$

Firstly we will define the point at which $x(t)$ changes sign, as a *corner*. For this purpose we introduce the following definitions.

Definition 1. Consider two updates i_1 and i_2. We define that update i_1 is *earlier* than i_2 (or equivalently, i_2 is *later* than i_1), iff: $i_1 < i_2$.

Definition 2. We define as *corner* i_c of a flight auction the latest update for which we know that price perturbations are drawn from [-10, x_i].

Definition 3. We define as *low bound* B the maximum real number for which we know that $y \geq B$.

Our goal is to calculate the corner. Lemma 1 provides a way to find a candidate low bound that may lead to a corner.

LEMMA 1. *If δp_i is the price variation at the ith update for which is $i \leq i_c$, $B_i=(\delta p_i-10)\cdot\frac{N}{i}$* it holds that

$$y \geq B_i \quad (3)$$

Due to space limitations we skip the proofs of this and the following Lemmas. If $B_i > B$ then we have found a new low bound of the flight auction. This also gives us useful information about the corner of the flight auction, according to the following Lemma.

LEMMA 2. *If a flight auction has low bound B, then the corner i_c is given by* $i_c=\left\lfloor 10\frac{N}{|B|} \right\rfloor$, if $B \leq -10$, or $i_c=N$, otherwise.

It is important to clarify the distinction between the corner that is derived from auction data and the *real corner* of the flight auction, i.e. the update at which the bound x_i really changes sign. Our goal is to calculate the real corner. Since this variable changes linearly, the values of perturbations are drawn from [x_i, 10], after x_i changes from zero to a negative value. In this case,

prices will obviously rise because x_i is much less than 10. Thus, the price changes are positive and generally high. This explains why there is a steeply decreasing stage for flight auctions whose y parameter is below –10. After that stage an increasing one follows. The prices are reduced until the real corner of the flight auction is "crossed" and then they increase. Therefore, it is a good strategy to buy just when an update is estimated to be close to the real corner.

It is likely that the corner calculated by Lemma 2 is different than the real one. For example, let us consider a flight auction that has a real corner i_{rc} = 36. Setting $x_{i_{rc}}=0$ in (2), we get y=-15. In order for a new low bound to occur, price changes should be noticeably big, according to inequality (3). If this is not the case, the only information we are able to know about y is that $y \geq -20$. Therefore, B = -20 and the flight auction's corner is 27, according to Lemma 2. Buying at the 27th update implies that we will miss 9 updates at which price is very likely to decrease. This is of course undesirable. Thus, instead of buying at the auction corner, we define a new buying criterion based on the following Lemma:

LEMMA 3. *If* $B_i=(\delta p_i-10)\cdot\frac{N}{i}$, *B is auction's low bound and* i_c *is the flight auction corner, the following statements cannot be true at the same time:*

- $B_i < B$
- $i > i_c$
- *the real corner has been crossed*

for the ith *price update.*

Therefore, if a flight auction has bound B, corner i_c and for the ith update the following two statements are true:

- $i > i_c$
- $B_i \geq B$,

our agent assumes that the real corner has been crossed, and decides to buy the tickets needed.

Normally, in a flight auction many successively increasing bounds B_i may come up at the ith update. We define as *evidence couples* any couple of the form (B_i,i), and use them to estimate the value of variable y. By recording all evidence couples found during an auction we can provide the following definition.

Definition 4. *Bound evidence* (*BE*) is a vector of the form $BE=\left[(b_1,k_1)\ \ (b_2,k_2)\ \ \cdots\ \ (b_n,k_n)\ \ T\right]$, where $b_i=\frac{B_i}{20}$ and $i \in 1,2,\ldots n$.

Let us also define variable $s = y / 20$. The variable T is an integer equal to the last update of the flight auction.

It is very useful to calculate the probability $p(BE|s)$ of the fact that a specific bound evidence occurs in an auction, when the value of the variable s is known. This is provided by the following Lemma.

LEMMA 4. *If s=y/20, the probability of occurrence of the evidence couples* (b_1, k_1) *and* (b_2, k_2) *with* $b_2 > b_1$ *and* $k_2 > k_1$ *is*

$$\Upsilon(k1,\ k2,\ b1,\ s) = \left\{\prod_{i=k_1+1}^{k_2-1}\frac{(N+i\cdot b1)}{(N+i\cdot s)}\right\}\cdot\frac{k_2\cdot(s-b_1)}{(N+k_2\cdot s)} \qquad (4)$$

By generalizing Lemma 4, the probability $p(BE|s)$ can be calculated, using (4), by the following equation:

$$p(BE \mid s) = \prod_{i=1}^{n-1}\Upsilon(k_i,k_{i+1},b_{i-1},s)\cdot\prod_{j=kn+1}^{T}\frac{(N+j\cdot b_n)}{(N+j\cdot s)} \qquad (5)$$

Equation (5) describes $p(BE|s)$ as a function $g(BE, s)$ of the hidden variable s.

Since $y \in [-20,20]$, it is $s \in [-1,1]$. From probability $p(BE \mid s)$, we can calculate p(s| BE) which will give us an estimation of s and thus y. Using Bayesian probabilities we can write

$$p(s \mid BE) = \frac{p(BE|s)\cdot p(s)}{p(BE)} \qquad (6)$$

Setting s = s_1 and s_2 successively in Equation (6) we create two new equations from which we derive that $p(s_1|\ BE)p(BE|\ s_2) = p(s_2|\ BE)p(BE|\ s_1)$, or:

$$p(s_1|BE)g(BE,s_2)=p(s_2|BE)g(BE,s_1) \qquad (7)$$

Now let us consider a continuous subset $S_1 \subseteq [-1,\ 1]$. Keeping s_2 fixed and having $s_1 \in S_1$ by integrating equation (7) we have:

$$p(s\in S_1|BE)g(BE,s_2)=p(s_2|BE)\int_{S_1}g(BE,s) \qquad (8)$$

Assume that $S_2 = [b_n, 1]$. Since B_n is actually the low bound of the flight auction and b_n is the normalized low bound, it is:

$$p(s \in S_2 \mid BE) = 1 \qquad (9)$$

From Equations (8) and (9) for $s_2 \in S_2$ we have:

$$p(s\in S_1|BE)=\frac{\int_{S_1}g(BE,s)}{\int_{S_2}g(BE,s)}$$

If we consider $S_1 = [b_n, z]$ we can gradually increment z so that $p(s \in S_1 \mid BE)$ becomes equal to a fixed predefined value p. We chose this value to be 0.8. With this algorithm we can find a range $[b_n, z]$ in which the parameter s lies, with probability 0.8. This value of the z is called *z-point***.** The reason for applying such a methodology was motivated by an experimental analysis whose results are graphically described in Figure 2.

In Figure 2 we can see the relationship between the bound of the y parameter found until the 20th update for 50,000 auctions and the real value of y. Obviously, real y values tend to gather around and below the line $y=B$. This is the reason we use the range $[b_n, z]$ in our z-heuristic framework**.**

Having constructed a range in which we assume with sufficient evidence that the hidden parameter s lies, we can make an estimation of its real value. In order to accomplish this, we defined the *sign evidence (SE) vector* as a vector of the signs of all auction price updates. Following a procedure similar to the one used for the calculation of probabilities of occurrence of the *BE* vector we can calculate $p(SE)$ and $p(SE|s)$.

The estimation of s is enabled through the following equation**:**

$$\hat{s}=\frac{1}{z-b_n}\cdot\int_{S_1}s\cdot\frac{p(SE|s)}{p(SE)}ds \qquad (10)$$

Thus, the estimation of y is simply derived by the equation:

$$\hat{y}=20\cdot\hat{s} \qquad (11)$$

Equations (10) and (11) provide an approximation of y that can be used in conjunction with statistical methods, in order to enable an estimation of the price change from the arbitrary update i_a to the

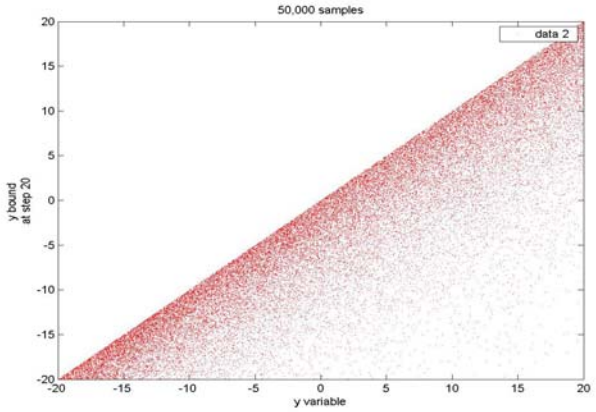


Figure 2. Low bounds found until the 20th update for 50,000 games. Samples are gathered around line $y=B$ which provides a good estimation of the real y value.

later update i_b, where $i_b < i_a$. By considering the mean value in the range [-10, x_i], we receive:

$$\widehat{\Delta p}(i_a, i_b) = \hat{y} \cdot \frac{(i_a+i_b)\cdot(i_b-i_a+1)}{4\cdot N}$$

Finally, we determine which values of the above estimations play a significant role in the bidding process. For this reason, we define the following decision criteria, considering a flight auction with current price *CP*, initial price *IP*, current update *i* and low bound *B*:

1. Is $T>52$?
2. Is $CP<160$\$?
3. Is *Corner* crossed?
4. Is $\hat{y}>4$ AND $B>-3$ AND $CP>IP$?
5. Is $B>7$?

3.2 Flight Auctions Analysis

Each of the previously defined criteria is potentially met according to the current status of the auction parameters. A *decision variable*, which is an integer $d = 0,\ldots,5$ represents one of the previously defined decision criteria. A positive value of d triggers the bidding mechanism, 0 means that no criterion is satisfied. For example, if $d = 5$, criterion 5 is met, hence prices will most probably skyrocket. By using these criteria, our agent may opt to postpone a bid submission or fragment the total quantity for which it intends to bid.

The results of auction analysis are reported in Table 1. This table shows the evolution of a real flight auction during the first 9 updates.

The flight auction starts off with an initial value of \$327. At update no.3 an increment of \$9 gives a new low bound $B = -18$ and the corner becomes 30. At the 5th update a price increment by 10 gives a new bound $B = 0$ and now the corner becomes 54, which implies that until the end of the auction, price fluctuations will be drawn on $[-10,x_i]$. Since bound is zero, it will be $y \geq 0$ and y is estimated to the value 8.99. At this time criterion 3 is fulfilled, therefore the decision variable changes to value 3. This indicates that current time is a good time to buy the auctioned ticket. The final price was ended up to \$388, thus the recommended decision on bidding results to a \$35 profit for our agent.

Table 1. Flight Auctions Analysis table (9 updates)

i	i_c	Initial price	Current price	Last bound	Z-point	Decision	Y est.	Buying Price
1	27	327	327	-20.0	11.88	0	-4	327
2	27	327	332	-20.0	11.60	0	-4.28	332
3	30	327	341	-18.0	15.49	0	-0.73	341
4	30	327	343	-18.0	15.29	0	-0.49	343
5	54	327	353	0.0	17.34	4	8.99	353
6	54	327	352	0.0	17.34	4	8.86	352
7	54	327	345	0.0	17.19	4	8.63	345
8	54	327	350	0.0	17.19	4	8.78	350
9	54	327	361	6.0	18.25	4	12.29	361

4. HOTEL BIDDING STRATEGY

At the beginning of the game, there is little or no information about hotel prices. For this reason we use the following procedure in order to calculate the initial bids to submit. Denoting (*arrival day*, *departure day*) a user preference over the arrival and departure dates, there are 10 possible such travel preferences: (1,2), (1,3), (1,4), (1,5), (2,3), (2, 4), (2, 5), (3, 4), (3, 5) and (4, 5). Four out of ten preferences include an in-flight at day 1, three at day 2, two at day 3 and one at day 4. Thus, we can define a parameter that indicates how likely for an in-flight day is to occur among client preferences. Let us call this parameter *statistical importance factor* of in-flights for the day *i*:

$$s_{in,i}=0.5-0.1i, i\in[1,4]\ ,$$

Similarly, the statistical importance factor of out-flights for the day *j* is:

$$s_{out,i} = 0.1j - 0.1,\ j \in [2,5]$$

Our goal is to estimate hotel closing prices. For this purpose, we use the only information, which is available in the beginning of each game, i.e. the initial flight prices. Our methodology uses these prices to estimate the average hotel room demand, which indicates how far from the statistical balance the game will evolve. The probability that a client will be staying for *i* days is equal to: $p(i) = 0.5 - 0.1 \cdot i$. Thus, the *mean trip duration* equals to $\sum_{i=1}^{4} p(i)\cdot i = \sum_{i=1}^{4} (0.5-0.1\cdot i)\cdot i = 2$ days and hence the average demand for hotel rooms per game is:

$$2\frac{rooms}{client} * 8\frac{clients}{agent} * 8\frac{agents}{game} = 128\frac{rooms}{game}$$

Since 16 rooms are sold in each of the 8 hotel auctions, the *hotel room supply* is equal to the average demand. It is reasonable to assume that in games with demand close to the average demand, a balanced bidding activity will be noticed. Otherwise, the game may become unstable and the hotel prices skyrocket.

Let us assume that the eight competitors need I_i in-flight tickets at day *i* and O_j out-flight tickets at day *j*. Total in-flight and out-flight demand will be $I = \sum Ii = 64$ and $O = \sum Oi = 64$, respectively, since we have 64 clients and one in-flight and out-

flight preference for each client. Let us also define the *relative* in-flight and out-flight ticket demand as $\varepsilon_i = Ii / I$ and $\mu_i = O_j / O$, respectively, with $i \in [1,4]$, $j \in [2,5]$. If the absolute hotel room demand for day i is H_i, then it is easy to prove that the following equations hold: $H_1 = I_1$, $H_2 = I_1 + I_2 - O_2$, $H_3 = -I_4 + O_4 + O_5$ and $H_4 = O_5$, or equivalently:

$$\mathbf{H}=\mathbf{R}\cdot\mathbf{F}, \tag{12}$$

where $\mathbf{H} = (H_1 \quad H_2 \quad H_3 \quad H_4)^{\mathrm{T}}$,

$$\mathbf{R} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

and $\mathbf{F} = \begin{pmatrix} I_1 & I_2 & I_3 & I_4 & O_2 & O_3 & O_4 & O_5 \end{pmatrix}^{\mathrm{T}}$. Equation (12) maps absolute flight demand to absolute hotel demand.

The estimation of the hotel demand for each day is calculated by the following equation:

$$\hat{\mathbf{H}} = \mathbf{R} \cdot \hat{\mathbf{F}}, \tag{13}$$

where:

$$\hat{\mathbf{F}} = 64 \cdot \begin{pmatrix} \hat{\varepsilon}_1 & \hat{\varepsilon}_2 & \hat{\varepsilon}_3 & \hat{\varepsilon}_4 & \hat{\mu}_2 & \hat{\mu}_3 & \hat{\mu}_4 & \hat{\mu}_5 \end{pmatrix}^{\mathrm{T}},$$

and $\hat{\varepsilon}_i$ $\hat{\mu}_j$ are the relative demand estimates for in-flight and out-flight tickets respectively. In order to calculate $\hat{\varepsilon}_i$ we use the following equations:

$$\hat{\varepsilon}_i = f_{in,i} / \sum_{k=1}^{4} f_{in,k} \text{ and } f_{in,i} = s_{in,i} \cdot (400 - IP_i) / 150,$$

where IP_i is the initial in-flight ticket prices for day i. The $\hat{\mu}_j$ coefficients are calculated in a similar manner.

After estimating hotel room demand we create initial bids by using look-up tables with historical data from previous games. These tables provide mappings between closing prices and hotel room demand estimates. As real information about market demand is being revealed in the game, our agent exploits the fuzzy reasoning method [4] developed for SouthamptonTAC [3] for price prediction. In our implementation, we used the prediction p combined with the mean value m of the closing prices denoted at the previous 14 games. We use a Linear Programming module in order to calculate the optimal allocation of the hotel rooms, when the closing price is estimated to be a linear conjunction of the previous prediction p and the mean value m. In particular, the closing price estimation, denoted I_{LP}, is calculated in a heuristic manner by the following rules:

IF $p \geq 2m$ **THEN** $I_{LP} = p$;
ELSE IF $p \leq m/2$ **OR** $p \geq (3/2)m$ **THEN** $I_{LP} = (2/3)p+(1/3)m$;
ELSE $I_{LP} = (1/2)(p+m)$;
END IF

During the last two minutes of the game, when uncertainty about the market is reduced, using the above estimation of hotel closing prices increases bidding efficiency

5. ENTERTAINMENT BIDDING STRATEGY

Entertainment auctions are continuous double sided auctions, where agents receive new price quotes every 30 seconds and bids are processed continuously. Although the entertainment auctions adhere to the continuous double auction protocol applied in the stock market, they formulate a more simplified auction environment for two reasons. First, the number of eight participants is significantly lower than the ones met in a typical stock market. Second, the agents remain adherent to their initial plans about acquiring the desired tickets. This is not always the case in a real stock market, where traders may deploy totally unpredicted bidding behaviors.

Mertacor's bidding mechanism in entertainment auctions is based on a simple and consistent algorithm that aims to achieve a long-term profit at the end of the auction.

Our selling strategy in the entertainment auctions is implemented by the procedure *MertacorSellStrategy* illustrated in Figure 3. We describe its functionality in what follows.

```
PROCEDURE MertacorSellStrategy
 1:  FOR each entertainment ticket in possession
 2:      Assign a pre-specified value to target;
 3:      mean ← getMeanValue();
 4:      M ← A*target + B*mean;
 5:      IF M ≤ (1/2)*target OR M ≥ (3/2)*target THEN
 6:        M ← relocateM();
 7:      END IF
 8:      V ← calcVal();
 9:      IF V < Vo THEN V ← Vo; END IF
10       profit ← ask– 2*V;
11       Mt ← w(t)*M;
12       Ra ← M;
13       Rb ← 3*M/2;
14       Rc ← rand(M/2, M);
15       IF profit ≥ Mt-Ra THEN sellTicket();
16       ELSE IF profit ≥ Mt-Rb THEN ask (Mt-Ra+2*V);
17       ELSE ask (Mt - Rc +2*V);
18       END IF
19  END FOR
```

Figure 3. The selling strategy deployed by Metacor in the entertainment auctions.

The procedure iterates over all tickets that the agents possesses. The *target* variable represents the long-term average profit that the agent aims at achieving. This is set to a pre-specified value for each of the entertainment tickets. Typical values for *target* lie on [5, 12]. The variable *mean* is set equal to the current mean value of profit gained over the previously completed transactions. This is calculated by the function *getMeanValue()* in row 3. Variable M, which is given by the following equation:

$$M = A \cdot target + B \cdot mean \tag{14}$$

is used to determine the range of the profit sought. After experiments, the weights A and B were chosen to be A=0.7 and B=0.3. Thus, our agent seeks for a profit highly influenced by the value of *target*. In order to keep our agent adherent to this goal,

we impose (row 5) the variable M to only take values in the range [(1/2)·*target*, (3/2)·*target*]. For this reason, we use function *relocateM*() (row 6), which returns values in the range [min {*target*, *mean*}, max {*target*, *mean*}]. In order to calculate current ticket's valuation V, our agent calls function *calcVal()*, in row 8, which makes use of the LP model. Since we are interested in always selling higher than a determined reserve price V_o, if $V<V_o$ we set $V=V_o$ in row 9. If we denote *ask* as the ask price, then Mertacor calculates the profit it would make in a potential transaction by using the equation

$$profit = ask - 2V \quad (15)$$

This preserves our agent's goal, which is to keep $E\{profit\}=target$, where the $E\{.\}$ operator denotes the mean value. In row 11 we define a new variable $M_t = w(t)M$ that introduces a time-depended low bound for the desired profit. The $w(t)$ function is graphically represented in Figure 4. The peak that appears in the middle of the game (4.5 minutes) represents the maximum profit sought at that time. The demand for profit decreases and then remains constant at a relatively low value as the game reaches its end. Next, we define three variables, namely R_a, R_b and R_c, in rows 12, 13 and 14 respectively. Function *rand()* in row 14 returns a number uniformly distributed in the range [M/2, M]. These three variables determine, along with the variable M_t the ranges of an acceptable profit. In order to decide when to bid, our agent uses the three decision rules shown in rows 15-17. According to these rules, Mertacor sells the ticket it currently possesses if the profit to be achieved is M_t-R_a or above. This is done by function *sellTicket()*, in row 15. Otherwise, it asks for those prices that will result to a profit equal to or bigger than M_t-R_a or M_t-R_c. Our agent submits its ask price (derived from (15)) for both cases, by invoking the *ask()* function in rows 16 and 17, respectively. Since $R_c \leq R_a$ the expected profit will be at least equal to ($w(t)$-1)M**.**

From Figure 4 we can see that $w(t) \geq 1$ for most of the time. During the last 30 seconds of the game $w(t)$ is fixed to value 0.8 and the profit becomes -0.2M, which is a negative value. Thus, if M has a big value, a transaction occurred during the last 30 seconds of the game, will lead to a big loss. In addition to this, transactions close to the middle of the game are not so likely to happen, since the quantity ($w(t)$-1)M is relatively big. This will lead to a decrease of the overall average profit, and thus M as it is derived from Equation (14). On the other hand, if M is small, transactions that lead to a positive profit are more likely to happen, while negative profit transactions will result to a small loss. The

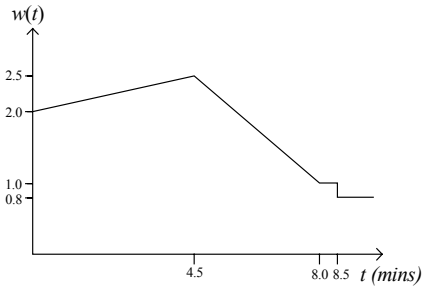


Figure 4. The *w*(*t*) function plotted over time (in minutes). It indicates a time-dependant bidding attitude

aforementioned mechanism is highly adaptive to changes in the market environment, since when a big profit is assumed the number of transactions is reduced and vice versa. The buying bidding strategy of Mertacor is completely symmetrical to the selling strategy.

6. RESULTS

In the 6th TAC, eleven agents participated from the beginning of the competitions. Among these only eight managed to end up to the finals. The agents were evaluated according to the average score they gained at each round. The score of each agent is calculated by the server when a game finishes and it is equal to the utility minus expenditure costs. For each round the average score determines the performance metric for all agents.

TAC-05 consisted of 5 rounds. In the first (qualifying) round of TAC all eleven agents participated in 600 games, running for almost two weeks in a constant 24-hour basis. Our agent ranked fourth, gaining a score of 3918.45, which was 240.33 below the top score achieved by agent whitebear05. In this round Mertacor employed a completed version of the Flight Auctions strategy, while it used a naïve bidding algorithm for hotel auctions and a greedy strategy for entertainment auctions. Next, in the seeding round 688 games were played using the same server configuration. At the beginning of this round we changed our hotel bidding mechanism to the one presented in Section 4 and preserve it until the finals. We also used for the fist time the CDA strategy described in Section 5. We set the parameter *target* in Eq. (14) to value *target* = 12, i.e. our agent was seeking for a big profit. In this round Mertacor improved its overall performance. It finished third with score 4033.32, managing to reduce its distance from the top score agent whitebear05 to 135.29. In the semi-final round ten participants were able to compete in 56 games. Mertacor improved many aspects of all bidding algorithms and mostly fixed many bugs. Thus, it retained its performance by gaining a score of 4023.88.

Eight out of the ten competitors who participated in the semi-finals were invited to the finals held on August 3rd in Edinburgh, Scotland. Table 2 shows the scores of the eight finalists. In the final round all agents played in 80 games, concurrently run in two servers (40 games in each server). Compared to the semi-final round, Mertacor improved its entertainment bidding strategy, by fine-tuning the parameter in the entertainment stategy. In particular, we lowered the value of the *target* parameter to 5. This resulted in both lower profits and losses. This intervention proved to significantly improve our agent's performance who eventually ranked first.

Apart from the overall benchmarking results provided by TAC, we have also conducted an additional analysis of the agents' performance regarding only the entertainment auctions, because fine tuning the *target* parameter in the entertainment bidding algorithm proved to be the key element that increased our agent's performance. In the context of our experiments, we measured the profits that the agents made in their transactions, in all the games of the final round. The results of this analysis are shown in Fig. 5. For each agent we measured the average profit collected in buying, selling and both types of transactions. Fig. 5 shows that the proposed algorithm is particularly efficient in buying transactions. Mertacor performed almost as well as its competitors in selling transactions. In the latter case the best performing agent

is Dolphin. Noticeably, Mertacor and whitebear05 are the only agents that managed to receive a positive profit when participating in buying transactions, with Mertacor outperforming all agents. This resulted in an overall better performance of Mertacor in the entertainment auctions.

Table 2. Score table for TAC-05 final round

Agent	Average Score	Zero Games
Mertacor	4126.49	0
Whitebear05	4105.68	0
Walverine	4058.90	2
Dolphin	4022.79	0
SICS02	3972.29	0
LearnAgents	3899.24	0
e-Agent	3451.25	0
RoxyBot	3167.64	10

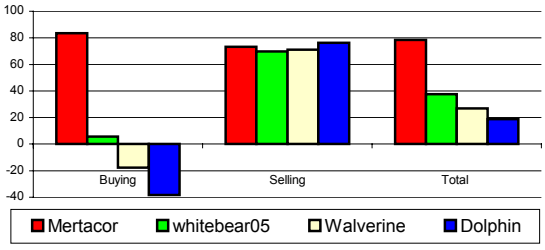


Figure 5. Average Profit collected by the four top-scoring agents in the final round when they participated in buying, selling and in both types of transactions

7. CONCLUSIONS

This paper presented the details of Mertacor, a trading agent which took part in TAC-05 competing against eleven agents in a simultaneous interrelated auctions environment. We analyzed our agent's architecture and provided the details of the internal bidding mechanisms. Specifically, we presented the bidding strategies deployed by our agent per each of the three different types of auctions involved in the TAC game. In flight auctions we tackled with the decision problem of choosing the best time to bid in the auction, i.e. the time that leads to a maximum profit. In order to calculate the time at which bidding is more profitable, we have developed a methodological framework called z-heuristic based on probabilistic reasoning. The outcome of z-heuristic is then used by our agent in conjunction with a set of simple bidding rules in order to decide when to bid. Next, we explained the price prediction method adopted for efficient bidding in the hotel auctions. This method comes from previous related work, and it was slightly enhanced in order to meet our agent's design requirements. Although we developed a new technique for price prediction in hotel auctions, we applied it on Mertacor only at the qualifying round, while we then switched to the adopted methodology, which we kept until in the finals rounds for the sake of performance stability. In order to deal with bidding in double-sided entertainment auctions we developed a strategy with the goal to achieve a long- term profit specified by a target value. This strategy proved to be robust and easily adaptable to market fluctuations, especially when Mertacor participated in buying transactions. Our agent, which exploited a combination of strategies, exhibited outperformed performance in the context of the TAC.

Future work involves further improvement of the presented techniques and continuation of our experiments.

8. ACKNOWLEDGEMENTS

We would like to thank the TAC organizing teams, University of Michigan and SICS for their technical support before and during the competition.

9. REFERENCES

[1] Cheng, S.-F., Leung, E., Lochner, K. M., O'Malley, K., Reeves D. M., Schvartzman L. J., and Wellman, M. P. Walverine: A walrasian trading agent. In *Proceedings of the Second international Joint Conference on Autonomous Agents and Multiagent Systems* (Melbourne, Australia, July 14 - 18, 2003). AAMAS '03. ACM Press, New York, NY, 465-472.

[2] Greenwald, A, and Boyan, Bidding Under Uncertainty: Theory and Experiments. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*. (Jul. 2004), 209-216.

[3] He, M., and Jennings, N. R. Designing a Successful Trading Agent: A Fuzzy Set Approach. *IEEE Transactions on Fuzzy Systems, 12,* 3 (Jun. 2004), 389-410.

[4] Kandel, A., and Langholz G. *Fuzzy Control Systems*, CRC Press, 1993.

[5] Kolman, B., and Beck, R. *Elementary Linear Programmig with Applications*. Academic Press Inc. (London), 1995.

[6] Sardinha J. A. R. P., Milidiú R. L., Paranhos P. M., Cunha P. M., Lucena C. J. P. An Agent Based Architecture for Highly Competitive Electronic Markets. In *Proceedings of the 18th International FLAIRS Conference,* (Clearwater Beach, Florida, USA, May 16-18, 2005).

[7] Stone, P., Littman, M. L., Singh, S., and Kearns, M. ATTac-2000: An Adaptive Autonomous Bidding Agent. *Journal of Artificial Intelligence Research 15* (2001), 189-206.

[8] Stone, P., Schapire, R., Littman, M., Csirik, J., and McAllester, D. Decision-theoretic bidding based on learned density models in simultaneous, interacting auctions, *Journal of Artificial Intelligence Research, 19*, (2003), 209-242.

[9] Vetsikas, I. A. and Selman, B. 2003. A principled study of the design tradeoffs for autonomous trading agents. In *Proceedings of the Second international Joint Conference on Autonomous Agents and Multiagent Systems* (Melbourne, Australia, July 14 - 18, 2003). AAMAS '03. ACM Press, New York, NY, 473-480.

[10] Wellman, M. P., Reeves D. M., Lochner, K. M., and Vorobeychik, Y. Price prediction in a trading agent competition. *Journal Of Artificial Intelligence Research*, *20* (2004) 1-17.

[11] Wellman, M. P. Wurman, P. R. Oapos Malley, K. Bangera, R. Lin, S.-d. Reeves, D. Walsh, W. E. Designing the market game for a trading agent competition. *IEEE Internet Computing, 5*, 2 (2001), 43-51.

[12] Zimmermann, H. J. *Fuzzy set theory and its applications*, Kluwer, Boston, 3rd edition, 1999.

A Comparison Between Mechanisms for Sequential Compute Resource Auctions

Andrew Byde HP Labs
Filton Road, Stoke Gifford
Bristol, BS34 8QZ
andrew.byde@hp.com

General Terms

Economics, Experimentation, Management

Keywords

Bidding, Utility Computing, GRID, Market-based Control, Genetic Algorithms

Categories and Subject Descriptors

I.6.3 [**Simulation and Modelling**]: Applications—*Economics*; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems

1. INTRODUCTION

This paper describes simulations designed to test the relative efficiency of two different sequential auction mechanisms for allocating compute resources between users in a shared data-center. For each of two possible auction types we apply a genetic algorithm to a broad class of bidding strategies to determine a near-optimal bidding strategy for a specified auction type, and use statistics of the performance of these strategies to determine the most suitable auction type for this domain.

To ground the simulations, we have based our model on the SE3D project [3], in which a select group of up-and-coming film-makers was given access to a state-of-the-art CGI rendering utility owned by HP, and were forced to acquire resources via markets. We hypothesise an extension to SE3D in which animators are given bidding agents to relieve them of the chore of bidding, and seek to understand what appropriate bidding strategies for those agents might be, and what impact market mechanism choice would have.

2. RELATED WORK

The work of Ferguson et al on distributed markets for compute resources [2] is some of the earliest. Like [1] and [4], among many others, allocation in Ferguson's system is done at the level of the individual server, in order to manage distributed resources. The benefit of this approach is that by distributing the mechanism a central point of control (and hence potential failure) is eliminated. The downside of a distributed mechanism is that allocations might be far from optimal, and might have unstable dynamics, depending on the coordination mechanisms used to migrate a job between nodes on the basis of pricing. In any case, since this paper concerns a datacenter in which all compute resources are centrally hosted, there is no benefit to using a distributed mechanism.

Bidding to acquire complex sets of resources is not a novel area of study for agents: the canonical example in this field is of course the Trading Agent Competition [7]. Optimal strategy selection is notoriously dependent on the specific configuration of goods, mechanisms, preferences and so on. Our focus however is not so much on finding good strategies as it is in comparing different mechanisms in the domain of Utility Computing.

A variety of auction mechanisms have been tried as the basis for a market-based resource allocation system. In particular the Proportional Share and Generalized Vickrey mechanisms we use here have both been used before – for example PS in [4], GV in [8]. In addition other mechanisms have been used such as an ascending N^{th} price mechanism (in [5]) similar to GV but in which the price paid is the same for all winners and equal to the highest losing bid; and a first price mechanism in [2], [6]. Almost all the work cited evaluates a particular mechanism rather than comparing alternative market mechanisms.

3. RENDER UTILITY SIMULATOR

3.1 Rendering Model

To simulate the challenges that an agent might be expected to face we have to simulate the operational details of the rendering application acting on a "realistic" payload. The basic unit of rendering is a *frame*, which is defined by its intrinsic work content, measured in server milliseconds. When a frame is assigned to a server, work begins on the frame, and proceeds either until the relevant number of milliseconds has elapsed, or until rendering on the frame is terminated. To be faithful to the SE3D environment the rendering simulator obeys the following constraints:

1. Work already done on a frame cannot be check-pointed, so that if rendering on a frame is terminated before the frame is complete, all work so far done on the frame is lost, and must be done again.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. A frame is not parallelizable at all, so that for a single frame two servers would take the same amount of time as one.

The combination of these two properties makes performance non-linear with respect to resource levels, especially if frame workloads are large compared to the timescale of the auctions. The task that each agent has to work on is a *render job*, consisting of a set of frames, and start time, end time (deadline) and total budget.

3.2 Auction Models

In SE3D, resources were sold on a rolling basis both for immediate use and in advance of later use. In these simulations we restricted attention to auctions for resources available immediately. Thus time is divided into "rounds". In each round there are three steps: first bids are gathered from each agents; then an auction is run to determine resource allocations until the next round; then work is done on each job with the resulting number of resources, according to the rendering model described above in Section 3.1.

There were two auction mechanisms tested in the SE3D project, and these were faithfully reproduced in the simulation.

3.2.1 Proportional Share Auction

In a Proportional Share (PS) auction for N servers, each agent submits a bid b^a; the number of resources allocated to each agent a is as close as possible to the share

$$N^a_{lim} = N\frac{b^a}{\sum_{a'} b^{a'}}. \quad (1)$$

Since this number is not in general an integer, we cannot in general give exactly N^a_{lim} servers (which are indivisible). Therefore the actual number N^a allocated is determined in a time-dependent way such that $|N^a - N^a_{lim}| \leq 1$ and such that if all bids remain the same for multiple successive rounds in the auction, then the average allocation $\langle N^a \rangle$ will converge[1] on N^a_{lim}. Before processing, bids are truncated to lie in $[0, B^a]$, where B^a is agent a's total remaining balance.

3.2.2 Generalized Vickrey Auction

In a Generalized Vickrey (GV) auction for N servers, each agent submits a vector of bids, $(b^a(1), b^a(2), \ldots, b^a(N))$. The number $b^a(j)$ represents the maximum amount of money that agent a would be willing to pay to secure exactly j resources. The resource allocation N^a to each agent is calculated so as to maximize the sum of maximum prices $S(N) = \sum_a b^a(N^a)$. The price paid by each agent is set to be the negative impact that the presence of that agent has on this total as calculated over the other agents: with agent a removed we calculate the allocation $N^{a'}_{-a}$ for the other agents a' that maximizes $S_{-a}(N_{-a}) = \sum_{a' \neq a} b^{a'}(N^{a'}_{-a})$. The price paid by agent a is

$$S_{-a}(N_{-a}) - (S(N) - b^a(A_a)) = \sum_{a' \neq a} b^{a'}(N^{a'}_{-a}) - b^{a'}(N^{a'}). \quad (2)$$

If there are several allocations that give rise to the same maximal total $S(N)$, we cycle between those that minimize the difference between allocations to different agents. For example, if two agents bid $(10, 20, 30)$ for a total of 3 resources, any allocation of all the resources has value 30, the two allocations that minimize the difference between agents are $(1, 2)$ and $(2, 1)$ and so we cycle between these. The period of this cycle in time steps is a parameter of the allocation mechanism[1].

In SE3D, to simplify bidding in the GV auction so that bidders were not compelled to enter a complete vector of bids for each time step, we constrained them to use bid vectors from a 3-parameter family, parameterized by the maximum number of units for which they wanted to bid (N_{max}), a total price for all units ($b_{max} = b^a(N_{max})$), and a risk factor (r) in $[-1, 1]$ which determines the rate of decrease of marginal demand $b^a(n) - b^a(n-1)$.

[1]In practice, extra resources cycled around once every three time steps, to avoid too much "thrashing"

4. BIDDING STRATEGIES

4.1 Strategy Space

In its most general form, an agent's bid depends on the current state of its job, the list of past bids it has made and resulting allocations and prices, and any accumulated history of previous runs. In cases of interesting scale the space of strategies is therefore far too large to enumerate.

We chose to restrict the space of bidding strategies by making the fraction of remaining budget bid, dependent only on current time step t and time to deadline d, with rational polynomial dependency. In the PS markets for example, each bidding strategy was defined by six constants $c_1, \ldots, c_6$, via the function

$$b^a/B^a = (c^a_1/d + c^a_2 + c^a_3.d + c^a_4.d^2) \times (1 + c^a_5.t + c^a_6.t^2), \quad (3)$$

where B^a is agent a's remaining funds at the time of bidding.

Likewise in GV markets each of the three strategic parameters, total bid b_{max}, maximum number of units N_{max}, and risk factor r, were represented in this way, leading to an 18-dimensional bidding strategy space.

4.2 Strategy Optimization

It is clear that even within the restricted families described above we have no hope of finding any equilibria analytically. When confronted with this situation, different researchers have taken different approaches. We chose to apply a genetic algorithm to the sets of parameter values in order to evolve appropriate strategies.

To do this, a population of strategies was instantiated to multiple copies of the default strategy ($b/B = 1/d$ for PS markets, $b_{max}/B = 1/d$, $r = 1$, $N_{max} = N$ for GV). In each generation each individual was placed into a run of the simulation with randomly selected opponents drawn from the population; each agent was given a randomly generated job to perform, and the simulation run to determine the performance of the agents – and hence the strategies. After a number of such runs for each individual, the averaged performance data were used as a measure of the fitness of each individual. To generate the next generation, genomes were selected with preferential weighting given to high-fitness individuals, and a new set of parameters generated from the two parents by crossover. Mutation was applied to each parameter, and the resulting parameter-set inserted into the new population. Elitism was used, so that the best individual in a given generation was always transferred to the next generation.

5. RESULTS

The standard simulation environment consisted of 6 agents competing for 15 resources with job start time $\sim U[0, 10]$; time from start to deadline $\sim U[20, 40]$; number of frames $\sim U[10, 20]$; and budget per frame $\sim N(0.8, 0.2)$. Agent performance was measured by the number of frames completed by the job's deadline.

These parameters were chosen to give maximal performance difference between the mechanisms. When the system is under very little contention all agents get all their jobs done all the time; when the system is under very heavy contention, agents fail so often that their performance is not much affected by mechanism choice either. In between the mechanism makes a difference, and this was the parameter zone from which the experimental conditions were selected.

A benchmark resource allocation mechanism is "fair share" (FS) in which each active agent is simply given an equal share of the resources.

With the setup above, using FS, the mean number of frames incomplete at the end of a simulation run was 4.12 with a standard deviation of 5.54 – a significant number given that the total number of frames per job was in the range 10 – 20.

5.1 Performance data

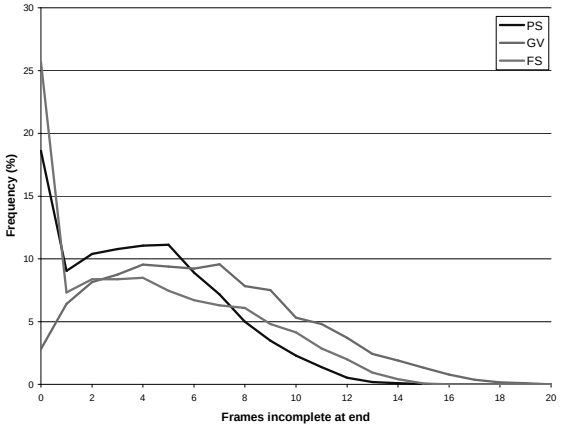


Figure 1: Distribution of number of incomplete frames.

Figure 1 shows the distribution of the number of frames that have not been rendered by the time of the jobs deadline. Although FS has the greatest frequency of runs in which all frames are completed 25.7%, it has a higher proportion of cases in which a large number of frames is incomplete than PS. This is reflected in the standard deviation of frames complete – 5.54 for FS as opposed to 3.85 for PS. In fact PS has both the lowest variance and the lowest mean: 3.85 for PS as opposed to 4.90 for FS. GV has the highest mean 6.38 and the highest standard deviation 7.43.

To try to analyse why mechanisms might give the performance metrics they do, we can examine the statistics of resource allocation. We can examine the difference between runs at a given time. By time 15 all agents' jobs have started and none have ended, so all agents are active. Since there are 6 agents and 15 resources, each agent gets either 2 or 3 with 50% probability, giving a standard deviation of allocation of 0.5. For PS the standard deviation is 0.97, while for GV it is 1.31, demonstrating that the allocations fluctuated more widely with the GV mechanism than PS, which itself varied more widely than FS.

Some of this variability is due to variability in job budget, start time etc. We can control for this by examining the change in allocation from time step to time step throughout the lifetime of each job. The standard deviations of allocation change are 1.12, 1.14 and 1.24 for PS, FS and GV respectively, showing that although the allocations generated by FS have markedly lower variance than the other mechanisms at time 15, over the lifetime of a job FS gives a similar although slightly higher level of variability to PS. Interestingly PS is far ahead when it comes to the number of time steps in which allocation did not change at all: 75%, as opposed to 66% and 67% for GV and FS respectively.

These relative standings with respect to volatility are the same when repeated for for allocation change measured relative to the average allocation given to a job over its lifetime. Here the standard deviations are 0.699 for PS and FS, and 0.881 for GV.

6. CONCLUSION

While it might seem that the choice of market mechanism through which to sell shared compute resources is arbitrary on the grounds that supply and demand should lead to the same equilibrium, in fact we find that when supply and demand fluctuate and are uncertain, and when allocation changes are costly, the precise dynamics of a mechanism can greatly affect the efficiency of the whole system. An inappropriate mechanism – GV in this case – can be worse than simply sharing the resources equally among participants irrespective of need; an appropriate mechanism however can consistently do better for the community as a whole.

7. REFERENCES

[1] B. N. Chun and D. E. Culler. Market-based proportional resource sharing for clusters. Technical report, University of California at Berkeley, Berkeley, CA, USA, 2000.

[2] D. Ferguson, Y. Yemini, and C. Nikolau. Microeconomic algorithms for load balancing in distributed systems. In *Proc. of the* 8th *International Conference on Distributed Computer Systems*, pages 491–499. IEEE, 1988.

[3] H. Labs. SE3D project. http://www.hpl.hp.com/SE3D.

[4] K. Lai, B. A. Huberman, and L. Fine. Tycoon: A distributed market-based resource allocation system, 2004.

[5] D. M. Reeves, M. P. Wellman, J. K. MacKie-Mason, and A. Osepayshvili. Exploring bidding strategies for market-based scheduling. *Decision Support Systems*, 39:67–85, mar 2005.

[6] I. Stoica, H. Abdel-Wahab, and A. Pothen. A microeconomic scheduler for parallel computers. In *Proc IPPS '95 Workshop on Job Scheduling Strategies for Parallel Processing*, pages 122–135, apr 1995.

[7] Trading agent competition. http://www.sics.se/tac.

[8] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta. SPAWN: A distributed computational economy. *Software Engineering*, 18(2):103–117, 1992.

Robust Mechanisms for Information Elicitation

Aviv Zohar Jeffrey S. Rosenschein
School of Engineering and Computer Science
The Hebrew University of Jerusalem
Jerusalem, Israel
{avivz, jeff}@cs.huji.ac.il

ABSTRACT

We study information elicitation mechanisms in which a principal agent attempts to elicit the private information of other agents using a carefully selected payment scheme based on proper scoring rules. Scoring rules, like many other mechanisms set in a probabilistic environment, assume that all participating agents share some common belief about the underlying probability of events. In real-life situations however, underlying distributions are not known precisely, and small differences in beliefs about these distributions may alter agent behavior under the prescribed mechanism.

We propose designing elicitation mechanisms in a manner that will be robust to small changes in belief. We show how to algorithmically design such mechanisms in polynomial time using tools of stochastic programming and convex programming, and discuss implementation issues for multi-agent scenarios.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed AI—*Intelligent Agents, Multiagent Systems*;
H.1.1 [**Models and Principles**]: Systems and Information Theory—*Value of Information*

General Terms

Algorithms, Economics

Keywords

Information Elicitation, Computational Mechanism Design, Robust Mechanisms, Stochastic Programming, Scoring Rules

1. INTRODUCTION

In this paper we examine a scenario in which a principal agent is interested in purchasing information about some event from some other agent (or group of agents) that has private access to that information. The sellers are required to invest some effort in order to learn the information, and may be tempted to guess or report falsely if they expect to benefit from doing so. The buyer of information must therefore design its payments in a way that will induce truthfulness on the part of the sellers. This is ordinarily done using *Proper Scoring Rules* [2]. With a well-designed payment scheme, the expected utility of the sellers is maximized only when they invest the effort to learn the information and reveal it truthfully.

To construct such a mechanism, the designer must take into account the *beliefs* of the sellers about the probabilities of events, since these affect the cost-benefit analysis the sellers make. Unfortunately, these probabilities might not be common knowledge, and may in fact be secret information the agents do not wish to reveal.

We propose to design information elicitation mechanisms to be robust not only against manipulation by the participants, but also against small variations in the beliefs they may hold. The classic approach to dealing with variations in beliefs (or "type") of agents within mechanism design is the use of direct revelation mechanisms. These are mechanisms in which the participants reveal everything to the mechanism, which in turn acts optimally on their behalf — eliminating the need to lie. This approach is not appropriate in scenarios involving information elicitation where information is considered a commodity that is to be sold and not revealed freely.

2. THE SCENARIO

We model the information of agents using discrete random variables. Each seller i is assumed to own a private variable X_i that it can access at a cost of c_i. These variables are not necessarily independent. Once the transaction is complete, the buyer is given access to a random variable denoted Ω. The variable Ω is assumed to be somewhat coupled with the variables X_i, and provides a limited means of verification about their true values. Alternatively, one may think of Ω as some outcome that the buyer is attempting to predict, which eventually becomes known. We denote the governing probability distribution $Pr(\Omega = \omega, X_1 = x_1, \ldots, X_n = x_n)$ by $p_{\omega,x_1\ldots x_n}$. Payments to the agents are made after the value of Ω is revealed and may thus depend on the outcome, as well as the reports of all the agents (it is impossible to create the incentives for truthfulness without some measure of the correctness of the information provided). We denote the payment to agent i by $u^i_{\omega,x_1,\ldots,x_n}$. When dealing with only one agent, we shall drop the script i from all notations.

We require three things from a proper payment scheme. These are presented below in the case of a single agent:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. **Truth Telling.** Once an agent knows its variable is x, it must have an incentive to reveal it, rather than any lie x'.

$$\forall x, x' \quad s.t. \quad x \neq x' \quad \sum_{\omega} p_{\omega,x} \cdot (u_{\omega,x} - u_{\omega,x'}) > 0 \quad (1)$$

Here $p_{\omega,x}$ is the probability of what actually occurs, while the payment $u_{\omega,x'}$ is based only on what *the agent* reported.

2. **Individual Rationality.** An agent must have a positive expected utility from participating in the game:

$$\sum_{\omega,x} p_{\omega,x} \cdot u_{\omega,x} > c \quad (2)$$

3. **Investment.** The *value of information* for the agent must be greater than its cost. Any guess the agent makes without actually computing its value must be less profitable (in expectation) than revealing the variable:

$$\forall x' \quad \sum_{\omega,x} p_{\omega,x} \cdot (u_{\omega,x} - u_{\omega,x'}) > c \quad (3)$$

When a mechanism is designed for multiple agents, similar requirements should be considered. Their exact nature depends on the level of cooperation possible among the selling agents (transfer of utility, shared information, etc.). They can still be described in the form of linear constraints, but the number of constraints can sometimes be exponentially large in the number of agents.

2.1 Building Non-Robust Mechanisms

The three requirements above can all be characterized using linear constraints and can thus be solved efficiently using linear programming methods. Furthermore, a solution can be found that minimizes some target function such as the expected cost of the mechanism to the buyer.

A great deal of insight into the design problem can be obtained when considering the vectors defined by $\vec{p}_x \triangleq (p_{\omega_1,x} \ldots p_{\omega_k,x})$ and $\vec{u}_x \triangleq (u_{\omega_1,x} \ldots u_{\omega_k,x})$. Using this notation, the truthfulness constraints can be viewed as the requirement that the probability vectors $\vec{p}_x, \vec{p}_{x'}$ will be linearly separated by the vector of payments $\vec{u}_x - \vec{u}_{x'}$.

The following proposition shows necessary and sufficient conditions for the existence of a proper payment scheme.

PROPOSITION 1. *In the single agent case, a proper payment scheme exists iff the probability vectors $\vec{p}_x$ are pairwise independent. Furthermore, if any proper payment scheme exists then there is one with a mean cost as close to c as desired. Such a scheme is optimal, due to the individual rationality constraint.*

PROOF SKETCH. When two vectors $\vec{p}_x$, $\vec{p}_{x'}$ are linearly dependent, there is no way to satisfy the two truthfulness constraints for x, x'. If they are independent, then the truthfulness constraints can be satisfied (for example) by setting for any $\alpha > 0$ and any $\vec{\beta}$, $\vec{u}_x = \frac{\vec{p}_x}{||\vec{p}_x||} \cdot \alpha + \vec{\beta}$. α and $\vec{\beta}$ can be adjusted independently to satisfy the individual rationality and investment constraints tightly. Notice that this implies that the truthfulness constraints are the only things that might prevent us from finding a proper payment scheme. See [3] for a more detailed proof. □

3. DESIGNING ROBUST MECHANISMS

We shall now turn our attention to the case where the probabilities associated with the random variables Ω, X_i are not common knowledge. From now on, $\hat{p}_{\omega,x}$ shall denote the probabilities the principal agent believes in, and $p_{\omega,x}$ will denote the beliefs of an agent. Since beliefs about probabilities are usually grounded by observations, they are not likely to be very far from the truth. We shall assume that different beliefs are "close" to one another according to some distance metric: $\hat{p}_{\omega,x} = p_{\omega,x} + \epsilon_{\omega,x}$ and $||\vec{\epsilon}|| < \epsilon$. We have opted for the use of the L_∞ norm for measuring distance, since it is easily described using linear constraints, but the methods we present can be easily adapted for any other norm.

3.1 The Robustness of a Payment Scheme

Definition 1. We shall say that a given *payment scheme* $u_{\omega,x}$ is *ϵ-robust* with regard to an elicitation problem with distribution $\hat{p}_{\omega,x}$ if it is a proper solution to every elicitation problem with distribution $\hat{p}_{\omega,x} + \epsilon_{\omega,x}$ such that $\|\vec{\epsilon}\|_\infty < \epsilon$, and is infeasible for at least one problem instance of any larger norm.

The definition above is conservative. An ϵ-robust payment scheme will create the proper incentives for truthfulness for *every* possible belief variation of the agent that is within a distance of ϵ. This represents a "large margins" point of view, where the solution is required to satisfy all constraints with enough redundancy to handle small changes in them.

3.1.1 Determining the Robustness of a Given Scheme

Given a payment scheme $u_{\omega,x}$, and an elicitation problem with probabilities $p_{\omega,x}$, we can determine the robustness level of $u_{\omega,x}$ by finding out how much the probabilities must be perturbed to violate one of the constraints required for a truthful mechanism. We can do this by solving a linear program for every constraint. For example, the following program finds a perturbation that violates the truth-telling constraint for a secret x and a lie x'.

$$\begin{array}{ll} \min & \epsilon \\ \text{s.t.} & \\ & \sum_{\omega} (\hat{p}_{\omega,x} + \epsilon_{\omega,x})(u_{\omega,x} - u_{\omega,x'}) \leq 0 \\ \forall x, \omega & \hat{p}_{\omega,x} + \epsilon_{\omega,x} \geq 0 \\ & \sum_{\omega,x} \epsilon_{\omega,x} = 0 \\ \forall x, \omega & -\epsilon \leq \epsilon_{\omega,x} \leq \epsilon \end{array}$$

A solution to this program will be a small perturbation $\epsilon_{\omega,x}$ with a small norm ϵ that manages to violate the constraint. The requirements on $\epsilon_{\omega,x}$ assure us that the perturbed values are still legal probabilities. Once we solve a linear program for every constraint, we simply take the minimal value of ϵ that was found. This is the robustness level of the mechanism.

3.1.2 Efficiently Finding Some ϵ-Robust Mechanism

From a design point of view, we may be interested in finding a solution that is at least ϵ-robust for some ϵ and has a minimal cost. We can do this using tools for *Stochastic Programming*. A stochastic program is simply a mathematical program that contains uncertainty about the exact constraints that need to be satisfied, or the function that is optimized. The following stochastic program describes our problem:

$$
\begin{aligned}
\min \quad & \sum_{\omega,x} \hat{p}_{\omega,x} \cdot u_{\omega,x} \\
\text{s.t.} & \\
\forall x \neq x' \quad & \sum_{\omega} p_{\omega,x}(u_{\omega,x} - u_{\omega,x'}) > 0 \\
& \sum_{\omega,x} p_{\omega,x} \cdot u_{\omega,x} > c \\
\forall x' \quad & \sum_{\omega,x} p_{\omega,x}(u_{\omega,x} - u_{\omega,x'}) > c \\
\text{where:} & \\
\forall x, \omega \quad & p_{\omega,x} = \hat{p}_{\omega,x} + \epsilon_{\omega,x} \\
& p_{\omega,x} \geq 0 \\
& \sum_{\omega,x} p_{\omega,x} = 1 \\
& -\epsilon \leq \epsilon_{\omega,x} \leq \epsilon
\end{aligned}
$$

Stochastic programs with the conservative solution we require can be solved efficiently. [1] presents methods for doing so by treating the problem as a regular convex programming problem. Naturally, there can be cases where no solution exists for the given robustness level.

3.2 The Robustness Level of a Problem

Definition 2. We define the *robustness level* ϵ^* *of the problem* $\hat{p}$ as the supremum of all mechanism robustness levels ϵ for which there exists a proper mechanism:

$$\epsilon^* \triangleq \sup_{\vec{u}} \{\epsilon | \exists \vec{u} \text{ that is an } \epsilon\text{-robust mechanism for } \hat{p}\} \quad (4)$$

To find the robustness level of a problem, one can simply perform a binary search. The robustness level is certainly somewhere between 0 and 1. One may test at every desired level in between to see if there exists a mechanism with some specified robustness by solving the stochastic program above. The space between the upper and lower bounds is then narrowed according to the answer that was received.

3.2.1 A Bound for Problem-Robustness

A simple bound for robustness of the problem can be derived from examining the truthfulness conditions. In fact, Proposition 1 for non-robust mechanisms can be viewed as a private case of the following proposition when applied to 0-robust mechanisms:

PROPOSITION 2. *The robustness level* ϵ^* *of a problem* $\hat{p}$ *can be bounded by the smallest distance between a vector* $\hat{p}_x$ *and the optimal hyper-plane that separates it from* $\hat{p}_{x'}$*:*

$$\epsilon^* \leq \min_{x,x'} ||\hat{p}_x - (\hat{p}_x^{tr} \cdot \vec{\varphi}_{x,x'}) \cdot \vec{\varphi}_{x,x'}||_\infty \quad ; \quad \vec{\varphi}_{x,x'} = \frac{\hat{p}_x + \hat{p}_{x'}}{||\hat{p}_x + \hat{p}_{x'}||_2}$$

The optimal separating hyper-plane is a hyper-plane that separates the points and is of maximal (and equal) distance from both of them.

PROOF SKETCH. If there exist x, x' that give distance ϵ to the hyperplane then the vectors $\hat{p}_x, \hat{p}_{x'}$ can be perturbed towards the hyperplane with a perturbation of norm ϵ, until they are linearly dependent. For this problem instance, according to Proposition 1, there is no possible mechanism. □

In the case where $|\Omega| = 2$, the vectors $\hat{p}_x$ are situated in a two-dimensional plane, and it can be shown that the bound given above is tight — the problem robustness is determined exactly by the closest pair of vectors.

3.3 Mechanisms for Multiple Agents

We assume here that sellers cannot collude, share information, or transfer utility among themselves. Models that allow for collusion are interesting to explore from the viewpoint of robustness, but are beyond the scope of this paper.

When designing mechanisms for multiple agents, the designer has to choose the solution concept it wishes to achieve. It is sometimes possible to design mechanisms that will provide the agents with dominant strategies of truth telling — each agent is better off telling the truth no matter what the other agents do. However, in some cases this is not possible, and the designer must implement a mechanism that provides for good behavior only in equilibrium, when *all* players are behaving well. In these cases the payments made to the players utilize the information given by their peers. Each solution concept translates into a different linear program that may provide a different answer.

In this case, the design of a robust mechanism becomes even more difficult. The designer must now take into account not only the possible beliefs of agents about the probabilities of events, but also their beliefs about the beliefs of other agents. For an agent to believe that some strategy is in equilibrium, it must also be convinced that its counterparts believe that their strategies are in equilibrium, or are otherwise optimal. This will only occur if the agent believes that they believe that it believes that its strategy is in equilibrium — and so on to infinity. Any uncertainty about the beliefs of other agents grows with every step up the belief hierarchy. If agent A knows that all agents have some radius ϵ of uncertainty in beliefs, and its view of the world consists of some probability distribution p it assigns to events, then it is possible that agent B believes the distribution is p' and further believes that agent A believes the distribution is some p'' which is at a distance of up to 2ϵ from p. With an infinite belief hierarchy, it is therefore possible to reach any probability if we go high enough in the hierarchy.

A possible solution to this problem is to use a mixture of solution concepts. The mechanism can often be designed to make each agent's payment depend only on a subset of agents that precedes it. In this case it only needs to take their beliefs into consideration when deciding on a strategy. The necessary belief hierarchy is then finite, which limits the possible range of beliefs about beliefs. The most extreme case of this is to design the mechanism for dominant strategies only. Naturally, a solution constructed in such a way may be less efficient or may not exist at all. Another possibility is to consider only bounded agents that can only consider a finite number of levels in the belief hierarchy.

4. ACKNOWLEDGMENT

This work was partially supported by grant #039-7582 from the Israel Science Foundation.

5. REFERENCES

[1] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25:1–13, 1999.

[2] T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. Technical Report 463, Department of Statistics, Univ. of Washington, 2004.

[3] A. Zohar and J. S. Rosenschein. Robust information elicitation mechanisms. Technical Report 2006-3, Leibniz Institute, Hebrew Univ. of Jerusalem, 2006.

A Comparative Study of Sequential and Simultaneous Auctions

Shaheen S. Fatima
Department of Computer Science
University of Liverpool
Liverpool L69 7ZF, U.K
shaheen@csc.liv.ac.uk

ABSTRACT

This paper compares the sequential and simultaneous auctions for the following scenario. There are multiple similar objects for sale, each object is sold in a separate auction, and each bidder needs only one object. We use English auction rules and first determine equilibrium bidding strategies for the simultaneous and sequential cases. We do this for both common and private value objects by treating a bidder's information about these values as *uncertain*. We then consider the case where the private and common values have a uniform distribution and compare the two mechanisms in terms of three key properties: a bidder's ex-ante expected profit, the auctioneer's expected cumulative revenue, and the total expected surplus. For both common and private value objects, our study shows the following result. The expected cumulative revenue and the expected total surplus is higher for the sequential mechanism. However, a bidder's exante expected profit depends on the number of objects being auctioned and the number of participating bidders, and it is sometimes higher for the sequential mechanism and sometimes for the simultaneous one.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; K.4.4 [**Computers and Society**]: Electronic Commerce

General Terms

Algorithms, Design, Economics

Keywords

Game-theory, Auctions, Multi-object auctions.

1. INTRODUCTION

Multiple objects can be auctioned using a number of mechanisms. Now the bidding behaviour is different for different mechanisms. So the outcome of such auctions (i.e., who wins and at what price) is also different for different mechanisms. Hence, the auctioneer and the bidding agents want to know which mechanism will be the most beneficial to them. To this end, we study and compare two different mechanisms: *sequential* and *simultaneous*. To date, considerable research effort has been devoted to the study of these mechanisms. But the main limitation of existing work is that it has studied sequential and simultaneous auctions in isolation and so far there has been very little work on the comparison of these two mechanisms in terms of their outcomes. Our objective is therefore to provide a comparative analysis of sequential (SEQ) and simultaneous (SIM) auctions in an incomplete information setting. We do this for both common value (CV) and private value (PV) objects. For both CV and PV objects, our study shows the following result. The expected cumulative surplus and the auctioneer's cumulative revenue is higher for the SEQ mechanism. However, the mechanism that generates a higher ex-ante expected profit for the bidders depends on the number of objects being auctioned and the number of participating bidders, and it is sometimes higher for SEQ mechanism and sometimes for the SIM one.

Section 2 determines equilibrium for the SEQ and SIM mechanisms. Section 3 compares the outcomes for the two mechanisms. Section 4 concludes.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profi t or commercial advantage and that copies bear this notice and the full citation on the fi rst page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specifi c permission and/or a fee.
AAMAS'06 May 8-12, 2006, Hakodate, Japan
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. EQUILIBRIUM BIDDING STRATEGIES

Equilibrium bidding strategies for a single object with both common and private values have been obtained in [4]. We therefore briefly summarize its key results and then determine the equilibrium for our m objects case.

2.1 Single object

We first describe the CV case. For a single object with CV V_1, the equilibrium obtained in [4] is as follows. Let f^n denote the first order statistic of the surplus for the n bidders and s^n the second order statistic. Then the bidder with the highest surplus wins and the winner's expected profit is $EP_w(n) = E(f^n) - E(s^n)$. The expected surplus is $ES(n) = E(V_1)$. Finally, the expected revenue is $ER(n) = ES(n) - EP_w(n)$.

On the other hand, for PV objects [6], the winner's expected profit is $\overline{EP}_w(n) = E(\overline{f}^n) - E(\overline{s}^n)$. The expected surplus is $\overline{ES}(n) = E(\overline{f}^n)$ and the expected revenue is $\overline{ER}(n) = E(\overline{s}^n)$ where $\overline{f}^n$ and $\overline{s}^n$ denote the first and second order statistic for the private value of the object.

2.2 Multiple sequential auctions

The m similar objects are sold in m separate English auctions conducted sequentially one after another. Each bidder needs only one object and draws its CV/PV for an auction just before that auction begins. Since a bidder needs only one object, it participates in the series of auctions until it wins an object. After that, it does not take

part in any of the remaining auctions[1]. In our model, there is a single probability distribution function for the CVs and PVs of all the objects and each bidder receives its signals for an object before the auction for that object begins (like in [5, 1]).

The English auction rules are as follows. The auctioneer continuously raises the price, and bidders publicly reveal when they withdraw from the auction. Bidders who drop out from an auction cannot re-enter that auction. A bidder's strategy for the jth (for $1 \leq j \leq m-1$) auction depends on its expected profit from the $(m-j)$ future auctions. Since the mth auction is the last one, a bidder's strategy for this auction is the same as the single object case. But the bidding behaviour for the first $(m-1)$ auctions is different from the single object case.

We first obtain the equilibrium bids for CV objects and then extend it to PVs. For the former case, a bidder's strategy for an individual auction depends on its probability of winning any one of the remaining auctions that are yet to be conducted. If the number of bidders for the first auction is n, then let $\beta(y, j, m, n)$ denote a bidder's ex-ante probability of winning the yth (for $j \leq y \leq m$) auction in the series from the jth to the mth auction before the jth one begins. This probability is:

$$\beta(y, j, m, n) = [\Pi_{k=j}^{y-1}(1 - 1/(n-y+k+1))] \times (1/(n-y+1))$$

For CV objects, let $EP_w(j, m, n)$ denote the winner's expected profit for the jth auction in a series of m auctions with n bidders for the first one. Also, let $\alpha(j, m, n)$ denote a bidder's ex-ante expected profit for winning any one auction in the series from the jth (for $1 \leq j \leq m$) to the mth auction where

$$\alpha(j, m, n) = \Sigma_{y=j}^{m} \beta(y, j, m, n) EP_w(y, m, n)$$

For CV objects, the n-tuple of strategies $(B(\cdot), \ldots, B(\cdot))$ with $B(\cdot)$ defined below constitutes a symmetric equilibrium for the jth (for $1 \leq j < m$) auction at a stage where k bidders have dropped out (see [3] for details):

$$B_0^j(x_{ij}) = E(v_{ij}|S_{ij} = x_{ij}) - \alpha(j+1, m, n)$$

$$B_k^j(x_{ij}; b_1, \ldots, b_k) = \frac{n-j+1-k}{n-j+1} E(v_{ij}|S_{ij} = x_{ij}) + \frac{1}{n-j+1} \sum_{y=0}^{k-1} E(v_{ij}| \; B_y(x_{ij}; b_1, \ldots, b_y) = b_{y+1}) - \alpha(j+1, m, n)$$

where x_{ij} is bidder i's surplus for the jth object.

Let $ER(j, m, n)$ denote the expected revenue from the jth auction in a series of m CV auctions with n bidders for the first one. Also, let $ES(j, m, n)$ denote the expected surplus for the jth auction. Then for the jth (for $1 \leq j \leq m$) auction, the winner's expected profit ($EP_w(j, m, n)$), the expected surplus ($ES(j, m, n)$), and the expected revenue ($ER(j, m, n)$) are:

$$EP_w(j, m, n) = E(f^{n-j+1}) - E(s^{n-j+1}) + \alpha(j+1, m, n)$$
$$EP_w(m, m, n) = E(f^{n-m+1}) - E(s^{n-m+1})$$
$$ES(j, m, n) = E(V_j)$$
$$ER(j, m, n) = ES(j, m, n) - EP_w(j, m, n)$$

[1]This model is an extension of [1]: [1] analyses a sequence of two private value auctions while we analyse $n \geq 2$ auctions for common and private values.

We now find the cumulative revenue, the cumulative surplus, and a bidder's ex-ante expected profit from all m auctions. For $n > m$, let $X_{seq}(m, n)$ denote the cumulative revenue and $Y_{seq}(m, n)$ a bidder's exante profit from all m auctions where $X_{seq}(m, n) = \Sigma_{j=1}^{m} ER(j, m, n)$ and $Y_{seq}(m, n) = \alpha(1, m, n)$. In general (i.e., $n > m$ or $n \leq m$), the cumulative revenue ($ECR_{seq}(m, n)$) is as follows:

$$ECR_{seq}(m, n) = \begin{cases} 0 & \text{if } n = 0, \text{ or } n = 1 \\ X_{seq}(m, n) & \text{if } n > m \\ X_{seq}(n-1, n) & \text{if } n \leq m \end{cases}$$

This is because if $n \leq m$, then the number of bidders for the first $n-1$ auctions is at least 2. For the nth auction there is only one bidder and so it wins the object for nothing resulting in a revenue of zero. For all the remaining $m-n$ auctions, there are no bidders and so the revenue is zero again.

Let $EEP_{seq}(m, n)$ denote a bidder's ex-ante expected profit for all the m auctions where

$$EEP_{seq}(m, n) = \begin{cases} E(V) & \text{if } n = 1 \\ Y_{seq}(m, n) & \text{if } n > m \\ Y_{seq}(n-1, n) + \beta(n, 1, n, n)E(V) & \text{if } n \leq m \end{cases}$$

This is because if ($n \leq m$) for the first $n-1$ auctions there are at least two bidders. But for the nth auction, since there is only one bidder, it gets the object for nothing. For this auction, the winner's expected profit is the expectation of the CV. Since the CVs of all the objects are drawn from a single distribution function (F), we drop the subscript in $E(V)$.

Finally, the expected cumulative surplus is:

$$ECS_{seq}(m, n) = \begin{cases} mE(V) & \text{if } n \geq m \\ nE(V) & \text{if } n < m \end{cases}$$

We now turn to PV objects. For these objects, we denote the first and second order statistic of the PV for the jth auction as $\overline{f}^{n-j+1}$ and $\overline{s}^{n-j+1}$ respectively. For these objects, a bidder's bid for an auction is obtained by subtracting its ex-ante expected profit for the future auctions from its bid for the current one. Hence, for the jth auction, the winner's expected profit (denoted $\overline{EP_w}(j, m, n)$) is:

$$\overline{EP}_w(j, m, n) = E(\overline{f}^{n-j+1}) - E(\overline{s}^{n-j+1}) + \overline{\alpha}(j+1, m, n)$$
$$\overline{EP}_w(m, m, n) = E(\overline{f}^{n-m+1}) - E(\overline{s}^{n-m+1})$$

where $\overline{\alpha}(j, m, n)$ is a bidder's ex-ante expected profit for winning any one auction in the series from the jth (for $1 \leq j \leq m$) to the mth auction and is:

$$\overline{\alpha}(j, m, n) = \Sigma_{y=j}^{m} \beta(y, j, m, n) \overline{EP}_w(y, m, n)$$

The expected revenue and the expected surplus are:

$$\forall_{j=1}^{m-1} \overline{ER}(j, m, n) = E(\overline{s}^{n-j+1}) - \overline{\alpha}(j+1, m, n)$$
$$\overline{ER}(m, m, n) = E(\overline{s}^{n-m+1})$$
$$\forall_{j=1}^{m} \overline{ES}(j, m, n) = \overline{ER}(j, m, n) + \overline{EP}_w(j, m, n)$$

For $n > m$, let $\overline{X}_{seq}(m, n)$ denote the expected cumulative revenue for all the m objects and $\overline{Y}_{seq}(m, n)$ a bidder's ex-ante expected profit from all the m auctions where $\overline{X}_{seq}(m, n)$ is equal to $\Sigma_{j=1}^{m} \overline{ER}(j, m, n)$ and $\overline{Y}_{seq}(m, n) = \overline{\alpha}(1, m, n)$. In general, (i.e., for $n > m$ or $n \leq m$), the cumulative revenue is:

$$\overline{ECR}_{seq}(m, n) = \begin{cases} 0 & \text{if } n = 0, \text{ or } n = 1 \\ \overline{X}_{seq}(m, n) & \text{if } n > m \\ \overline{X}_{seq}(n-1, n) & \text{if } n \leq m \end{cases}$$

and a bidder's ex-ante expected profit ($\overline{EEP}_{seq}(m,n)$) is:

$$\overline{EEP}_{seq}(m,n) = \begin{cases} E(\overline{f}^1) & \text{if } n = 1 \\ \overline{Y}_{seq}(m,n) & \text{if } n > m \\ \overline{Y}_{seq}(n-1,n)+ & \\ \beta(n,1,n,n)E(\overline{f}^1) & \text{if } n \leq m \end{cases}$$

Finally, the expected cumulative surplus is:

$$\overline{ECS}_{seq}(m,n) = \begin{cases} \sum_{j=1}^{m} E(\overline{f}^{n-j+1}) & \text{if } n \geq m \\ \sum_{j=1}^{n} E(\overline{f}^{n-j+1}) & \text{if } n < m \end{cases}$$

2.3 Multiple simultaneous auctions

As before, we have $m > 1$ similar objects, n risk-neutral bidders, and each bidder needs only one object. Given this, the SIM auctions game is played as follows. The m objects are sold in m different auctions conducted simultaneously and independently of each other. Each bidder randomly selects one of the m auctions and bids in it. It is obvious that after a bidder selects an auction, its equilibrium bids are the same as that for a single object.

This game has been analysed in [2] for private value objects for the limiting case where both m and n tend to infinity. The analysis is done by assuming complete information. Here, we extend this analysis to a more realistic case where both m and n are finite, the objects are either CV or PV, and there is uncertainty about these values. For this SIM auctions game, the equilibrium outcome depends on how the n bidders arrive at the m auctions. We let T_m^n denote the number of different ways in which n bidders can be distributed between m auctions where

$$T_m^n = \begin{cases} 1 & \text{if } n = 0 \text{ or } m = 1 \\ \Sigma_{k=0}^{n} C(n,k) & \text{if } m = 2 \\ \Sigma_{k=0}^{n} C(n,k) T_{m-1}^{n-k} & \text{if } m > 2 \end{cases}$$

For m SIM auctions, the probability that k bidders arrive at an auction ($P(m,k)$) is:

$$P(m,k) = (C(n,k)T_{m-1}^{n-k})/T_m^n$$

Consider any one auction. The number of bidders for this auction can vary between 0 and n. If no bidders arrive at the auction, the object remains unsold. If only one bidder arrives, it gets the object for nothing. If $k \geq 2$ bidders arrive, then the expected revenue is greater than zero.

Consider CV objects. For these objects, the expected cumulative revenue from all the m auctions ($ECR_{sim}(m,n)$) is:

$$\begin{aligned} ECR_{sim}(m,n) &= \Sigma_{k=0}^{n} P(m,k)[ER(k)+ \\ &\quad ECR_{sim}(m-1,n-k)] \end{aligned}$$

Also, for an individual auction, the winner's expected profit is:

$$\begin{aligned} EP_w(m,n) &= P(m,1)[E(V)]+ \\ &\quad \Sigma_{k=2}^{n} P(m,k)[E(f^k) - E(s^k)] \end{aligned}$$

and hence a bidder's ex-ante expected profit ($EEP_{sim}(m,n)$) is:

$$\begin{aligned} EEP_{sim}(m,n) &= P(m,1)[E(V)]+ \\ &\quad \Sigma_{k=2}^{n} P(m,k)(1/k)[E(f^k) - E(s^k)] \end{aligned}$$

Finally, the total expected surplus (denoted $ECS_{sim}(m,n)$) is:

$$\begin{aligned} ECS_{sim}(m,n) &= \Sigma_{k=1}^{n} P(m,k)[ES(k)+ \\ &\quad ECS_{sim}(m-1,n-k)] \end{aligned}$$

Now consider PV objects. For these objects, the expected cumulative revenue is:

$$\begin{aligned} \overline{ECR}_{sim}(m,n) &= \Sigma_{k=0}^{n} P(m,k)[\overline{ER}(k)+ \\ &\quad \overline{ECR}_{sim}(m-1,n-k)] \end{aligned}$$

Also, for an auction, the winner's expected profit is:

$$\begin{aligned} \overline{EP}_w(m,n) &= P(m,1)[E(\overline{f}^1)]+ \\ &\quad \Sigma_{k=2}^{n} P(m,k)[E(\overline{f}^k) - E(\overline{s}^k)] \end{aligned}$$

and hence a bidder's ex-ante expected profit is:

$$\begin{aligned} \overline{EEP}_{sim}(m,n) &= P(m,1)[E(\overline{f}^1)]+ \\ &\quad \Sigma_{k=2}^{n} P(m,k)(1/k)[E(\overline{f}^k) - E(\overline{s}^k)] \end{aligned}$$

Finally, the total expected surplus is:

$$\begin{aligned} \overline{ECS}_{sim}(m,n) &= \Sigma_{k=1}^{n} P(m,k)[\overline{ES}(k)+ \\ &\quad \overline{ECS}_{sim}(m-1,n-k)] \end{aligned}$$

3. A COMPARISON OF THE TWO AUCTION MECHANISMS

We now compare the two mechanisms in terms of three key properties: the expected cumulative revenue, the expected cumulative surplus, and a bidder's ex-ante expected profit. To do this, we consider the uniform distribution function for both PVs and CVs. For this distribution, our study shows that from an auctioneer's perspective, the SEQ mechanism is better because it generates a higher revenue. From a global perspective too, the SEQ mechanism is better (because it generates a higher expected cumuative surplus). However, from a bidder's perspective, the ex-ante expected profit depends on the number of participating bidders and the number of objects being auctioned, and it is sometimes higher for the SEQ mechanism and sometimes for the SIM one.

4. CONCLUSIONS

This paper analyses two key auction mechanisms for multiple objects: *sequential* and the *simultaneous*. This analysis is done for both common and private value objects by treating a bidder's information about these values as uncertain. We used the English auction rules and determined equilibrium for the two mechanisms. We then considered the case where the private and common values have a uniform distribution and compared the mechanisms in terms of their expected cumulative revenues, their expected cumulative surplus, and a bidder's ex-ante expected payoff. For both common and private values, our study shows that, the sequential mechanism is better both from the auctioneer's and also from a global perspective. However, from a bidder's perspective, the choice of a mechanism depends on the number of objects and the number of bidders.

5. REFERENCES

[1] D. Bernhardt and D. Scoones, A note on sequential auctions, *American Economic Review*, 84(3):653–657, 1994.

[2] R.Engelbrecht-Wiggans and R. J. Weber, An example of a multi-object auction game, *Management Science*, 25(12):1272–1277, 1979.

[3] S. S. Fatima and M. Wooldridge and N. R. Jennings, Sequential auctions for objects with common and private values, In *Proceedings of the Fourth International Conference on Autonomous Agents and Multi-Agent Systems*, pages 635–642, Utrecht, Netherlands, 2005.

[4] J. K. Goeree and T. Offerman, Competitive bidding in auctions with private and common values, *The Economic Journal* 113(489):598–613, 2003.

[5] A. Ortega-Reichert, Models of competitive bidding under uncertainty, *Stanford University Technical Repeort*, 8, 1968.

[6] W. Vickrey, Counterspeculation, auctions and competitive sealed tenders, *Journal of Finance* 16:8–37, 1961.

Competing Sellers in Online Markets: Reserve Prices, Shill Bidding, and Auction Fees

Enrico H. Gerding, Alex Rogers, Rajdeep K. Dash and Nicholas R. Jennings
University of Southampton, Southampton, SO17 1BJ, UK.
{eg,acr,rkd02r,nrj}@ecs.soton.ac.uk

ABSTRACT

In this paper, we consider competition between sellers offering similar items in concurrent online auctions, where each seller must set its individual auction parameters (such as the reserve price) in such a way as to attract buyers. We show that there exists a pure Nash equilibrium in the case of two sellers with asymmetric production costs. In addition, we show that, rather than setting a reserve price, a seller can further improve its utility by shill bidding (i.e., pretending to be a buyer in order to bid in its own auction). But, using an evolutionary simulation, we show that this shill bidding introduces inefficiences within the market. However, we then go on to show that these inefficiences can be reduced when the mediating auction institution uses appropriate auction fees that deter sellers from submitting shill bids.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]

General Terms

Economics

Keywords

Mechanism Design, Competition, Shill Bidding, Auction Fees

1. INTRODUCTION

Online markets are becoming increasingly prevalent and extend to a wide variety of areas such as e-commerce, Grid computing, recommender systems, and sensor networks. To date, much of the existing research has focused on the design and operation of individual auctions or exchanges for allocating goods and services. In practice, however, similar items are typically offered by multiple independent sellers that compete for buyers and set their own terms and conditions (such as their reserve price and the type and duration of the auction) within an institution that mediates between buyers and sellers. Examples of such institutions include eBay, Amazon and Yahoo!, where at any point in time multiple concurrent auctions with different settings are selling similar objects, often resulting in strong competition. Given this competition, a key research question is how a seller should select their auction settings in order to best attract buyers and so increase their expected profits. In this paper, we consider this issue in terms of setting the seller's reserve price (since the role of the reserve price has received attention in both competitive and non-competitive settings). In particular, we extend the existing analysis by considering how sellers may improve their profit by shill bidding (i.e. bidding within their own auction as a means of setting an implicit reserve price). Moreover, we investigate how the institution can deter this undesirable shill bidding through the use of appropriate auction fees.

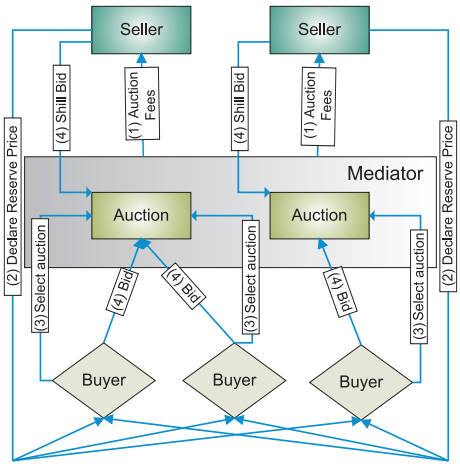


Figure 1: The competing sellers game.

2. MODEL OF COMPETING SELLERS

The model of competing sellers proceeds in four stages (see figure 1). First, the mediator (an institution such as eBay or Yahoo! that runs the auctions) announces the auction fees to the sellers. The sellers then simultaneously post their reserve prices in the second stage. In the third stage, the buyers simultaneously select an auction (or, equivalently, a seller) based on the observed reserve prices. In the final stage, the buyers (and possibly the sellers who are shill bidding) submit bids and the auctions are executed concurrently.

3. ANALYSIS

A complete analysis of equilibrium behaviour and market efficiency for the complete model is intractable [3]. Therefore, in this section, we analyse a simplified version with

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

two sellers and no auction fees (in section 4 we address the complete model). We assume that there are N risk neutral buyers, each of whom requires just one item. Each buyer has valuation v independently drawn from a commonly known cumulative distribution F with density f and support $[0, 1]$. Each risk neutral seller offers one item for sale, has production costs x_i, and decides upon a reserve price r_i and shill bid s_i. The preferences of buyers and sellers are described by von Neumann and Morgenstern utility functions.

3.1 Buyer Equilibrium Behaviour

The buyers' behaviour for two sellers has been analysed in [2]. A rational buyer with valuation $v < r_1$ will not attend any auction. Furthermore, if $r_1 < v < r2$, the buyer will always go to seller 1. The interesting case occurs when $v > r2$. In a symmetric Nash equilibrium, there is a unique cut-off point $1 \geq w \geq r2$ where buyers with $v < w$ will always go to seller 1, and buyers with $v \geq w$ will randomize equally between the two auctions. The cut-off point w is exactly where a buyer's expected utility is equal for both auctions, and is thus found by solving:

$$r_1\mathcal{F}(r_1, w)^{N-1} + (N-1)\int_{r_1}^{w} y\mathcal{F}(y, w)^{N-2}dF(y) = r_2\mathcal{F}(w, w)^{N-1}$$

where $\mathcal{F}(y, w) = F(y) + [1 - F(w)]/2$. Given the buyers' cut-off point, we can now calculate the sellers' expected revenue.

3.2 Seller Equilibrium Behaviour

To calculate the equilibrium behaviour of the sellers, we derive a general expression for the sellers' expected utility. This is calculated by considering the probability of one of three events occurring: (i) no bidders having valuations above the reserve price and the item does not sell, (ii) only one bidder having a valuation above the reserve price and the item sells at the reserve price, or (iii) two or more bidders having valuations above the reserve price and the item sells at a price equal to the second highest valuation. Thus, the expected utility of seller i who has a production cost of x_i and sets a reserve price of r_i is

$$U_i(r_i, x_i) = N(r_i - x_i)\mathcal{G}(r_i)(1 - \mathcal{G}(r_i))^{N-1} + N(N-1)\int_{r_1}^{1}(x_i - y)\mathcal{G}'(y)\mathcal{G}(y)(1 - \mathcal{G}(y))^{N-2}dy \quad (1)$$

where $\mathcal{G}(y)$ is the probability that a bidder is present in the auction *and* that this bidder has a valuation greater than y.

Now, in the standard auction with no competing sellers, we have the standard result that $\mathcal{G}(y) = 1 - F(y)$ and $\mathcal{G}'(y) = -f(y)$. However, for two competing sellers, we must account for the fact that the number and valuation of the bidders in the auction is determined by the bidders' cut-off point w. Thus, for sellers 1 and 2 (where seller 1 has the lower reserve price), $\mathcal{G}_1$ and $\mathcal{G}_2$ are given by:

$$\mathcal{G}_1(y) = \begin{cases} \frac{1+F(w)}{2} - F(y) & y < w \\ \frac{1-F(y)}{2} & y \geq w \end{cases} \quad \mathcal{G}_2(y) = \begin{cases} \frac{1-F(w)}{2} & y < w \\ \frac{1-F(y)}{2} & y \geq w \end{cases} \quad (2)$$

Thus, the sellers' expected utility depends on the reserve price of both sellers and the equilibrium behaviour is complex. We now apply this result to three different cases: (i) where both sellers declare public reserve prices, (ii) where one seller declares a public reserve price and the other submits a shill bid, and (iii) where both sellers shill bid[1].

[1]When a seller shill bids, the declared reserve price has no additional benefit. Thus we assume they declare no reserve price (or, equivalently, declare a zero reserve price).

		Seller 2	
		RP	SB
Seller 1	RP	0.452 , 0.189	0.403 , 0.220
	SB	0.457 , 0.188	0.423 , 0.220

Table 1: Sellers' expected utility when either declaring a reserve price (RP) or to shill bidding (SB).

Both Sellers Announce Public Reserve Prices. In this case, the equilibrium strategy of each seller is given by a Nash equilibrium at which each seller's reserve price is a utility maximising best response to the reserve price of the competing seller. When $x_1 = x_2$, no pure strategy Nash equilibrium exists [2]. However, when the sellers have sufficiently different production costs, we find that a pure Nash equilibrium exists where the reserve price of both sellers is higher than their production costs. We find this equilibrium numerically by iteratively discretising the space of possible reserve prices. That is, for all possible values of r_1 and r_2 that satisfy the conditions $x_1 \leq r_1 \leq 1$ and $r1 \leq r_2 \leq 1$, we calculate w and hence the expected utility of the two sellers. We then search these reserve price combinations to find the values of r_1^* and r_2^* that represents the utility maximising best responses to one another. By iterating the process and using a finer discretisation at each stage, we are able to calculate the Nash equilibrium to any degree of precision. The outcomes show that the symmetric case is very much a special case, and the majority of possible production cost combinations yield unique pure strategy Nash equilibria.

One Seller Shill Bids. Rather than announce a public reserve price, either seller may choose to announce a reserve price of zero to attract bidders, and then submit a shill bid to prevent the item from selling at too low a price. Thus, the seller who does not shill bid (seller 2 since r_2 will be greater than r_1) should declare a reserve price that is a best response to the zero reserve price announced by the bidder who does shill bid. This reserve price is simply given by the value of r_2 that maximises $U_2(r_2, x_2)$, given that we calculate $\mathcal{G}_2(y)$ as in equation 2 and take $r_1 = 0$ in order to calculate w. Given the best response reserve price of seller 2, and the resulting value of w, we can also calculate the shill bid that seller 1 should submit in order to maximise its own expected utility. By substituting s_1 for r_1 in equation 1, and using $\mathcal{G}_1(y)$ as given in equation 2, we find the shill bid that maximises $U_1(s_1, x_1)$.

Both Sellers Shill Bid. Finally, when both sellers declare a zero reserve price and shill bid, the bidders will randomise equally between either auction, since there is no reserve price information to guide their decision. Thus we find the equilibrium shill bids of both sellers by again substituting s_i for r_i in equation 1 and hence finding the value of s_i that maximises $U_i(s_i, x_i)$ when $w = 0$.

Table 1 shows an example of the resulting four strategy combinations as a normal form game (in this case $N = 10$, $x_1 = 0.25$, and $x_2 = 0.5$). Note that both sellers have a dominant strategy to submit shill bids (this result holds in general in the absence of auction fees). At this equilibrium, seller 2 achieves its maximum possible utility. However, seller 1 receives more when neither seller shill bids and is thus better off with a mechanism that deters all parties from submitting shill bids.

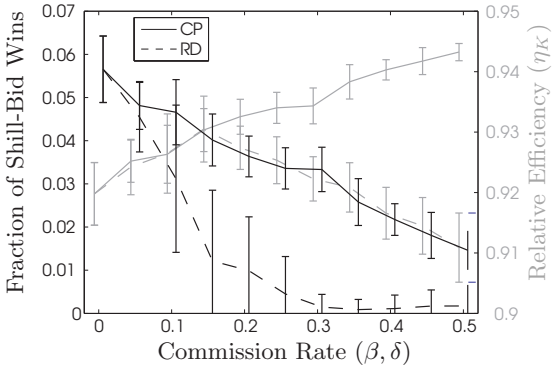


Figure 2: Average fraction of shill-bid wins (black) and relative efficiency (grey) for CP & RD auction fees ($N = 10$ and $x_1 = x_2 = 0$). The error-bars indicate the standard deviation.

4. AUCTION FEES

We now consider auction fees and market efficiency in the competing sellers game. We compare two types of auction fees: a closing price (CP) fee that is a fraction, β, of the selling price (where β is the CP commission rate), and a reserve-difference (RD) fee that is calculated as a fraction, δ, of the difference between the selling price and the seller's declared reserve price, (where δ is the RD commission rate). The first type of fee is the most common in online auctions such as eBay, Yahoo! and Amazon. The second type of fee was introduced in previous literature, and is shown to prevent shilling for particular bidder valuation distributions in a single multi-stage auction [4].

Auction fees add considerable complexity to the analysis of the competing sellers game since a seller now needs to optimally set both the reserve price and the shill bid. Therefore, we investigate auction fees using a simulation based on evolutionary algorithms (EAs) [1]. The EA maintains a population of possible seller strategies, where a strategy determines the shill bid and reserve price for each auction. At each generation, M seller strategies are randomly selected from the population and compete against one another in a number of consecutive games. The fittest strategies survive and are transferred to the next generation, whereas poor performing strategies are removed from the population. New strategies are explored by slightly modifying existing individuals using a mutation operator. This evolutionary process is repeated for a fixed number of iterations.

We compare auction fees by considering the fraction of *shill-bid wins* (i.e., the fraction of auctions that result in a shill bid being the highest bid), and a measure of the relative efficiency η_K of an allocation K, where η_K is given by:

$$\eta_K = \frac{\sum_{i=1}^{N} v_i(K) + \sum_{i=1}^{M}(x_i - x_i(K))}{\sum_{i=1}^{N} v_i(K^*) + \sum_{i=1}^{M}(x_i - x_i(K^*))}, \qquad (3)$$

where $K^* = \arg\max_{k \in \mathcal{K}}[\sum_{i=1}^{N} v_i(k) - \sum_{i=1}^{M} x_i(k)]$ is an efficient allocation, $\mathcal{K}$ is the set of all possible allocations, $v_i(k)$ is bidder i's utility for an allocation $k \in \mathcal{K}$, and $x_i(k)$ is seller i's production costs for a given allocation (in order to prevent a negative value we add production costs x_i in both the denominator and the numerator).

Now, in order to directly compare the auction fees, we first consider the case without production costs. To this end, figure 2 shows the fraction of shill-bid wins and relative efficiency for the two types of auction fees. These results show that the RD fee is better able to reduce shill bidding for this setting. However, in case of high commission rates, a higher efficiency is obtained using the CP fee. Since the RD fee is effectively a combination of a CP fee and a negative RD fee, it rewards a high reserve price. If the commission rate is sufficiently high, the reserve price is set higher than the equilibrium price without any auction fees, and thus, results in an inefficient outcome. The CP fee, on the other hand, is neutral with regard to the reserve price.

In case of production costs we find similar results. However, if both production costs and CP commission rates are very high (e.g. above 30%), a sharp decline in efficiency is observed using the CP fee. A seller will then set a very high reserve price to prevent making a loss in case it sells, and, as a result, efficiency becomes very low. Using the RD fee, on the other hand, efficiency does not decrease because sellers never make a loss as long as they report a reserve price above their production costs. The latter type of fee is therefore more robust to high production costs.

To conclude, the experiments show that in most cases highest efficiency is obtained by the commonly applied CP auction fees. The RD auction scheme, however, is more effective in deterring shill bidding and is also more robust to high production costs. This is consistent with earlier results showing that RD auction fees can deter shill bidding for isolated auctions [4]. However, our results show, for the first time, that these fees are also effective for a setting where sellers compete and in case of production costs. Moreover, we see that, when using the RD fee, sellers pay much less to the mediator overall compared to CP auction fees. The latter is especially important in a larger setting where multiple mediating institutions compete to attract sellers, since high fees will deter potential sellers.

5. CONCLUSIONS

Traditionally, competition among sellers has been ignored when designing auctions and setting auction parameters. However, when faced with competition, we have shown that auction parameters are important in determining the number and type of buyers that are attracted to an auction. We have also shown that such competition provides an incentive for sellers to shill bid, but this can be avoided by a mediator that applies appropriate auction fees. These results are particularly relevant for online markets and multi-agent systems, where competition is strong due to the ease with which a buyer (or a software agent) can search for particular goods. Thus, in these settings, our results can be used by sellers seeking to maximise their profit, or alternatively, by the auction institution itself, who wishes to use appropriate auction fees to deter shill bidding and thus increase the efficiency of the market as a whole.

6. REFERENCES

[1] S.M. Bohte, E. Gerding, and J.A. La Poutré. Market-based recommendation: Agents that compete for consumer attention. *ACM Transactions on Internet Technology*, 4:420–448, 2004.

[2] R. Burguet and J. Sákovics. Imperfect competition in auction design. *International Economic Review*, 40(1):231–247, 1999.

[3] R. Preston McAfee. Mechanism design by competing sellers. *Econometrica*, 61(6):1281–1312, 1993.

[4] W. Wang, Z. Hidvégi, and A.B. Whinston. Shill-proof fee (SPF) schedule: The sunscreen against seller self-collusion in online english auctions. Working Paper, 2004.

Evaluating Bidding Strategies for Simultaneous Auctions

Teddy Candale
University of Tulsa
teddy-candale@utulsa.edu

Sandip Sen
University of Tulsa
sandip@utulsa.edu

ABSTRACT

Bidding for multiple items or bundles on online auctions raises challenging problems. We assume that an agent has a valuation function that returns its valuation for an arbitrary bundle. In the real world all or most of the items of interest to an agent is not present in a single combinatorial auction. We study the problem of bidding for multiple items in a set of simultaneous auctions, each of which sell only a single unit of a particular item. Hence an agent has to bid in multiple auctions to obtain preferred item bundles. While an optimal bidding strategy is known when bidding in sequential auctions, only suboptimal strategies are available when bidding for items sold in auctions running simultaneously. To decide on an agent's bid for simultaneous auctions, we investigate a multi-dimensional bid improvement strategy, which is optimal given an infinite number of restarts, . We provide a comparison of this algorithm with existing ones, both in terms of utilities generated and computation time, along with a discussion of the strengths and weaknesses of these strategies.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Coherence and coordination, Multiagent systems, Intelligent agents*

General Terms

Algorithm, Performance, Experimentation

Keywords

bidding, simultaneous auctions

1. INTRODUCTION

Auction theory has received significant attention from researchers following the development of electronic auctions on the Internet. Researchers are interested both in designing auctions with desirable properties [5, 6] and designing automated agents to represent interests of human users [3, 4, 7].

In a multi-auction setting, multiple single-item auctions are run concurrently or sequentially. A potential bidder needs to estimate closing prices of such auctions to compute optimal bids. The bidder incurs the computational cost of estimating closing prices and calculating optimal bids given its valuation for items and the expected closing prices. The problem of computing optimal bids is complex. While an optimal bidding strategy is known for sequential auctions [3, 8], none is known for simultaneous auctions.

We have developed a multidimensional bid improvement strategy for bidding in simultaneous auctions. We use an optimization techniques to determine bids that maximize the expected utility function. The focus of this paper is a comparative evaluation of our proposed algorithm with existing ones in terms of time complexity and the quality of the solutions generated.

2. RELATED WORK

Bidding for bundles in simultaneous auctions has received a lot of attention partly due to the Trading Agent Competition (TAC)[1]. Stone *et al.* and Greenwald and Boyan have studied the possibility to bid in multi-auctions in the context of the TAC [4, 2, 8, 7], which resulted in the design of two top-scoring agents. Their approaches, however, were fundamentally different from the one we have developed [1]. Our multi-dimensional bid improvement algorithm is motivated by optimization techniques, where, given any bid vector, we generate the next bid vector by sequentially optimizing the bid for each item while assuming the bid for the other items are held constant. The approaches used by both Stone *et al.* and Greenwald involve assigning valuations to individual items based on expected marginal utilities of the items. The marginal utility of an item i is the extra-profit generated by the acquisition of i at price equal to zero. The idea behind the marginal utility is that one is willing to pay for an item up to the benefit of getting it. We provide a more detailed description of Stone *et al*'s and Greenwald and Boyan's algorithms in Section 4 and offer a comparison of the effectiveness of their algorithms and ours in Section 5.

3. SIMULTANEOUS AUCTION MODEL

We consider a bidder which plans to acquire items from the set $\mathcal{I} = \{1, \ldots, N\}$. A valuation function ϑ represents the bidder's preference by assigning a value the bidder is

[1]http://www.sics.se/tac/.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

willing to pay for each bundle. Item i is available in only one single-item single-unit auction a_i. An auction is modeled by the cumulative probability distribution F_i of the closing prices in the range $[\underline{p_i}, \overline{p_i}]$. We assume these distributions to be continuous, independent, and known to the bidder. When a_i closes, the bidder gets the item if it has placed a bid b_i greater than or equal to the closing price p_i ($b_i \geq p_i$) and the payment is equal to the closing price. All auctions are run in parallel and their closing times are not known by the bidder. $B = (b_1, \ldots, b_N)$ represents the bids placed simultaneously in all auctions by the bidder and $P = (p_1, \ldots, p_N)$ are the closing prices. We assume the bidder to be rational, i.e., it wants to maximize its expected utility from the purchases, if any. Hence it tries to find a bid vector B^* such that $B^* = \underset{B \in \mathcal{B}}{\operatorname{argmax}}\ \bar{\alpha}(B)$ where $\mathcal{B}$ is the bid space and $\bar{\alpha}(B)$ is the expected utility of bidding B.

4. BIDDING STRATEGIES

We review some prominent bundle bidding strategies before discussing our approach.

4.1 Marginal utility bidding ($\overline{MU}$)

We first present the optimal bidding strategy for bundles in sequential auctions[2, 8]. In the remainder of this paper, we will refer to this method as *expected marginal utility bidding* or $\overline{MU}$. When bidding for the ith item, the bidder places the bid $b_i = \bar{\mu}(i, I_h, I_r)$ in auction a_i. I_h contains items held by the bidder, I_r contains items to be auctioned, $\bar{\mu}(i, I_h, I_r)$ is the expected marginal utility of item i which can be viewed has the extra-profit due to the acquisition of i at zero-cost.

$\overline{MU}$ is optimal when bidding in sequential auctions as shown by Greenwald and Boyan in [2]. $\overline{MU}$ is suboptimal for simultaneous auctions. In particular, when items are substitutable, the bidder may acquire two items it may not desire to acquire together. Another downside of $\overline{MU}$ is its complexity. The calculation of the marginal utility is exponential since it requires the knowledge of the profit generated by each possible bundle.

To adopt $\overline{MU}$ for simultaneous auctions, a bidder has to compute all bids simultaneously. In that case, $\mathcal{I}_h = \emptyset$. The bidder bids for each item as if it is the first item to be auctioned. In other words, the bidder places $b_i = \bar{\mu}(i, \emptyset, \mathcal{I} \setminus \{i\})$ in auction a_i.

4.2 Expected Value Marginal Utility bidding (EVMU)

In this section, we present a variant of $\overline{MU}$ introduced by Greenwald and Boyan [2]. This variant tries to overcome the issue raised by leaving out of consideration some bidder preferences. To prevent the bidder from obtaining two "similar" items, a set of preferred items is precomputed. We refer to this set as the *acquisition set* $\mathcal{I}_{evmu}$. The acquisition set is the optimal set of items to be obtained assuming that items are auctioned at deterministic prices equal to the expected closing prices for these items. The bidder places a bid for all items in $\mathcal{I}_{evmu}$ and the bids are equal to the expected marginal utility of the items. Like $\overline{MU}$, the computation of marginal utility in EVMU is exponential.

4.3 Multi-Dimensional Bid Improvement (MDBI)

Now we present an approach to bidding in simultaneous auctions based on an incremental optimization technique [1]. Assume that the bidder has decided, by some means, to bid the vector B. We try to improve B replacing b_i by $\beta_i(B_{-i})$ where $\beta_i(B_{-i})$ is the best bid for item i holding other bids constant ($\beta_i(B_{-i}) = \underset{b_i \in [\underline{p_i}, \overline{p_i}]}{\operatorname{argmax}}\ \bar{\alpha}(b_i \vee B_{-i})$). We exclude the proof due to space limitations but observe that $\beta_i(B_{-i})$ is such that $\frac{\partial \bar{\alpha}}{\partial b_i}(\beta_i(B_{-i}) \vee B_{-i}) = 0$, i.e., $\beta_i(B_{-i})$ is the *marginal expected welfare* of bidding B_{-i}.

To use this approach the bidder has to choose an initial bid vector B_0 by some means, e.g., randomly. Let B_t be the bid after t iterations. B_{t+1} is obtained from B_t by sequentially improving bids for each of the N items in turn. While considering the improvement of the bid for an item, we keep the bids for other items constant in B_t. We call this $N-step$ perturbation an $N-sequential$ improvement and the algorithmic process a multi-dimensional bid improvement (MDBI) scheme. The process is stopped when $||B_{t+1} - B_t|| < \varepsilon$ where ε is a positive constant defined by the user and $||\cdot||$ is any vectorial norm.

To improve the bid for item i, we need to calculate β_i. The formula, however, requires exponential computation. It is possible to approximate β_i by price sampling. The following presents the pseudo-code to approximate β_i, where K is the number of price samples generated from the closing price distributions for all the items.

$\beta \leftarrow 0$
for $k = 1..K$ **do**
  $P \leftarrow \text{generatePriceSamples}(F_1, \ldots, F_N)$
  $\beta \leftarrow \beta + \left(\vartheta\left(\mathcal{I}_{ac}(\overline{p_i} \vee B_{-i}, P)\right) - \vartheta\left(\mathcal{I}_{ac}(\underline{p_i} \vee B_{-i}, P)\right)\right)$
end for
return $\frac{\beta}{K}$.

Three variants of MDBI are available: **Random Start Bid Improver (RSMDBI):** RSMDBI starts with a randomly chosen bid vector and does not use restarts. **Random Start Bid Improver With Restart (RSMDBIWRn):** RSMDBIWRn restarts the improvement process with random bid vectors $n-1$ times and outputs the bid vector with the highest expected utility. **Valuation Start Bid Improver (VSMDBI):** VSMDBI uses the initial bid $B_0 = (v_1, \ldots, v_N)$ where $v_i = \vartheta(\{i\})$ and does not use restarts.

The complexity of the approximation of $\beta_i(B_{-i})$ is linear given K, the number of samples. In MDBI, each N-sequential improvement requires N approximations of $\beta_i(B_{-i})$. The number of N-sequential improvements, $C(\varepsilon, N)$, correspond to the number of iterations of the improvement loop. Thus, the complexity of MDBI is $O(K\,N\,C(\varepsilon, N))$. Experimental results presented in Section 5 shows that the actual run-time complexity of MDBI is satisfactory.

5. EXPERIMENTAL RESULTS

Our experimentation goal is to compare the efficiency of different variants of the MDBI scheme with variants of MU bidding both in terms of the quality of the solution generated and time efficiency. To assess the absolute performances of these algorithms, we use an exhaustive search-based brute force algorithm (BF) which is asymptotically optimal

We ran our experiments in an environment containing four single-item single-unit auctions. All closing prices are drawn from selected discrete closing price distributions. A simulation consists of one bidder with the knowledge of all closing price distributions. A run of our experiment consists

Strategy	Score
RSMDBIWR10	1940.29
BF	1909.42
RSMDBIWR5	1907.25
VSMDBI	1859.65
RSMDBI	1856.85
EVMU	1708.18
$\overline{MU}$	1600.59

(a) SI

Strategy	Score
BF	7693.41
RSMDBIWR5	7693.32
RSMDBIWR10	7674.74
VSMDBI	7513.83
RSMDBI	7434.81
EVMU	7413.88
$\overline{MU}$	7202.51

(b) CI

Strategy	Score
RSMDBI	3748.42
VSMDBI	3731.57
RSMDBIWR10	3725.22
RSMDBIWR5	3724.76
$\overline{MU}$	3715.95
BF	3658.38
EVMU	3522.67

(c) NRI

Strategy	Score
RSMDBIWR10	6898.97
BF	6871.01
RSMDBIWR5	6828.67
RSMDBI	6239.84
EVMU	6224.37
VSMDBI	5946.33
$\overline{MU}$	5184.76

(d) RI

Table 1: Cumulative profits of different algorithms.

of seven simulations, one for each of the bidders RSMDBI, RSMDBIWRn with n=5 or 10, VSMDBI, BF, $\overline{MU}$, EVMU. For a run, bidders in each simulation share the same valuation function. We generate four kinds of valuation functions: (a) SI, where items are substitutable, (b) CI, where items are complementary, (c) NRI, where items are non-related, and (d) RI, where valuations for bundles are random.

5.1 Quality of solutions generated by algorithms

We present the cumulative average profit of the bidders using each algorithm in Table 1. The algorithms are ordered by the profits they generate. Algorithms with no statistical performance difference are grouped together. To obtain such groupings, we consider the average profits obtained by our algorithms as random variables and use the Wilcoxon test to identify the significance of the difference in their values.

We highlight the following observations from Table 1: (1) RSMDBI performs similar to or better than previously known algorithms. (2) VSMDBI performs similar to RSMDBI except for RI valuations where performances are worse. (3) With few random restarts our algorithm always performs similar to BF which is asymptotically optimal. (4) RSMDBIWRn, $n = 5, 10$ has better performances than RSMDBI except for non-related items where performances are equal. (5) Every algorithm is optimal when items are non-related except EVMU, (6) EVMU performs better than $\overline{MU}$ except when items are non-related when its performance is worse.

5.2 Time efficiency of MDBI variants

We discussed the complexity of MDBI in Section 4.3. The expression of this complexity contains an unknown function $C(\varepsilon, N)$. We wanted to find out the complexity of MDBI variants in practice. We run experiments using the RSMDBI and VSMDBI by varying the number of items, N. The time complexity of VSMDBI and RSMDBI appears to be linear in N. Since the complexity of MDBI is equal to $O(K\,N\,C(\varepsilon,\,N))$, $C(\varepsilon,\,N)$ is constant for VSMDBI given N. We also collected the average value of $C(\varepsilon,\,N)$ from our experiments and can highlight the following observations: (1) Except for non-related items, the number of N-sequential improvements ($C(\varepsilon,\,N)$) increases very slowly. The largest range is [2, 4], (2) The number of N-sequential improvements is always better for VSMDBI, (3) $C(\varepsilon,\,N) = 1$ for VSMDBI and $C(\varepsilon,\,N) = 2$ for RSMDBI when items are non-related.

6. CONCLUSION

We compared the performance of an optimization based bidding algorithm for obtaining bundles of items in simultaneous auctions, MDBI, that we have recently developed with existing algorithms from literature. We discuss variants of the basic iterative bid-improvement algorithm: an approach with finite number of restarts and one with no restarts but a carefully chosen starting bid vector. Performance similar to or better than existing algorithms in literature, $\overline{MU}$ and EVMU, is achieved without restarts when items are substitutable, complementary or non-related. We additionally observe that a small number of restarts can be used to approximate optimal expected utility.

Another strong point of the MDBI variants is their polynomial time complexities whereas existing algorithms have exponential complexity. While exact calculation of bid improvement needs combinatorial number of operations, we approximate such calculations in linear time by a constant number of price samples. We are currently working on formally characterizing the process and the properties of the MDBI algorithms.

Acknowledgments: This work has been supported in part by an NSF award IIS-0209208.

7. REFERENCES

[1] T. Candale. Hill-climbing approach to bidding for bundles in simultaneous auctions. Master's thesis, University of Tulsa, August 2005.

[2] A. Greenwald and J. Boyan. Bidding under uncertainty: theory and experiments. In *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 209–216. AUAI Press, 2004.

[3] A. Greenwald and J. Boyan. Bidding algorithms for simultaneous auctions: A case study. *Journal of Autonomous Agents and Multiagent Systems*, 10(1):67–89, 2005.

[4] A. Greenwald and P. Stone. Autonomous bidding agents in the trading agent competition. *IEEE Internet Computing*, 5(2):52–60, 2001.

[5] T. Sandholm. emediator: a next generation electronic commerce server. In *AGENTS '00: Proceedings of the fourth international conference on Autonomous agents*, pages 341–348. ACM Press, 2000.

[6] S. Sen, T. Candale, and S. Basak. Profit sharing auction. In *Proceedings of the Twentieth International Conference on Artificial Intelligence (AAAI-2005)*, July 2005.

[7] P. Stone and A. Greenwald. The first international trading agent competition: Autonomous bidding agents. *Journal of Electronic Commerce Research*, 2003.

[8] P. Stone, R. E. Schapire, M. L. Littman, J. A. Csirik, and D. McAllester. Decision-theoretic bidding based on learned density models in simultaneous, interacting auctions. *Journal of Artificial Intelligence Research*, 19:209–242, 2003.

Trust and reputation

Certified Reputation: How an Agent Can Trust a Stranger

Trung Dong Huynh Nicholas R. Jennings Nigel R. Shadbolt

Intelligence, Agents, Multimedia Group, School of Electronics and Computer Science,
University of Southampton , Southampton SO17 1BJ, UK.
{tdh02r,nrj,nrs}@ecs.soton.ac.uk

ABSTRACT

Current computational trust models are usually built either on an agent's direct experience of an interaction partner (interaction trust) or reports provided by third parties about their experiences with a partner (witness reputation). However, both of these approaches have their limitations. Models using direct experience often result in poor performance until an agent has had a sufficient number of interactions to build up a reliable picture of a particular partner and witness reports rely on self-interested agents being willing to freely share their experience. To this end, this paper presents *Certified Reputation* (CR), a novel model of trust that can overcome these limitations. Specifically, CR works by allowing agents to actively provide third-party references about their previous performance as a means of building up the trust in them of their potential interaction partners. By so doing, trust relationships can quickly be established with very little cost to the involved parties. Here we empirically evaluate CR and show that it helps agents pick better interaction partners more quickly than models that do not incorporate this form of trust.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Design, Reliability, Experimentation

Keywords

Trust, Reputation, Multi-Agent Systems

1. INTRODUCTION

A wide variety of networked computer systems (such as the Grid, the Semantic Web, and peer-to-peer systems) can be viewed as multi-agent systems (MAS) in which the individual components act in an autonomous and flexible manner in order to achieve their objectives. An important class of these systems are those that are *open*; here defined as systems in which agents can freely join and leave at any time and where the agents are owned by various stakeholders with different aims and objectives. From these two features, it can be assumed that in an open MAS: (1) the agents are likely to be self-interested and may not always complete tasks that are requested of them; (2) no agent can know everything about its environment; and (3) no central authority can control all the agents. Given such uncertainties, trust is central to effective interactions between the agents [7]. Indeed, this recognition accounts for the large number of computational models of trust (here defined as the subjective probability with which an agent a assesses that another agent b will perform a particular action, both before a can monitor such action and in a context in which it affects its own action [2]) that help agents to determine the most reliable interaction partner (see Section 4 for more details).

Although there are many differences in the way these models are implemented, the majority of them are built on an agent's direct experience of an interaction partner (interaction trust) or reports provided by third parties about their experiences with a partner (witness reputation). However, both these approaches have their shortcomings. First, when an agent first enters an environment, it has no history of interactions (with the other agents in that environment). In such circumstances, if its trust model is based solely on direct experience, it would need to explore the environment by interacting with other agents to learn about their performance. However, in so doing, the agent also inevitably risks making losses if it encounters unreliable partners. Moreover, because it can learn about only one agent per interaction, trust models using direct experience typically require a long time to be able to achieve stable performance. Second, models based on witness reports usually implicitly assume the altruism of agents in sharing their experiences. Now, this cannot be guaranteed in all cases because self-interested agents are unlikely to be willing to sacrifice their resources in order to provide witness reports. Moreover, in the distributed and open environments we consider, the required witnesses for any given agent can be difficult to locate. This is usually addressed by using some form of centralised mechanism to collect all the witness reports [13] or implementing a distributed search process to look for witnesses in an agent's social network [12]. However, the former is not compatible with an open MAS since agents representing different owners may well question the trustworthiness of a central authority and the latter may well involve high costs of time

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

and resources to locate witnesses.

Against this background, in this paper we present a novel type of trust called *certified reputation* (CR) whose mechanism overcomes the limitations of current trust models described above. Hence, CR can be used complementarily to other sources of trust information (e.g. direct experience, witness reports) to build a versatile composite trust model (which benefits from all the advantages of its components). In more detail, the CR of an agent is the reputation that is derived from third-party references about its previous performance. Agents that adopt the CR mechanism will actively collect and present such references in order to seek the trust of their potential partners. This, in turn, moves the burden of obtaining and maintaining trust information from the trust evaluator to the agent being evaluated (who is incentivised to do so)[1]. Hence, trust information (references) about a particular agent becomes available to those who want to interact with it. Moreover, agents using CR have neither to interact first with the agent being evaluated (as they do with direct experience), nor to locate its witnesses (as they do with witness reports). The potential downside, however, is that third-party references might not be reliable (since referees can collude with particular agents by providing falsely inflated references for them and negative ones for the others). Now, this is an inherent problem of using any type of third-party information and so it is important to be able to evaluate the credibility of a referee and to use this to weigh its information. To this end, we also develop such a model that records the history of a referee's performance in terms of providing accurate references (by comparing an agent's actual observations with the references it receives). Such histories of referees are then used to derive their credibility.

In so doing, we advance the state of the art in the following ways. First, we develop a new process of obtaining information for trust evaluation. This process addresses the inherent shortcomings of interaction trust (the lack of direct experience) and witness reputation (the difficulty in finding witness reports). Second, as CR allows agents to be able to evaluate trust themselves, without relying on a central mechanism, it is compatible with a wide range of open and distributed environments. Third, from empirical evaluation, it is shown that using CR improves an agent's utility gain by helping it to quickly pick good interaction partners. Finally, it is shown that our model is robust against various types of collusion.

The rest of the paper is organised as follows. Section 2 presents our model of certified reputation. This model is then empirically evaluated in Section 3. Section 4 presents related work and Section 5 concludes.

2. CERTIFIED REPUTATION

A trust model is typically used by an agent, say agent a, to evaluate the trustworthiness of another agent, say agent b, when it considers establishing an interaction with b. In this case, we call a the *evaluator* and b the *target*. Now, most trust evaluation is based on past experience of the target agent's performance. Such experience is here recorded in the form of *ratings* which are tuples of the following form: $r = (a, b, i, c, v)$, where a and b are the agents that participated in interaction i, and v is the rating a gave b for the term c. For instance, the quality of a news provider can be rated in terms of topicality, quality, and honesty. Here, the range of v is $[-1, +1]$, where -1, $+1$, and 0 means absolutely negative, absolutely positive, and neutral respectively. Each agent has a local *rating store* to collect the ratings it makes and the ones that it receives from others. Since ratings are context-dependent (described in the rating terms), each trust evaluation is also about a specific term. Here, we use $\mathcal{T}(a, b, c)$ to denote the trustworthiness of b in terms of c that is evaluated by a. In order to make such a trust evaluation, a needs to obtain a set of relevant ratings. We call this set $\mathcal{R}(a, b, c)$.

Now, having defined the basic notions, we turn to the CR model itself. The mechanism of CR is given in Section 2.1, which also describes how CR is calculated. Then Section 2.2 shows how the credibility of referees is evaluated.

2.1 The Mechanism of Certified Reputation

Certified reputation of a target agent b consists of a number of certified references[2] about its behaviour on particular tasks that are provided by third-party agents. Such information is obtained and stored by the target agent itself and made available to any other agent that wishes to evaluate its trustworthiness for further interactions (somewhat like a reference when a person is applying for a job). The agents giving references are called the *referees*. Here, references are in the form of ratings given by b's partners about its performance in (past) interactions between them. These ratings allow b to prove its (achievable) performance as viewed by its previous interaction partners and then to gain the trust of its potential partners. However, since b can choose which ratings to put forward, a rational agent will only present its best ones. Therefore, it should be assumed that CR information probably overestimates an agent's expected behaviour. Thus, although it cannot guarantee b's minimal performance in future interactions, the CR information does reveal a partial perspective on agent b's capabilities (which is certainly useful for trust evaluation in the absence of other sources of information).

Though CR may have lower predictive power than the other types of trust/reputation (where all bad and good ratings can be collected), it is useful because of its wide applicability. With the cooperation of its partners, agent b can obtain their references from just a small number of interactions[3]. From our evaluation, for instance, in a society where 100 agents provide a service to 500 others, agents using direct experience to evaluate trust require more than 100 interactions to achieve a reasonable level of utility gain,

[1]It can reasonably be assumed that agents want to have more interactions because of their potential utility gain. Hence, agents have incentives to provide trust information about themselves so that they can gain the trust of other agents which, in turn, will facilitate more interactions.

[2]It is assumed that some form of security mechanism (such as a public-key infrastructure) is employed to ensure that the provided references cannot be tampered with. For instance, all references could be accompanied by digital signatures from the issuers using their private keys [14]. By so doing, any change to a reference will be easily detected. Digital signatures are also a means to verify the references' origins.

[3]In many scenarios, such as those in the Internet, established service providers (e.g. news service, online merchants) usually have high volumes of interactions (at any time). Therefore, if they adopt the CR process outlined here, we can reasonably expect that such providers will have an abundance of performance ratings readily available.

which is still less than that achieved by agents using CR after 5 interactions (see Section 3 for more detail). In addition to its high availability, since references are stored by the target agent and provided directly to the evaluator, CR has a very low running cost (i.e. time, communication, processing cost) compared to sources like witness reputation.

In more detail, the process of CR is as follows:

- After every transaction, agent b asks its partners to provide their certified ratings about its performance from which it can choose the best ratings to store in its (local) rating store.
- When agent a contacts b to express its interest in using b's service, it asks b to provide references about its past performance with respect to an interested term c.
- Agent a receives the set of certified ratings of b from b, which we call $\mathcal{R}_{\mathsf{C}}(a,b,c)$ (C to denote this set is obtained via the CR mechanism), and calculates the CR of b based on this set.

In this process, since agent b relies on its interaction partner's cooperation to get references, agents may refuse to give out their ratings (as in the case of witness reputation). However, this is a much smaller problem than that in witness reputation because this information is requested far less frequently (each referee is requested to give its rating only once). Moreover, giving such information could be made a standard part of any agreement for task allocation and so agents could be forced to give it. The most notable point in this process is that when agent a makes the trust evaluation, it only involves agents a and b. Since the certified ratings about b are stored by b itself, they are immediately available to a as in the case when a uses its own experience.

Having obtained the references of b, a can estimate b's future behaviour, or more specifically, the expected rating value b is likely to receive in a future interaction. A common way to estimate that value is to calculate it as the arithmetic mean of all the rating values in the set. However, these ratings are usually not equally relevant when estimating the expected rating value. For example, some ratings may be older than others and, thus, may be out-of-date; and some may come from a more reliable source that suggests a higher level of credibility compared to others. Therefore, like many contemporary models [8, 10, 12], we use a *rating weight function* $\omega_{\mathsf{C}}(r_i)$ ($\omega_{\mathsf{C}}(r_i) \geq 0$) which calculates the relevance of each given certified rating r_i. Then, instead of considering all ratings equally, the trust value is calculated as the weighted mean of all the ratings available[4]:

$$\mathcal{T}_{\mathsf{C}}(a,b,c) = \frac{\sum_{r_i \in \mathcal{R}_{\mathsf{C}}(a,b,c)} \omega_{\mathsf{C}}(r_i) \cdot v_i}{\sum_{r_i \in \mathcal{R}_{\mathsf{C}}(a,b,c)} \omega_{\mathsf{C}}(r_i)} \quad (1)$$

where $\mathcal{T}_{\mathsf{C}}(a,b,c)$ is the CR value of b that agent a calculates with respect to term c, which is calculated from the rating set $\mathcal{R}_{\mathsf{C}}(a,b,c)$, and v_i is the value of the rating r_i. In short, the CR value is calculated as the sum of all the available ratings weighted by the rating relevance and normalised to the range of $[-1,1]$ (by dividing the sum by the sum of all the weights).

Now, we need to determine the relevancy of a given rating ($\omega_{\mathsf{C}}(r_i)$). Since an agent's environment may change rapidly, resulting in out-of-date ratings, we use the recency of ratings as one measure of their relevance. Specifically, the recency relevance of a rating r_i is calculated by an exponential decay function based on its recency [5]:

$$\omega_{\mathsf{Re}}(r_i) = e^{-\frac{\Delta t(r_i)}{\lambda}} \quad (2)$$

where $\omega_{\mathsf{Re}}(r_i)$ is the relevance value for the rating r_i in terms of recency, $\Delta t(r_i)$ is the time difference between the current time and the time when the rating r_i is recorded, and λ is the recency factor which is used to scale $\Delta t(r_i)$ according to a particular application's time unit.

As certified ratings are digitally signed, their authenticity can be verified and their content cannot be tampered with. However, since there is no guarantee about the honesty of agents in an open MAS, a referee can still collude with the target agent and provide falsely inflated references about its performance. Moreover, even if the referee is honest in providing references, its references can be inaccurate (either because of its incapability of making accurate ratings or because it has a different view to that of the evaluator). In either case, these inaccurate references will result in inaccurate CR values. Therefore, we need measures to prevent or to minimise the adverse effects of such references. To this end, we use the credibility of referees as another measure of rating relevance. In more detail, for a reference r_i from referee w, w's credibility value is calculated by a (called $\mathcal{T}_{\mathsf{RCr}}(a,w) \in [-1,1]$, RCr denotes referee credibility). Then, the relevance of r_i in terms of referee credibility (denoted by $\omega_{\mathsf{RCr}}(r_i)$) is defined as follows:

$$\omega_{\mathsf{RCr}}(r_i) = \begin{cases} 0 & \text{if } \mathcal{T}_{\mathsf{RCr}}(a,w) \leq 0 \\ \mathcal{T}_{\mathsf{RCr}}(a,w) & \text{otherwise} \end{cases} \quad (3)$$

Both the above measures of rating relevancy are then taken into account in weighing a rating:

$$\omega_{\mathsf{C}}(r_i) = \omega_{\mathsf{RCr}}(r_i) \cdot \omega_{\mathsf{Re}}(r_i) \quad (4)$$

It should be noted that, from Equations 3 and 4, references whose referees have negative credibility are discarded (by setting $\omega_{\mathsf{C}}(r_i) = 0$). Only references whose referees have positive credibility are taken into account when calculating CR. In such cases, they are weighted both by their referees' credibility and by their recency. In so doing, ratings from the more credible referees make a bigger impact on the CR value than those from the less credible ones. In cases where all the certified ratings collected are disregarded, due to negative credibility of their providers, no trust value will be produced. In such circumstances, agent a can give b a default low trust value (e.g. -0.5) because b fails to provide reliable references. Otherwise, agent a can re-request b for more reliable references.

2.2 Referee Credibility

The credibility of a referee in reporting its ratings about another agent can be derived from a number of sources. These include knowledge about: the relationships between the referee and the rated agent (e.g. cooperating partners may exaggerate each other's performance, competing agents may underrate their opponents, no relationship may imply impartial ratings); the reputation of the referee for being honest and expert in the field in which it is doing the rating (e.g. a reputable and independent financial consultant should provide fair ratings about the services of various banks); the relationships between the referee and the querying agent (e.g.

[4]We choose the weighted mean method here because it allows us to take the relevance of each rating into account. Other aggregation methods could equally well be used if desired.

agents with the same owner should provide honest reports to one another); and so on. Unfortunately, however, such types of knowledge are very application specific and may not be readily available in many cases. Therefore, although they could certainly be used to enhance the precision of a referee credibility measure, they are not suitable as a generic basis (although they could complement a generic measure in particular contexts). Therefore, here, we base our solution on a modified version of the witness credibility model we devised for evaluating witness reputation in [4] to assess a referee's credibility. We choose this model because it does not require such domain-specific information about relationships and because providing witness reports and references are of the same broad nature. Specifically, in this model, we view providing references as a service an agent provides. Thus its performance (i.e. trustworthiness and reliability) can be evaluated and predicted by a trust model. By so doing, the credibility model can benefit from a trust model's ability of learning and predicting an agent's behaviour (in this case, the behaviour of providing accurate reports) without having to implement its own method.

In more detail, after having an interaction with agent b, agent a records its rating about b's performance, denoted by r_a ($r_a = (a, b, i_a, c, v_a)$). Now, if agent a received a reference (i.e. a certified rating) from agent w, it then rates the credibility of w by comparing the actual performance of b (i.e. v_a) with w's rating about b. The smaller the difference between the two rating values, the higher agent w is rated in terms of providing accurate references (mutatis mutandis for bigger differences). For each certified rating that a received from w in evaluating the CR of b (denoted by $r_k = (w, b, i_k, c, v_k)$), the credibility rating value v_w for agent w is given in the following formula:

$$v_w = \begin{cases} 1 - |v_k - v_a| & \text{if } |v_k - v_a| < \iota \\ -1 & \text{if } |v_k - v_a| \geq \iota \end{cases} \quad (5)$$

where ι is called the *inaccuracy tolerance threshold* ($0 \leq \iota \leq 2$, 2 is the maximal difference since $v_k, v_a \in [-1, 1]$). Thus if the difference between a certified rating value and the actual performance is higher than ι, the referee is considered to be inaccurate or lying, and, therefore, receives a negative rating of -1 for its credibility. On the other hand, if the difference is within the tolerance threshold, it can be viewed as resulting from a subjective viewpoint and is deemed acceptable. In this case, the credibility rating value v_w is set to be inversely varied to the difference (e.g. higher difference, lower credibility). Since $0 \leq |v_n - v_a| \leq 2$, v_w is also in the range $[-1, 1]$ regardless of ι. The rating about w's credibility — $r_w = (a, w, i_w, \text{term}_{\mathsf{RCr}}, v_w)$ — is then recorded by a, where $\text{term}_{\mathsf{RCr}}$ is the rating term for performance in providing references and i_w is the interaction of agent w providing agent a the certified rating r_k about agent b.

Having recorded ratings about w's performance on providing references, a can evaluate w's credibility based on those ratings. As mentioned above, a can use its own trust model for so doing. Specifically, here we calculate a's trust on w's capability of providing accurate references similarly as per CR (Equation 1), except that we use only ratings retrieved from a's ratings store (instead of obtaining them via the CR mechanism) and weigh the ratings only by their recency[5] (Equation 2). This type of trust is here called interaction trust (IT) since it is based solely on a's direct experience from its interactions. The IT value of a on b in terms of c is denoted by $\mathcal{T}_{\mathsf{I}}(a, b, c)$ and is calculated as follows:

$$\mathcal{T}_{\mathsf{I}}(a, b, c) = \frac{\sum_{r_i \in \mathcal{R}_{\mathsf{I}}(a,b,c)} \omega_{\mathsf{I}}(r_i) \cdot v_i}{\sum_{r_i \in \mathcal{R}_{\mathsf{I}}(a,b,c)} \omega_{\mathsf{I}}(r_i)} \quad (6)$$

where $\omega_{\mathsf{I}}(r_i)$ is the rating weight of r_i and $\omega_{\mathsf{I}}(r_i) = \omega_{\mathsf{Re}}(r_i)$. Then, in order to determine the referee credibility of w ($\mathcal{T}_{\mathsf{RCr}}(a, w)$), we need to calculate $\mathcal{T}_{\mathsf{I}}(a, b, \text{term}_{\mathsf{RCr}})$. This is calculated from the set of $\text{term}_{\mathsf{RCr}}$ ratings — $\mathcal{R}_{\mathsf{I}}(a, b, \text{term}_{\mathsf{RCr}})$. However, if no such rating has been recorded, and, thus, the IT value is not available, we assign w the default referee credibility trust value, denoted by $\mathcal{T}_{\mathsf{DRCr}}$:

$$\mathcal{T}_{\mathsf{RCr}}(a, w) = \begin{cases} \mathcal{T}_{\mathsf{I}}(a, w, \text{term}_{\mathsf{RCr}}) & \text{if } \mathcal{R}_{\mathsf{I}}(a, w, \text{term}_{\mathsf{RCr}}) \neq \emptyset \\ \mathcal{T}_{\mathsf{DRCr}} & \text{otherwise} \end{cases} \quad (7)$$

It should be noted here that, at first, every referee receives the default credibility value since it has not provided references to agent a before. Hence, end users can set the value of $\mathcal{T}_{\mathsf{DRCr}}$ to reflect their policy towards newly encountered referees. For example, $\mathcal{T}_{\mathsf{DRCr}}$ can be set to 0 so that newly encountered referees are disregarded until they prove to be credible (by providing ratings in the acceptable accuracy threshold) or it can be set to 1 so that all referees are considered to be accurate and honest until proven otherwise.

3. EMPIRICAL EVALUATION

In order to empirical evaluate our CR model, we use the testbed designed in [5] with a few changes. In more detail, in order to verify our intuitions about CR, it will be evaluated in ideal (honest) environments where agents do not collude with each other (Section 3.3). It will then be evaluated in (biased) environments where colluding agents give high ratings to each other (Section 3.4) to determine its robustness for open multi-agent contexts. Before doing this, however, a brief description of the testbed is given in Section 3.1 (see [5] for a complete description and for justifications of the choices we made in it) and Section 3.2 presents the methodology and experimental settings for our experiments.

3.1 The Testbed

The testbed is a multi-agent system consisting of agents providing services (called *providers*) and agents using those services (called *consumers*). Without loss of generality, it is assumed that there is only one type of service in the testbed. Hence, all the provider agents offer the same service. However, their performance (i.e. the quality of the service) differs. The agents are situated randomly on a spherical world whose radius is 1.0. Each agent has a *radius of operation* r_o that models its capability in interacting with others (e.g. the available bandwidth or the agent's infrastructure). In addition, each consumer agent has a maximum number of friend providers (N_{FP}) that it may collude with when providing references about their performance. Such providers are selected randomly from a consumer's nearby providers when it enters the testbed.

Simulations are run in the testbed in rounds (of agent interactions), and the round number is used as the time unit. In each round, if a consumer agent needs to use the service it can contact the environment to locate nearby provider agents (in terms of the distance between the agents on the

[5]Since a makes the ratings itself, weighing them in terms of referee credibility is irrelevant.

spherical world). The consumer agent will then select one provider from the list to use its service. The selection process relies on the agent's trust model to decide which provider is likely to be the most reliable. Consumer agents without a trust model randomly select a provider from the list. The consumer agent then uses the service of the selected provider and gains some utility from the interaction (called UG). The value of UG is in $[-10, 10]$ and depends on the level of performance of the provider in that interaction. A provider agent can serve many consumers at a time.

After an interaction, the consumer agent will rate the service of the provider based on the level of performance it received and provide the rating to the provider to be potentially used as a reference. A provider will select $\mathsf{N_R}$ best references from those that it receives to store and to present when requested. Since a provider knows very old references are unlikely to be valued highly by its consumers, when selecting references to store, it takes into account both a reference's rating value (i.e. saying how well the provider performed) and the time when that reference is made. In particular, when comparing two references r_1 and r_2, the value of r_1 is biased by an amount of $\frac{t_1 - t_2}{\mathsf{TIMESCALE}}$, where t_i is the time that reference r_i is given, and TIMESCALE is selected to be 20.0 given the time unit used in the testbed[6].

In addition, since a referee may make biased references, we model this phenomenon by introducing five types of referees. Agents in the Hon group always give out their actual ratings as references. *Exaggerating referees* in groups Exag1 and Exag2, however, give falsely higher ratings than those they actually recorded for their friend providers[7] (and their actual ratings for the others). In addition to giving falsely inflated references for their friends, *extreme referees* in Extr1 and Extr2 also deliberately underrate the other providers. The difference between an actual rating value and its inaccurate one in Exag1 and Extr1 is randomly set in the range $[0.3, 1.0]$ (i.e. representing marginally inaccurate referees) and the respective range of Exag2 and Extr2 is $[1.0, 2.0]$ (i.e. representing extremely inaccurate referees).

In our testbed, the only difference in each interaction situation is the performance of the provider agents. Here, we consider four types of providers: good, ordinary, bad, and intermittent. Each of them, except the last, has a mean level of performance, denoted by μ_P. Its actual performance follows a normal distribution around this mean. The values of μ_P and the associated standard deviation (σ_P) of these types of providers are given in Table 1. Intermittent providers, on the other hand, yield unpredictable (random) performance levels in the range [PL_BAD, PL_GOOD]. In addition, the service quality of a provider is also degraded linearly in proportion to the distance between it and the consumer to reflect the greater uncertainties associated with service delivery (e.g. lower service quality resulting from increased delays or losses in information exchanges between two agents when they are far away from each other). Hence, from the same provider, each consumer may receive a different level of service quality depending on its location. This means honest ratings about that provider's performance by its consumers can be different; reflecting the phenomenon that every agent has its own context making its own view different.

Profile	Range of μ_P	σ_P
Good	[PL_GOOD, PL_PERFECT]	1.0
Ordinary	[PL_OK, PL_GOOD]	2.0
Bad	[PL_WORST, PL_OK]	2.0

Performance level	Utility gained
PL_PERFECT	10
PL_GOOD	5
PL_OK	0
PL_BAD	−5
PL_WORST	−10

Table 1: Profiles of provider agents.

3.2 Experimental Methodology

In each experiment, the testbed is populated with provider and consumer agents. Each consumer is equipped with a particular trust model, which helps it select a provider when it needs to use a service. Since the only difference among consumer agents is the trust models that they use, the utility gained (UG) by each agent reflects the performance of its trust model in selecting reliable providers for interactions. Hence, the testbed records the UG of each interaction along with the trust model used. In order to obtain an accurate result for performance comparisons between trust models, each one will be employed by a large number of consumer agents ($\mathsf{N_C}$). In addition, the average UG of agents employing the same trust models (called consumer groups) are compared with each other's using the two-sample t-test [1] (for means comparison) with a confidence level of 95%. The result of an experiment is then presented in a graph with two y-axes (see Fig. 1 for an example); the first plots the UG means of consumer groups in each interaction and the second plots the corresponding performance rankings obtained from the t-test (prefixed by "R.", where the group of rank 2 outperforms that of rank 1). The experimental variables are presented in Table 2 and these will be used in all experiments unless otherwise specified.

Simulation variable	Symbol	Value
Number of simulation rounds	N	500
Total number of provider agents:	$\mathsf{N_P}$	100
+ Good providers	$\mathsf{N_{PG}}$	10
+ Ordinary providers	$\mathsf{N_{PO}}$	40
+ Bad providers	$\mathsf{N_{PB}}$	45
+ Intermittent providers	$\mathsf{N_{PI}}$	5
Number of consumers in each group	$\mathsf{N_C}$	500
Max. number of friend providers	$\mathsf{N_{FP}}$	4

Table 2: Experimental variables.

Now, in each experiment, we include several groups of consumers in order to compare their performance. These include one group employing CR, one employing the SPORAS model[8] (see Section 4 for details), one employing IT[9],

[6]This is obviously just one way of comparing two references but it is a way that is effective in our system. If we wanted to make rating recency less relevant then we could make TIMESCALE smaller, or if we wanted to make rating recency more relevant then we could make it bigger.

[7]This is motivated by examples where referees provided exaggerated reports about their friends.

[8]SPORAS is chosen as the control benchmark because it is a successful, independently developed trust model which several other researchers have used for benchmarking.

[9]IT is defined in Equation 6. It is calculated from a con-

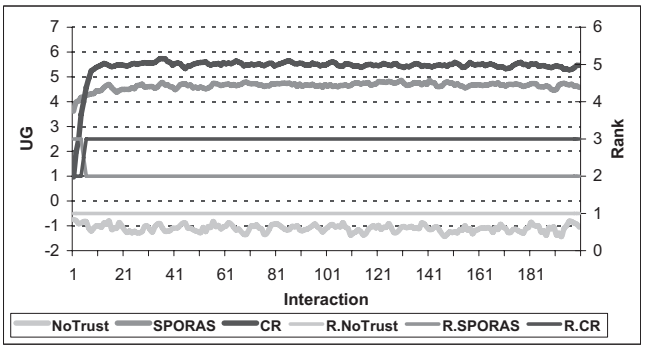


Figure 1: Performance of CR vs SPORAS and NoTrust.

and one consisting of agents with no trust model. We name these groups CR, SPORAS, IT and NoTrust. A summary of the parameters of CR is provided in Table 3. The recency factor λ is selected such that a 5-time-unit-old rating will have a recency weight of 0.5 (to suit the time unit used in the testbed). The default referee credibility $\mathcal{T}_{\mathsf{DRCr}}$ is set to 0.5 so that all ratings from newly encountered referees will be taken into account in calculating CR, but their weights are smaller than those of any proven accurate referee (which are typically greater than $\iota = 0.5$, see Equation 7) and larger than that of a proven inaccurate one (which is typically negative). The value of ι is handpicked based on the actual variability of honest rating values in the testbed (which never exceeds 0.5).

Parameters	Symbol	Value
Recency factor	λ	$-\frac{5}{\ln(0.5)}$
Number of stored references	$\mathsf{N_R}$	10
Referee credibility parameters:		
+ Default referee credibility	$\mathcal{T}_{\mathsf{DRCr}}$	0.5
+ Inaccuracy tolerance threshold	ι	0.5

Table 3: CR's parameters.

3.3 Honest Environments

Having defined the testbed and the evaluation methodology, we now turn to the experiments themselves. The first thing to test is whether CR helps consumer agents select profitable providers (i.e. those yielding positive UG) from the population and, by so doing, helps them gain better utility than without using CR. Hence, in this experiment we evaluate the performance of CR against that of NoTrust. SPORAS is also included in this experiment to compare the performance of CR with that of an independent benchmark that does not use CR. Here, Fig. 1 shows that the NoTrust group, selecting providers randomly without any trust evaluation, performs consistently the lowest (as we would expect). On the other hand, both SPORAS and CR prove to be beneficial to consumer agents, helping them obtain significantly high UG. This shows that the tested trust models can learn about the provider population and allow their agents to select profitable providers for interactions. In particular, SPORAS, being a centralised service, is able to gather ratings about all interactions in the system. This allows agents using it to achieve high performance right from the first interactions.

sumer's ratings about a provider's performance which that consumer collects in its rating store after every interaction.

In contrast, since each provider only shows a small number of ratings to agents using CR, they spend the first few interactions learning about their environment. Hence their initially lower performance than that of SPORAS. However, agents using CR quickly catch up with those in SPORAS (after 5 interactions) and they then maintain a higher stable performance thereafter (after the first 5 interactions, the average UG per interaction of SPORAS and CR are 4.65 and 5.48, respectively). The t-test also confirms that this difference is statistically significant as the ranks of SPORAS and CR switch after the first 5 interactions. Here, it should be noted that consumers may have different views on the same provider's performance because of the different distances between them and the provider (see Section 3.1). Therefore, although all the agents are honest, their ratings about that provider can be different due to their particular experience. Now, SPORAS takes all ratings equally and, thus, such differences are merely noise to it. This is the main reason that limits its performance. In contrast, the referee credibility model of CR compares each referee's ratings with an agent's own ratings and gives higher weights to those referees that have similar views to it. By so doing, the relevancy of each rating is determined and taken into account in calculating CR, making it more accurate than SPORAS.

Having shown that CR can outperform a centralised trust model based on witness reports, we now test CR against IT, which uses an agent's direct experience. Here, there are two groups of consumer agents, IT and CR, using IT and CR, respectively, as their trust models. The t-test results in Fig. 2 shows that CR outperforms IT in all interactions. The chart also shows that while it only takes agents using CR 5 interactions to achieve a stable level of performance, agents using IT need more than 100 interactions to obtain a reasonably stable level of performance. This is clearly because agents in IT do not have adequate ratings to quickly learn about their environment (i.e. the performance of providers). Moreover, even the maximum UG of IT (4.85) is lower than the average UG of CR (around 6.0). The reason is that agents using IT can miss identifying the best providers because ratings about them are less available than in the case of CR and because agents in our testbed will stick to choosing the highest trusted provider after a period of learning about the environment[10] (resulting in locally sub-optimal provider selections). Meanwhile, agents using CR always receive the most recent and the best references that are actively presented by providers. Moreover, the high availability of these references helps CR greatly increase its learning speed and, thus, avoid the problems associated with IT.

In summary, we have shown that in honest environments, CR provides a robust trust measure that is also highly available. However, we now go onto consider the more realistic situations in which collusion can occur.

3.4 Biased Environments

In this section, we evaluate CR in a wider range of environments. In particular, the consumers in each experiment consist of honest referees and those from one of the four colluding referee groups (Exag1, Exag2, Extr1, and Extr2 — see Section 3.1). For example, Fig. 3 presents the result from the experiment where the consumers consist of 20% honest referees and 80% colluding referees from the Extr2 group.

[10]Agents in our testbed use a Boltzmann exploration strategy to explore (i.e. try the service of) unknown providers.

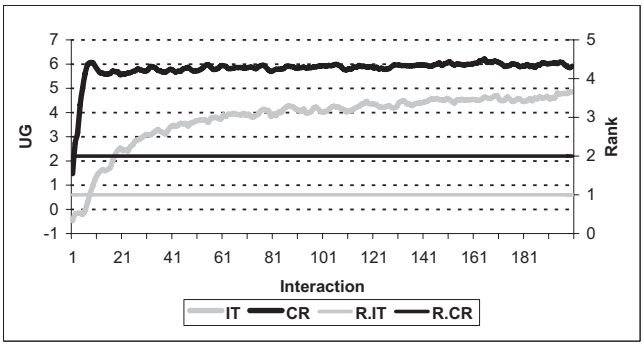


Figure 2: Performance of CR vs IT.

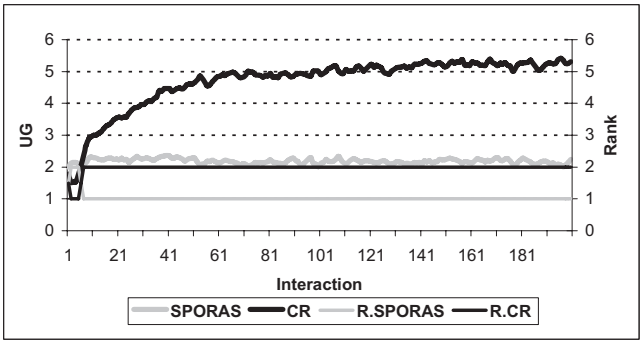


Figure 3: 80% Extr2 referees.

Since NoTrust performs consistently poorly in all the experiments, its results are omitted from our charts for the sake of simplicity. Moreover, because of space constraints, we cannot provide a detailed result of every experiment as in Fig. 3. Instead, we plot the average UG per interaction of SPORAS and CR in each experiment on the summary charts in Figs. 4 and 5. In these charts, the plots are named as GroupName.RefereeType. For example, SPORAS.Extr1 is the plot for the average UG of agents in the SPORAS group when the colluding referees in the testbed are of type Extr1.

The first thing these charts show is that collusion adversely affects the performance of trust models (as we would expect). For instance, Fig. 3 shows that it takes longer for CR to reach a stable level of performance (i.e. to learn about the colluding agents) and this performance is also lower than that in an honest environment. In such circumstances, SPORAS also yields a low UG and, without a credibility model, it cannot improve its performance over time. We can also see that the average performance of both SPORAS and CR generally decreases when the number of colluding agents increases (Figs. 4 and 5). However, CR always outperforms SPORAS except in the case when 100% of the consumers are Extr2 agents. In this particular experiment, since all consumers are identified by CR as lying (because all of them provide highly distorted references for all the providers), CR stops using their references, thus, depressing performance. However, this particular case (i.e. 100% extreme collusion) is highly unlikely to happen in practice (and if it did one might just retract trust from the population altogether). In the remaining experiments, CR can easily detect referees providing highly distorted references and maintain a generally high performance (see plots CR.Exag2 and CR.Extr2). CR is less effective in filtering out the colluding agents in

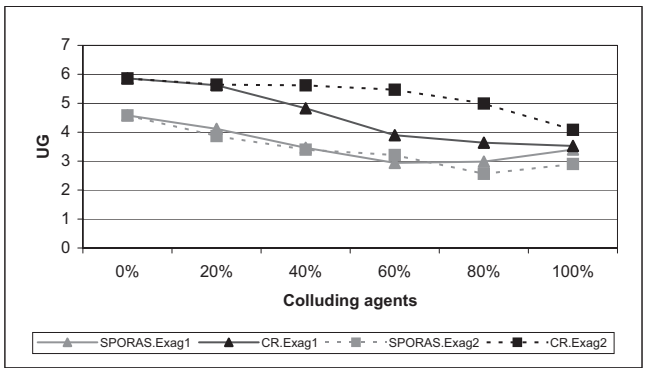


Figure 4: Performance with exaggerating referees.

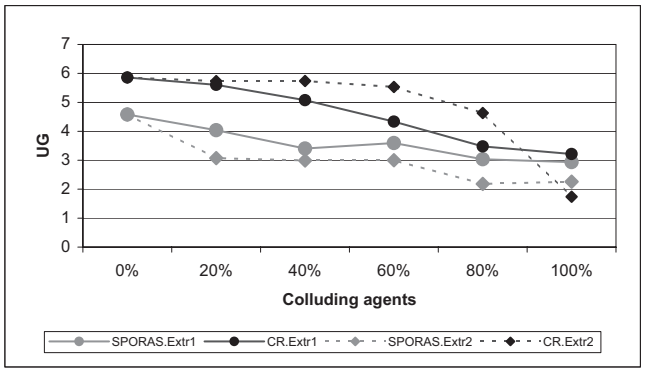


Figure 5: Performance with extreme referees.

Exag1 and Extr1 since their colluded references are less distorted than in the case of Exag2 and Extr2 (i.e. more difficult to detect lying). This suggests that the inaccuracy tolerance threshold ι should be carefully selected to reflect the nature of biased behaviours in a particular environment, or better, learning techniques could be used to enable an agent to adjust this parameter according to the prevailing context.

In sum, this section shows that the credibility model of CR enables it to outperform SPORAS in dealing with biased behaviours. Specifically, it allows agents using CR to maintain a robust and high performance in a variety of cases, especially when the level of collusion is less than 50% (which is, in our opinion, the most likely case in realistic scenarios).

4. RELATED WORK

Many trust and reputation models have been devised in recent years due to the increasing recognition of their roles in controlling social order in open systems [7]. SPORAS [13] is one of the most notable of these models. In this model, each agent rates its partner after an interaction and reports its ratings to the centralised SPORAS repository. The received ratings are then used to update the global reputation values of the rated agents. The model uses a learning function for the updating process so that the reputation value can closely reflect an agent's performance. In addition, it also introduces a reliability measure based on the standard deviations of the rating values. However, it has been designed without considering the problem of inaccurate reports and so it suffers disproportionately when false information is collected (as shown in Section 3).

Speaking more generally, as many trust models are built

on witness reports, the problem of disinformation has come to the fore in several recent works on trust. Regret [8] models a witness' credibility based on the difference between that witness' opinion and an agent's past experience. This differs to our approach in that it depends on the availability of an agent's past experience. Moreover, this approach cannot deal with the situations where the target agent's behaviour changes since it does not take the new behaviour into account in its witness credibility assessment. Thus, it can falsely punish honest witnesses (who report the target agent's new behaviour). In Whitby et al.'s system [11], the "true" rating of an agent is defined by the majority's opinions. In particular, they model the performance of an agent as a beta probability density function (PDF) which is aggregated from all witness ratings received. Then a witness is considered unreliable and filtered out when the reputation derived from its ratings is judged to be too different from the majority's (by comparing the reputation value with the PDF). Due to the dependency on PDFs of witness reports, if these reports are scarce and/or too diverse it is not able to recognise lying witnesses. Moreover, it is possible that a witness can lie in a small proportion of their reports without being detected. In addition, isolated honest opinions (i.e. different than that of the majority) can be falsely punished. To rectify this, TRAVOS uses an approach that is similar to ours [10]. However, as it uses beta PDFs for representing trust derived from binary outcomes (i.e. 0 for 'failed', 1 for 'success'), it is not suitable for our CR model because we require a more fine-grained and continuous range for trust values. In our earlier work [5], we presented a preliminary model of CR. However, this model could not cope with collusion and did not take the variance of referees situations (see Section 3.3) into account.

The mechanism of CR has similarities to trust policy management engines such as PolicyMaker [3] and Trust-Serv [9]. These engines grant rights (i.e. trust) to an agent based on its certificates of its identity according to predefined policies (i.e. rules, such as 'if a is a registered user and it possesses a valid credit card then it can book flights'). However, these engines are designed to determine the access rights of agents, rather than to determine the expected performance of these agents (i.e. how the agents behave after they can access a particular service). Similarly, the PGP web of trust model [14] allows a person to prove the authenticity of his public key by others' digital signatures, but not that person's behaviour. Certified ratings are also similar to the concept of endorsements in [6] — certificates endorsing that a service (provider) is trusted and preferred by their issuers. However, such endorsements are one-value ratings and therefore cannot show how good that service is (as our certified ratings can by having values in $[-1, +1]$ in a context sensitive way using rating terms). Moreover, the work in [6] does not consider the problem of collusion.

5. CONCLUSIONS AND FUTURE WORK

This paper has presented a novel mechanism for a new type of trust — certified reputation. This model provides a number of advantages over current approaches. First, its mechanism addresses the problem of the lack of direct experience (since agents can typically collect a large number of references themselves and they are incentivised to present these to establish new trust relationships). Second, agents are freed from the various costs involved in locating witness reports (e.g. resource, time, and communication costs). Third, since CR allows agents to evaluate trust for themselves it does not require a centralised service and, thus, is compatible with open multi-agent environments. In addition, in our evaluation we have shown CR performs significantly better than both a direct experience only model and a successful centralised trust mechanism. We have also shown that our CR model is robust against various types of collusion between agents.

In the future, we aim to devise a method to automatically adjust the accuracy tolerance threshold during the system's operation (instead of handpicking a value as at present). This can be achieved by analysing the recorded performance levels of service providers that an agent has interacted with to determine the likely variability of honest ratings. With respect to the CR mechanism itself, we plan to elevate the current presenting references process into a full dialog between the evaluator and the target agent. In so doing, the evaluator can specify what type of reference it wants to see (e.g. only the most recent ones, only references about transactions of high values, or those from a selected set of referees). This dialog can be a full negotiation process in which the evaluator can ask why a reference is bad and the target agent can explain a plausible reason for that. This will further allow both agents to learn more about each other, which clearly facilitates the trust evaluation process.

6. REFERENCES

[1] P. R. Cohen. *Empirical Methods for Artificial Intelligence*. The MIT Press, 1995.

[2] D. Gambetta. *Trust: Making and Breaking Cooperative Relations*. Dept. of Sociology, University of Oxford, 2000.

[3] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Comm Surveys & Tutorials*, 3(4), 2000.

[4] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt. On handling inaccurate witness reports. In *Proc. 8th Int. Workshop on Trust in Agent Societies*, pages 63–77, 2005.

[5] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt. An integrated trust and reputation model for open multi-agent systems. *Journal of AAMAS*, 2006. (in press).

[6] E. M. Maximilien and M. P. Singh. Reputation and endorsement for web services. *ACM SIGEcom Exchanges*, 3(1):24–31, 2002.

[7] S. D. Ramchurn, T. D. Huynh, and N. R. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19(1):1–25, March 2004.

[8] J. Sabater. *Trust and Reputation for Agent Societies*. PhD thesis, Universitat Autònoma de Barcelona, 2003.

[9] H. Skogsrud, B. Benatallah, and F. Casati. Model-driven trust negotiation for web services. *IEEE Internet Computing*, 7(6):45–52, 2003.

[10] W. T. L. Teacy, J. Patel, N. R. Jennings, and M. Luck. Coping with inaccurate reputation sources: Experimental analysis of a probabilistic trust model. In *Proc. 4th Int Joint Conf on AAMAS*, pages 997–1004, 2005.

[11] A. Whitby, A. Jøsang, and J. Indulska. Filtering out unfair ratings in bayesian reputation systems. In *Proc. 7th Int Workshop on Trust in Agent Societies*, 2004.

[12] B. Yu and M. P. Singh. Detecting deception in reputation management. In *Proc. 2nd Int Joint Conf on Autonomous Agents and Multi-Agent Systems*, pages 73–80, 2003.

[13] G. Zacharia and P. Maes. Trust management through reputation mechanisms. *Applied Artificial Intelligence*, 14(9):881–908, 2000.

[14] P. R. Zimmermann. *The Official PGP Users Guide*. MIT Press, Cambridge, MA, 1995.

Trust and Honour in Information-based Agency

Carles Sierra
Institut d'Investigacio en Intel.ligencia Artificial
Spanish Scientific Research Council, UAB
08193 Bellaterra, Catalonia, Spain
sierra@iiia.csic.es

John Debenham
Faculty of Information Technology
University of Technology, Sydney
NSW, Australia
debenham@it.uts.edu.au

ABSTRACT

An argumentation based negotiation model is supported by information theory. Argumentative dialogues change the models of agents with respect to ongoing relationships. Trust and Honour are key components. Trust measures expected deviations of behaviour in the execution of commitments. Honour measures the expected integrity of the arguments exchanged. We understand rhetorical moves in dialogues as actions to project the current relationships into the future.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Theory

Keywords

Multiagent systems, Trust

1. INTRODUCTION

Negotiation dialogues have been traditionally organised around the basic illocutionary particles *Offer*, *Accept* and *Reject*. Previous work has been centred on the design of negotiation strategies and on proposing agent architectures able to deal with the exchange of offers [8, 4]. Game theory [13], possibilistic logic [5] and first-order logic [9] have been used for this purpose. Some initial steps in proposing rhetoric particles have been made, especially around the idea of *appeals*, *rewards* and *threats* [15]. However, no formal model of the meaning of these speech acts has been proposed yet. Expanded negotiation dialogues, including these rhetoric moves, are known as *argumentation-based negotiations*[1]. *Argumentation* in this sense is mainly to do with building (business) relationships. When we reward or threaten we refer to a future instant of time where the reward or threat will be effective, its scope goes beyond the current negotiation round. We will understand argumentation in this paper as an information exchange process between agents. Every illocution that an agent utters gives away valuable information. To evaluate each illocution exchanged we base our information-based agent architecture on information theory.

We distinguish between the *trust* that an agent displays through the enactment of its contracts, and the *honour* that an agent displays in sustaining its trading relationships. The enactment of a contract is uncertain to some extent, and trust, precisely, is a measure of how uncertain the enactment of a contract is. Trust is therefore a *measure of expected deviations of behaviour* along a dimension determined by the type of the contract [14]. In this sense, the higher the trust the lower the expectation that a (significant) deviation from what is signed occurs.

A (business) *relationship* between a set of agents consists of a historical sequence of related trades together with expectations of fair, trusted and honourable trade in the future. These expectations are established by the trading history and by the information exchanged [in appeals] and conditional promises made [in threats and rewards]. Exchanged information may turn out to be partly false, and promises may be partly broken. We introduce the concept of Honour to measure the *expected integrity of the information and promises exchanged*. This leads to a natural hierarchy: individual deals, (business) relationships and businesses.

We first introduce an argumentation language in Section 2. Then, after giving our intuitions on building relationships in Section 3 we define an information based model in Section 4 that supports our model of Honour in Section 5. How the information is managed is explained in Section 6. Finally, the agent architecture is explained in Section 7.

2. AN ARGUMENTATION LANGUAGE

In order for communication to be effective agents need to agree upon an ontology and a communication language.

2.1 Ontology

To model the agent dialogues we define *ontology* to include a (minimum) repertoire of elements: a set of *concepts* organised in an *is-a hierarchy*, captured by a partial order relation, a set of *relations* defined over these concepts, and a set of

[1]Not to be confused with argumentation in a classical sense: the generation of arguments, usually as logical proofs, for and against a given course of action that support decision making processes.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

axioms defined over the concepts and relations. We model ontologies following an algebraic approach and present ontologies as logical theories in the form of pairs $O = (S, A)$ where S is the *ontological signature*, representing the vocabulary, and A is a set of ontological axioms, specifying the interpreation of the vocabulary in a given context. The signature is usually represented as a mathematical structure and the set of axioms are usually written in a logical formalism. In our case, we define an ontology signature as a tuple $S = (C, R, \leq, \sigma)$ where C is a finite set of concept symbols (including basic data types); R is a finite set of relation symbols; $\leq$ is a reflexive, transitive and anti-symmetric relation on C (a partial order); and, $\sigma : R \to C^+$ is the function assigning to each relation symbol its arity. Concepts play somehow the role of *type*, and the is-a hierarchy is the notion of subtype. Thus, type inference mechanisms can be used to type all symbols appearing in expressions. Once the basic ontology signature is fixed we need to define a language to express, in our case, contracts, promises, rewards and so on. We denote that language as L_S. The axioms in the ontology will be written in that language, $A \subseteq L_S$, a first order language defined over the ontological expressions.

In our dialogues an essential concept is that of *contract*. Contracts are usually thought of as having a number of issues (dimensions) associated with values (or regions). Thus an (incomplete) example of ontology[2] for contracts could be defined as $\mathcal{S} = (C, R, \leq, \sigma)$ where
– $C = \{contract, quantity, integer, money, free, payment, date, plastic, wood, material, animal, crocodile\}$
– $R = \{price, made\_of, delivery, number, signed\}$
– $money \leq payment, free \leq payment, plastic \leq material, wood \leq material, crocodile \leq animal, integer \leq quantity$
– $\sigma(price) = contract \times payment, \sigma(made\_of) = contract \times material, \sigma(delivery) = contract \times date, \sigma(number) = contract \times quantity, \sigma(signed) = contract \times date$

For instance, the contract "3 plastic crocodiles at the price of €10" can be represented as:

$number(c_1, 3) \wedge made\_of(c_1, plastic) \wedge price(c_1, €10)$

A contract saying "3 crocodiles of any material, and at a maximum price of €10" becomes:

$number(c_1, 3) \wedge made\_of(c_1, material) \wedge price(c_1, [€0, €10])$

And, the expression "in all future contracts, I'll give you a free bottle of wine"

$\forall c : Contract\ .\ signed(c, [now, \infty]) \to present(c, bottle)$

We define the *ontological context* of formula $\varphi \in L_S$, denoted $O(\varphi)$, as the set of concepts in C used by the literals of φ. In the previous example, $O(\varphi) = \{contract, quantity, plastic, payment\}$. We extend the subtype relationship, $\leq$, in the following way,

$$\varphi \leq \psi \iff \forall c_i \in O(\varphi)\ (\exists c_j \in O(\psi).c_i \leq c_j)$$

We base the definition of the *semantic distance* between two concepts on the path length over the $\leq$ of the signature (more distance in the graph is more semantic distance), and the *depth* of the subsumer concept on the shortest path between the two concepts (the deeper in the hierarchy, the closer the meaning of the concepts) [10]:

$$Sim(c_1, c_2) = e^{-\kappa_1 l} \cdot \frac{e^{\kappa_2 h} - e^{-\kappa_2 h}}{e^{\kappa_2 h} + e^{-\kappa_2 h}}$$

[2]generic concepts and particular domains concepts are mixed in this example.

where l is the shortest path between the concepts, h is the depth of the deepest concept subsuming both concepts, and κ_1 and κ_2 are parameters scaling the contribution of shortest path length and depth respectively.

From the semantic distance between two concepts we define the semantic distance between formulae as the maxmin distance between concepts in their ontological context.

$$Sim(\varphi, \psi) = \max_{c_i \in O(\varphi)} \min_{c_j \in O(\psi)} \{Sim(c_i, c_j)\}$$

2.2 Communication language

Agent α is in a trading relationship with an agent β. They aim to strike deals $\delta = (a, b)$ where $a \in L_S$ is α's commitment and $b \in L_S$ is β's. Our agents share a communication language, $\mathcal{C}$, where the propositional content is expressed in L_S and the illocutionary particles are: [3]
– Offer(α, β, δ). Agent α offers agent β a deal $\delta = (a, b)$ with action commitments $a \in L_S$ for α and $b \in L_S$ for β.
– Accept(α, β, δ). Agent α accepts agent β's previously offered deal δ.
– Reject$(\alpha, \beta, \delta[, info])$. Agent α rejects agent β's previously offered deal δ. Optionally, information $info \in L_S$ explaining the reason for the rejection can be given.
– Withdraw$(\alpha, \beta[, info])$. Agent α breaks down negotiation with β. Extra $info \in L_S$ justifying the withdrawal may be given.
– Inform$(\alpha, \beta, info)$. Agent α informs β about $info \in L_S$ and commits to the truth of *info*.
– Reward$(\alpha, \beta, \delta, \varphi[, info])$. Intended to make the opponent accept a proposal with the promise of a future compensation. Agent α offers agent β a deal δ. In case β accepts the proposal, α commits to make $\varphi \in L_S$ true. The intended meaning is that α believes that worlds in which φ is true are somehow desired by β. Optionally, additional information in support of the deal can be given.
– Threat$(\alpha, \beta, \delta, \varphi, [info])$ Intended to make the opponent accept a proposal with the menace of some sort of retaliation. Agent α offers agent β a deal δ. In case β does not accept the proposal, α commits to make $\varphi \in L_S$ true. The intended meaning is that α believes that worlds in which φ is true are somehow not desired by β. Optionally, additional information in support of the deal can be given.
– Appeal$(\alpha, \beta, \delta, info)$ Intended to make the opponent accept a proposal as a consequence of the belief update that the accompanying information might bring about. Agent α offers agent β a deal δ. Additionally, α passes a pack of information in support of the deal. An Appeal can be understood as a combination of an offer and an inform, that is Appeal$(\alpha, \beta, \delta, info)$ = Offer(α, β, δ);Inform$(\alpha, \beta, info)$ — we borrow ';' from Dynamic Logic to mean action concatenation.

The accompanying information, *info*, can be of two basic types: (i) referring to the process (plan) used by an agent to solve a problem, or (ii) data (beliefs) of the agent including preferences. When building relationships, agents will therefore try to influence the opponent by changing their

[3]It is commonly accepted since the works by Austin and Searle that illocutionary acts are actions that succeed or fail. We will abuse notation in this paper and will consider that they are predicates in a first order logic meaning 'the action has been performed'. For those more pure-minded an alternative is to consider dynamic logic.

processes (plans) or by providing new data.

Dialogues, especially in Electronic Institutions [3], tend to be structured in order to facilitate the decision making of the participants. This structuring of dialogues is what is usually called a *protocol*. We will not detail it here, but will assume that such a protocol exists.

2.3 Information model predicates

The predicate $\text{Acc}(\alpha, \beta, \delta)$ means "agent α finds the deal δ with agent β acceptable", and $\text{Build}(\alpha, \beta, \rho)$ means "agent α considers agent β to be a potential trading partner for deals in a relationship ρ". Where ρ is a super type (more abstract set of concepts) than δ, that is $\delta \leq \rho$. ie: a relationship ρ could be "supply of chops", and a particular deal δ could be "supply 6 lamb chops today". Our agent estimates probabilities that are attached to $P(\text{Build}(\alpha, \beta, \rho))$ representing the certainty that it has in this proposition. Thus, if $P(\text{Build}(\alpha, \beta, \rho)) > P(\text{Build}(\alpha, \gamma, \rho))$ then it does *not* mean that β is better than γ. It *does* mean that for ρ-deals α is more certain that β is OK than that γ is OK.

To estimate $P(\text{Build}(\cdot))$ we express $\text{Build}(\cdot)$ in terms of a *social model* comprising:

– $\text{Capable}(\alpha, \beta, \rho)$ is a measure of the ability of an agent to do what it says it can do for deals of type ρ. Eg: "If β *says* he can supply both sausages and chops then he *can* supply both sausages and chops.".

– $\text{Trust}(\alpha, \beta, \rho)$ is a measure of expected deviations of behaviour along a given dimension (ρ). The assessment of Trust incorporates $\text{Reputation}(\beta)$ = "the reputation of β reported by other agents — not necessarily truthfully".

– $\text{Honour}(\alpha, \beta, \rho)$ measures the expected integrity of the arguments exchanged. That is, α's expectation that "β will honour previously issued appeals, threats and rewards when negotiating deals of type ρ". We expect rewards to compensate for bad proposals, and threats when our own position is seen to be dishonourable.

– $\text{Rel}(\alpha, \beta, c)$ measures the reliability of the information that β provides — this measure is keyed to sections of the ontological context, $c \in C$. For example, how reliable is β's information about wine, or about the stock market.

3. BUILDING RELATIONSHIPS

A simple bargaining agent α can only do three things: submit and respond to proposals, and "walk away". The only way in which a simple bargaining agent α can "reward or threaten" is to submit proposals that are more or less attractive to β than α would normally do so. A negotiation can "breakdown", and α may be interested in estimating $p_{b,\beta}$ — the probability that β will walk away in the subsequent negotiation round.

An argumentation agent extends the capacity of a simple bargaining agent by using threats, rewards and appeals to sustain trading relationships. An argumentation agent is concerned both with future single trades and with *trading relationships* that encapsulate expectations of future acceptable trades. Just as a negotiation can "breakdown", a trading relationship can *collapse*. So, just as a bargaining agent is interested in $p_{b,\beta}$, an argumentation agent will also be interested in estimating the probability that a trading relationship with β will collapse: $p_{c,\beta}$.

Our agent α uses argumentation to maintain a reliable set of potential trading partners, $\{\beta_i\}_{i \in \mathcal{B}_\rho}$, for each type of deal ρ. α considers β to be a reliable trading partner for ρ-deals if $P(\text{Build}(\alpha, \beta, \rho)) \geq \epsilon$ for some threshold constant ϵ. $\mathcal{B}_\rho = \{\beta \mid P(\text{Build}(\alpha, \beta, \rho)) \geq \epsilon\}$ will have an optimal size which is difficult to specify analytically due to dependencies between the probabilities. The size of the set $\mathcal{B}_\rho$ will be bounded by: first the difficulty in locating β_i for whom the p_{c,β_i} are "moderately" independent, and second the larger $\mathcal{B}_\rho$ the less opportunity α has to estimate $\{p_{c,\beta_i}\}_{i \in \mathcal{B}_\rho}$ directly. α may maintain these estimates indirectly by sharing information in a *trading pact*, but if α shares information about her trading partners then that may reduce the value of those partners to α.

Under what circumstances will α decide to "walk away" from a negotiation or a trading relationship? This is dealt with in α's negotiation plans, and α's plans for choosing potential trading partners — these are not described here. Presumably α will exercise her outside options if she believes that they will improve her position. α will only do this if $P(\text{Build}(\cdot))$ for her current partners are worse than the expected value for an untried partner. An argumentation agent's threat, reward and appeal illocutions are its weapons against collapse, as we discuss in Section 7.1.

4. AN INFORMATION-BASED MODEL

We ground our argumentation model on information-based concepts. *Entropy*, H, is a measure of uncertainty [11] in a probability distribution for a discrete random variable X: $H(X) \triangleq -\sum_i p(x_i) \log p(x_i)$ where $p(x_i) = P(X = x_i)$. Maximum entropy inference and minimum relative entropy inference are chosen partly because of their encapsulation of common sense reasoning [12].

Maximum entropy inference is used to derive sentence probabilities for that which is not known by constructing the "maximally noncommittal" [7] probability distribution, and minimum relative entropy inference is used to update these distributions. These forms of inference are criticised [6] for their dependence on the representation chosen — such as the way in which values for a continuous variable are represented as intervals. We argue to the contrary, that this choice enables the tailoring of the model in fine detail.

Given a prior probability distribution $\underline{q} = (q_i)_{i=1}^n$ and a set of constraints, the *principle of minimum relative entropy* chooses the posterior probability distribution $\underline{p} = (p_i)_{i=1}^n$ that has the least relative entropy with respect to $\underline{q}$, and that satisfies the constraints[4].

[1] describes the estimation of both $P(Acc(\alpha, \beta, \delta))$ and the estimation of $P(Acc(\beta, \alpha, \delta))$ which is α's estimate of β's willingness to accept δ. These estimates are derived by applying maximum entropy inference to the observed behaviour of the agents. In the subsequent subsection we'll see how α updates its sentence probabilities for $\text{Honour}(\cdot)$ by observation following receipt of the illocutionary particles described in Section 2.

4.1 Updating honour from observations

[4]Ie: $\arg\min_{\underline{p}} \sum_{i=1}^n p_i \log \frac{p_i}{q_i}$. The principle of minimum relative entropy is a generalisation of the principle of maximum entropy. If the prior distribution $\underline{q}$ is uniform, the relative entropy of $\underline{p}$ with respect to $\underline{q}$ differs from $-H(\underline{p})$ only by a constant. So the principle of maximum entropy is equivalent to the principle of minimum relative entropy with a uniform prior distribution.

The choice of trading partner is influenced by the observation of the fulfilment of informs, rewards and threats. Such illocutions are a *conditional commitment* to act. Eg: $\text{Reward}(\alpha, \beta, \delta, \varphi)$ when uttered by α has the meaning that α becomes socially committed to make φ happen if β enacts the illocution $\text{Accept}(\beta, \alpha, \delta)$. Similarly, $\text{Threat}(\alpha, \beta, \delta, \varphi)$ when uttered by α has the meaning that α becomes socially committed to make φ happen if β does *not* enact the illocution $\text{Accept}(\beta, \alpha, \delta)$ — usually by a given deadline. Finally, $\text{Inform}(\alpha, \beta, \varphi)$ means that α is socially committed to the truth of φ.

Agent α has the opportunity to observe the extent to which agent β sticks to his word in $\text{Reward}(\cdot)$, $\text{Threat}(\cdot)$ and $\text{Inform}(\cdot)$ illocutions, and to observe how accurate his *info* is. We base our measure of honour as the negative entropy of the probability distribution of possible outcomes following such a given illocution — honour measures the relationship between commitment and execution of promises. More precisely, between issued illocutions and their *perceived* execution. In this way, a natural way to base our modelling of honour is on a conditional probability, P^t, between commitment and observation given a context e as:

$$P^t(\text{Observe}(\alpha, \varphi') \mid (\text{Reward}(\beta, \alpha, \delta, \varphi); \text{Accept}(\alpha, \beta, \delta)), e)$$

where every reward execution represents a point in that distribution. For threats and informs the distributions are:

$$P^t(\text{Observe}(\alpha, \varphi') \mid (\text{Threat}(\beta, \alpha, \delta, \varphi); \neg\text{Accept}(\alpha, \beta, \delta)), e)$$

$$P^t(\text{Observe}(\alpha, \varphi') \mid (\text{Inform}(\beta, \alpha, \varphi)), e)$$

For simplicity, we denote these relationships between the commitment, φ, and the observation, φ', as $P^t(\varphi'|\varphi, e)$.

An important aspect that we want to model is the fact that beliefs 'evaporate' as time goes by. If we don't keep an ongoing relationship, we become unsure how *honourable* a trading partner is. If I stop buying meat from my butcher, I'm not sure anymore that he will commit to his promises. This decay is what justifies a continuous relationship between individuals. In our model, the conditional probabilities should tend to ignorance as represented by the *decay limit distribution* $\{d_i\}$. If we have the set of observations $\Phi = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ then complete ignorance of the opponent's expected behaviour means that given the opponent commits to φ the conditional probability for each observable outcome φ' becomes $d_i = \frac{1}{n}$ — i.e. the unconstrained maximum entropy distribution — but α may be prepared to make assumptions about β's decay limit distribution so that its entropy is less than this. This natural decay of belief is offset by new observations.

We define the evolution of the probability distribution that supports the previous definition of decay using an equation inspired by pheromone like models [2]:

$$P^{t+1}(\phi'|\phi) = \kappa \cdot \left((1-\nu) \cdot d_i + \nu \cdot \left(P^t(\phi'|\phi) + \Delta^t P(\phi'|\phi)\right)\right) \quad (1)$$

where κ is a normalisation constant and $\{d_i\}$ is the decay limit distribution for β. This equation models the passage of time for a conveniently large $\nu \in [0, 1]$ and where the term $\Delta^t P(\phi'|\phi)$ represents the increment in an instant of time according to the last experienced event as the following shows.

Similarity based. The question is how to use the observation φ' given a commitment φ (generated by a sequence of illocutions $\text{Reward}(\alpha, \beta, \delta, \varphi)$; $\text{Accept}(\beta, \alpha, \delta)$ — similarly for others) in the context of a deal $\delta \leq \rho$ in the update of the overall probability distribution over the set of all possible outcomes defined over L_C. Here we use the idea that given a particular deviation in a region of the space, *similar* deviations should be expected in other regions. The intuition behind the update is that if my butcher has not honoured commitments with respect to red meat, then I might expect similar deviations with respect to poultry.

This idea is built upon the previously defined Sim function (see Section 2) to take into account the difference between acceptance probabilities and similarity between the perception of the execution x of a reward (or threat) y. Thus, after the observation of φ' the increment of probability distribution at time $t+1$ is:

$$\Delta^t P(\phi'|\phi) = (1 - \mid Sim(\varphi', \varphi) - Sim(\phi', \phi) \mid) \quad (2)$$

Entropy based. Suppose that α has a business relationship ρ with agent β, that β promises ϕ, and this promise is sound. The material value of ϕ to ρ will depend on the future use that α makes of it, and that is unlikely to be known. So α estimates the value of ϕ to the honour he holds for β in ρ using a probability distribution $(p_1, \ldots, p_n)$ over a relationship evaluation space $E = (e_1, \ldots, e_n)$ that could range from "that is what I expect from the perfect trading partner" to "it is totally useless" — E may contain hard or fuzzy values. $p_i = w_i(\rho, \phi)$ is the probability that e_i is the correct evaluation of the enactment ϕ in the context of relationship ρ, and $\underline{w} : L_S \times L_S \to [0, 1]^n$ is the promise *evaluation function*.

Let $L_S = (\phi_1, \ldots, \phi_m)$ in some order. Then for a given ϕ_k, $(P^t_\beta(\phi_1|\phi_k), \ldots, P^t_\beta(\phi_m|\phi_k))$ is the prior distribution of α's estimate of what will actually occur if β promised that ϕ_k would occur and $\underline{w}(\rho, \phi_k) = (w_1(\rho, \phi_k), \ldots, w_n(\rho, \phi_k))$ is α's evaluation over E with respect to the relationship ρ of β's promise ϕ_k. α's expected evaluation of what will occur given that β has promised that ϕ_k will occur is: $\underline{w}^{\exp}(\rho, \phi_k) =$

$$\left(\sum_{j=1}^{m} P^t_\beta(\phi_j|\phi_k) \cdot w_1(\rho, \phi_j), \ldots, \sum_{j=1}^{m} P^t_\beta(\phi_j|\phi_k) \cdot w_n(\rho, \phi_j)\right).$$

Now suppose that α observes the event $(\varphi'|\varphi)$ in another relationship ρ' also with agent β. Eg: α may buy wine and cheese from the same supplier. α may wish to revise the prior estimate $\underline{w}^{\exp}(\rho, \phi_k)$ in the light of the observation $(\varphi'|\varphi)$ to: $(\underline{w}^{\text{rev}}(\rho, \phi_k) \mid (\varphi'|\varphi)) =$

$$\underline{g}(\underline{w}^{\exp}(\rho, \phi_k), \underline{w}(\rho', \varphi), \underline{w}(\rho', \varphi'), \rho, \rho', \phi, \varphi, \varphi'),$$

for some function $\underline{g}$ — the idea being, for example, that if the promise, φ, concerning the purchase of cheese, ρ', was not kept then α's expectation that the promise, ϕ, concerning the purchase of wine, ρ, will not be kept should increase. The entropy based approach estimates $\Delta^t_\beta P(\phi'|\phi)$ by applying the principle of minimum relative entropy[4]. Let:

$$\left(P^t_{\beta,C}(\phi_j|\phi)\right)_{j=1}^m = \arg\min_{\underline{p}} \sum_{i=1}^{m} p_i \log \frac{p_i}{P^t_\beta(\phi_i|\phi)} \quad (3)$$

satisfying the n constraints C, and $\underline{p} = (p_j)_{j=1}^m$. Then:

$$\Delta^t P(\phi'|\phi) = P^t_{\beta,C}(\phi'|\phi) - P^t_\beta(\phi'|\phi) \quad (4)$$

Where the n constraints C are: $\sum_{j=1}^{m} p_j \cdot w_i(\rho, \phi_j) =$

$$g_i(\underline{w}^{\exp}(\rho, \phi_k), \underline{w}(\rho', \varphi), \underline{w}(\rho', \varphi'), \rho, \rho', \phi, \varphi, \varphi')$$

for $i = 1, \ldots, n$. This is a set of n linear equations in m unknowns, and so the calculation of the minimum relative entropy distribution may be impossible if $n > m$. In this case, we take only the m equations for which the change from the prior to the posterior value is greatest. That is, we attempt to select the most significant factors.

5. AN HONOUR MODEL

Honour as expected behaviour. Consider a distribution of expected fulfilment of promises that represent α's "ideal" for a relationship with β, in the sense that it is the best that α could reasonably expect β to do. This distribution will be a function of β, α's history with β, anything else that α believes about β, and general environmental information including time — denote all of this by e, then we have $P_I^t(\varphi'|\varphi, e)$. For example, if α considers that it is unacceptable for the execution φ' to be less preferred than the promise φ then $P_I^t(\varphi'|\varphi, e)$ will only be non-zero for those φ' that α prefers to φ. The distribution $P_I^t(\cdot)$ represents what α expects, or hopes, β will do. Honour is the relative entropy between this ideal distribution, $P_I^t(\varphi'|\varphi, e)$, and the distribution of the observation of fulfilled promises, $P_\beta^t(\varphi'|\varphi)$. That is:

$$H(\alpha, \beta, \varphi) = 1 - \sum_{\varphi'} P_I^t(\varphi'|\varphi, e) \log \frac{P_I^t(\varphi'|\varphi, e)}{P_\beta^t(\varphi'|\varphi)} \quad (5)$$

where the "1" is an arbitrarily chosen constant being the maximum value that honour may have. This equation defines honour for one, single promise φ — for example, my honour in my butcher if he promises me a 10% discount for the rest of the year. It makes sense to aggregate these values over a class of promises, say over those φ that are subtypes of a particular relationship ρ, that is $\varphi \leq \rho$. In this way we measure the honour that I have in my butcher in relation to the promises he makes for red meat generally:

$$H(\alpha, \beta, \rho) = 1 - \frac{\sum_{\varphi:\varphi\leq\rho} P_\beta^t(\varphi) \left[\sum_{\varphi'} P_I^t(\varphi'|\varphi, e) \log \frac{P_I^t(\varphi'|\varphi, e)}{P_\beta^t(\varphi'|\varphi)}\right]}{\sum_{\varphi:\varphi\leq\rho} P_\beta^t(\varphi)}$$

where $P_\beta^t(\varphi)$ is a probability distribution over the space of promises that the next promise β will make to α is φ. Similarly, for an overall estimate of α's honour in β:

$$H(\alpha, \beta) = 1 - \sum_{\varphi} P_\beta^t(\varphi) \left[\sum_{\varphi'} P_I^t(\varphi'|\varphi, e) \log \frac{P_I^t(\varphi'|\varphi, e)}{P_\beta^t(\varphi'|\varphi)}\right]$$

Honour as expected 'preferability'. The previous notion of honour requires that an ideal distribution, $P_I^t(\varphi'|\varphi, e)$, has to be specified for each φ. The idea behind honour as expected 'preferability' is that in a trading relationship, ρ, if the execution φ' of a promise is preferable to that that was promised φ then α will honour β more. The extent to which an enactment, φ', can be 'preferable' to a promise, φ, is limited. For example, if my grocer promises me a 10% discount on tomatoes and gives me 15% then I may be delighted, but if he gives me a 95% discount then my desire to trade with him may diminish. Given a predicate $\text{Prefer}(c_1, c_2, e)$ meaning that α prefers c_1 to c_2 in environment e. An evaluation of $P^t(\text{Prefer}(c_1, c_2, e))$ may be defined using $Sim(\cdot)$ and the evaluation function $\underline{w}$ — but we do not detail it here. Then if $\varphi \leq \rho$:

$$H(\alpha, \beta, \varphi) = \sum_{\varphi'} P^t(\text{Prefer}(\varphi', \varphi, \rho)) P_\beta^t(\varphi' \mid \varphi)$$

and:

$$H(\alpha, \beta, \rho) = \frac{\sum_{\varphi:\varphi\leq\rho} P_\beta^t(\varphi) \left[\sum_{\varphi'} P^t(\text{Prefer}(\varphi', \varphi, \rho)) P_\beta^t(\varphi' \mid \varphi)\right]}{\sum_{\varphi:\varphi\leq\rho} P_\beta^t(\varphi)}$$

Honour as certainty in promise execution. Honour is consistency in expected acceptable executions of promises, or "the lack of expected uncertainty in those possible executions that are better than the promise as specified". Then if $\varphi \leq \rho$ let: $\Phi_+(\varphi, \rho, \kappa) = \{\varphi' \mid P^t(\text{Prefer}(\varphi', \varphi, \rho)) > \kappa\}$ for some constant κ. The idea here is that α will honour β more if variations, φ', from expectation, φ, are not random. The Honour that an agent α has on agent β with respect to the execution of a promise φ is:

$$H(\alpha, \beta, \varphi) = 1 + \frac{1}{B^*} \cdot \sum_{\varphi' \in \Phi_+(\varphi, \rho, \kappa)} P_+^t(\varphi'|\varphi) \log P_+^t(\varphi'|\varphi)$$

where $P_+^t(\varphi'|\varphi)$ is the normalisation of $P_\beta^t(\varphi'|\varphi)$ for $\varphi' \in \Phi_+(\varphi, \rho, \kappa)$,

$$B^* = \begin{cases} 1 & \text{if } |\Phi_+(\varphi, \rho, \kappa)| = 1 \\ \log |\Phi_+(\varphi, \rho, \kappa)| & \text{otherwise} \end{cases}$$

As above we aggregate this measure for those promises of a particular type ρ:

$$H(\alpha, \beta, \rho) = 1 + \frac{\sum_{\varphi:\varphi\leq\rho} \left[P_\beta^t(\varphi) \cdot \sum_{\varphi' \in \Phi_+(\varphi, \rho, \kappa)} P_+^t(\varphi'|\varphi) \log P_+^t(\varphi'|\varphi)\right]}{B^* \cdot \sum_{\varphi:\varphi\leq\rho} P_\beta^t(\varphi)} = 1 + \frac{\sum_{\varphi:\varphi\leq\rho} \sum_{\varphi' \in \Phi_+(\varphi, \rho, \kappa)} \left[P_+^t(\varphi', \varphi) \log P_+^t(\varphi'|\varphi)\right]}{B^* \cdot \sum_{\varphi:\varphi\leq\rho} P_\beta^t(\varphi)}$$

where $P_\beta^t(\varphi', \varphi)$ is the joint probability distribution, and $P_+^t(\varphi', \varphi)$ is the normalisation of it as above. And, similarly as before:

$$H(\alpha, \beta) = 1 + \frac{\sum_{\varphi} \left[P_\beta^t(\varphi) \cdot \sum_{\varphi' \in \Phi_+(\varphi, \rho, \kappa)} P_+^t(\varphi'|\varphi) \log P_+^t(\varphi'|\varphi)\right]}{B^*} = 1 + \frac{\sum_{\varphi} \sum_{\varphi' \in \Phi_+(\varphi, \rho, \kappa)} \left[P_+^t(\varphi', \varphi) \log P_+^t(\varphi'|\varphi)\right]}{B^*}$$

Evaluating honour. The various expressions for honour given above involve extensive calculations. For example, Eqn. 5 contains $\sum_{\varphi'}$ that sums over *all* possible enactments φ'. To develop the above into a more computationally friendly model we appeal to the structure of the ontology described in Sec. 2.1. For example, Eqn. 5 may be approximated to:

$$H(\alpha, \beta, \varphi) = 1 - \sum_{\varphi': Sim(\varphi', \varphi) \geq \eta} P_{\eta,I}^t(\varphi'|\varphi, e) \log \frac{P_{\eta,I}^t(\varphi'|\varphi, e)}{P_{\eta,\beta}^t(\varphi'|\varphi)}$$

where $P^t_{\eta,I}(\varphi'|\varphi,e)$ is the normalisation of $P^t_I(\varphi'|\varphi,e)$ for $Sim(\varphi',\varphi) \geq \eta$, and similarly for $P^t_{\eta,\beta}(\varphi'|\varphi)$. The extent of this calculation is controlled by the parameter η. An even tighter restriction may be obtained with: $Sim(\varphi',\varphi) \geq \eta$ *and* $\varphi' \leq \psi$. For example, if β promises a 10% discount on tomatoes next month, we may choose to evaluate $H(\alpha,\beta,\varphi)$ for promises φ' that are *both* close to this promise, φ, *and* that concern tomatoes, ψ.

6. DEALING WITH [*INFO*].

In this section we discuss how to deal with the information that β communicates in the illocutions, $info \in L_C$. We will base this treatment on β's *reliability* as an estimate of the extent to which this *info* is correct. For example, β may send α the *info* that "the price of fish will go up by 10% next week", and it may actually go up by 9%. α's argumentation and relationship building strategies are based on plans that are not described here. Those plans determine a set of probability distributions from which a world model is derived. *info* is represented as a set of linear constraints on one or more of those probability distributions. A chunk of *info* may not be directly related to one of those distributions, or may not be expressed naturally as constraints, and so some inference machinery is required to derive these constraints — this inference is performed by a set of model building function, J_s, for each active plan s chosen by α. If a plan s calls for the distribution D then J^D_s is the model building function for D, and $J^D_s(info)$ is the set of constraints on D derived from *info*.

6.1 Updating the world model with *info*

If at time u, α receives a message containing *info* it is time-stamped and source-stamped $info_{(\beta,\alpha,u)}$, and placed in a repository $\mathcal{Y}^t$. If α has an active plan, s, then the model building function, J_s, is applied to $info_{(\beta,\alpha,u)}$ to derive constraints on some, or none, of α's distributions. The extent to which those constraints are permitted to effect the distributions is determined by a value for the *reliability* of β, $R^t(\alpha,\beta,O(info))$.

An agent may have models of integrity decay for some particular distributions, but general models of integrity decay for, say, a chunk of information taken at random from the World Wide Web are generally unknown. However the values to which decaying integrity should tend in time *are* often known. For example, a prior value for the truth of the proposition that a "22 year-old male will default on credit card repayment" is well known to banks. As described in Sec. 4.1, if α attaches such prior values to a distribution D they are called the *decay limit distribution* for D, $(d^D_i)^n_{i=1}$. No matter how integrity of *info* decays, in the absence of any other relevant information it should decay to the decay limit distribution.

In the absence of new *info* the integrity of distributions decays. If $D = (q_i)^n_{i=1}$ then we use a geometric model of decay:

$$q^{t+1}_i = (1-\nu^D) \times d^D_i + \nu^D \times q^t_i, \text{ for } i = 1,\ldots,n \qquad (6)$$

where $\nu^D \in (0,1)$ is the decay rate. This raises the question of how to determine ν^D. Just as an agent may know the decay limit distribution it may also know something about ν^D. In the case of an information-overfed agent there is no harm in conservatively setting ν^D "a bit on the low side" as the continually arriving *info* will sustain the estimate for D.

We now describe how new *info* is imported to the distributions. A single chunk of *info* may affect a number of distributions. Suppose that a chunk of *info* is received from β and that α attaches the epistemic belief probability $R^t(\alpha,\beta,O(info))$ to it. Each distribution models a facet of the world. Given a distribution $D = (q_i)^n_{i=1}$, q_i is the probability that the possible world ω_i for D is the true world for D. The effect that a chunk *info* has on distribution D is to enforce the set of linear constraints on D, $J^D_s(info)$. If the constraints $J^D_s(info)$ are taken by α as valid then α could update D to the posterior distribution $(p^{info}_i)^n_{i=1}$ that is the distribution with least relative entropy with respect to $(q^t_i)^n_{i=1}$ satisfying the constraint:

$$\sum_i \{p^{info}_i : J^D_s(info) \text{ are all } \top \text{ in } \omega_i\} = 1. \qquad (7)$$

But $R^t(\alpha,\beta,O(info)) = r \in [0,1]$ and α should only treat the $J^D_s(info)$ as valid if $r = 1$. In general r determines the extent to which the effect of *info* on D makes it closer to $(p^{info}_i)^n_{i=1}$ or to the prior $(q^t_i)^n_{i=1}$ distribution by:

$$p^t_i = r \times p^{info}_i + (1-r) \times q^t_i \qquad (8)$$

But, we should only permit a new chunk of *info* to influence D if doing so gives us new information. For example, if 5 minutes ago a trusted agent advises α that the interest rate will go up by 1%, and 1 minute ago a very unreliable agent advises α that the interest rate may go up by 0.5%, then the second unreliable chunk should not be permitted to 'overwrite' the first. We capture this by only permitting a new chunk of *info* to be imported if the resulting distribution has more information *relative to* the decay limit distribution than the existing distribution has. Precisely, this is measured using the Kullback-Leibler distance measure[5], and *info* is only used if:

$$\sum_{i=1}^n p^t_i \log \frac{p^t_i}{d^D_i} > \sum_{i=1}^n q^t_i \log \frac{q^t_i}{d^D_i} \qquad (9)$$

In addition, we have described in Eqn. 6 how the integrity of each distribution D will decay in time. Combining these two into one result, distribution D is revised to:

$$q^{t+1}_i = \begin{cases} (1-\nu^D) \times d^D_i + \nu^D \times p^t_i & \textit{if} \text{ usable } \textit{info} \text{ is} \\ & \text{received at time } t \\ (1-\nu^D) \times d^D_i + \nu^D \times q^t_i & \textit{otherwise} \end{cases}$$

for $i = 1,\cdots,n$, and decay rate ν^D as before. We have yet to estimate $R^t(\alpha,\beta,O(info))$ — that is described in Sec. 6.2. We now illustrate the above by showing how an important class of *info* illocutions, namely preference information, is managed.

Preferences. Preference information is a statement by an agent that it prefers one class of deals to another where deals may be multi-issue. Preference illocutions may refer to particular issues within deals — e.g. "I prefer red to yellow", or to combinations of issues — e.g. "I prefer a car with a five year warranty to the same car with a two year warranty than costs 15% less". Here α receives preference information, *info*, from β through an $\text{Inform}(\beta,\alpha,info)$ illocution, or through one of the other argumentation illocutions — see Sec. 2 — and attaches $R^t(\alpha,\beta,O(info))$ to it.

[5]This is just one criterion for determining whether the *info* should be used.

What happens next will depend on α's plans. Suppose that α has a plan s that constructs the probability distribution $P^t(\text{UPrefer}(\beta, \alpha, \delta))$ over all δ meaning that "δ is the deal that β prefers most". α may know the decay limit distribution for UPrefer$(\cdot)$. Suppose α has a prior distribution $(q_i^t)_{i=1}^n$ for UPrefer$(\cdot)$. Now suppose that α learns the *info* "x% of the time β prefers deals with property Q_1 to those with property Q_2". J_s^{UPrefer} derives the following linear constraint on the $P^t(\text{UPrefer}(\beta, \alpha, \delta))$ distribution:

$$\frac{x}{100} = \frac{\sum_{\delta:Q_1(\delta)} p_\delta}{\left(\sum_{\delta:Q_1(\delta)} p_\delta\right) + \left(\sum_{\delta:Q_2(\delta)} p_\delta\right) - \left(\sum_{\delta:Q_1 \wedge Q_2(\delta)} p_\delta\right)}$$

and the procedure described above follows as described. Note that this manages two different probabilities: first, the probability within the *info* "x% of the time...", and second α's estimate of β's reliability in providing information of this sort, $R^t(\alpha, \beta, O(info))$, that is determined by the ontological context $O(info)$ of the *info*.

6.2 A reliability model

We estimate $R^t(\alpha, \beta, O(info))$ by measuring the error in information. α's plans will have constructed a set of distributions. We measure the 'error' in information as the error in the effect that information has on each of α's distributions. Suppose that a chunk of *info* is received from agent β at time u and is verified at some later time t. For example, a chunk of information could be "the interest rate will rise by 0.5% next week", and suppose that the interest rate actually rises by 0.25% — call that correct information *fact*. What does all this tell agent α about agent β's reliability? Consider one of α's distributions D that is $\{q_i^u\}$ at time u. Let $(p_i^{info})_{i=1}^n$ be the minimum relative entropy distribution given that *info* has been received as calculated in Eqn. 7, and let $(p_i^{fact})_{i=1}^n$ be that distribution if *fact* had been received instead. Suppose that the reliability estimate for distribution D was R_D^u. This section is concerned with what R_D^u should have been in the light of knowing *now*, at time t, that *info* should have been *fact*, and how that knowledge effects our current reliability estimate for D, $R^t(\alpha, \beta, O(info))$.

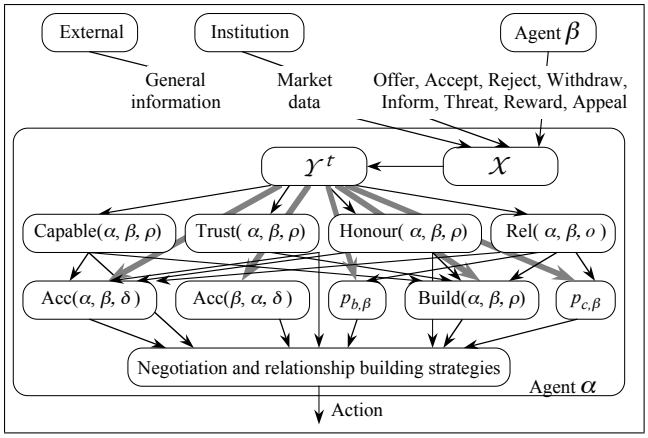
The idea of Eqn. 8, is that the current value of r should be such that, *on average*, $(p_i^u)_{i=1}^n$ will be seen to be "close to" $(p_i^{fact})_{i=1}^n$ when we eventually discover *fact* — no matter whether or not *info* was used to update D, as determined by the acceptability test in Eqn. 9 at time u. That is, given *info*, *fact* and the prior $(q_i^u)_{i=1}^n$, calculate $(p_i^{info})_{i=1}^n$ and $(p_i^{fact})_{i=1}^n$ using Eqn. 7. Then the *observed reliability* for distribution D, $R_D^{(info|fact)}$, on the basis of the verification of *info* with *fact* is the value of r that minimises the Kullback-Leibler distance between $(p_i^u)_{i=1}^n$ and $(p_i^{fact})_{i=1}^n$:

$$\arg\min_r \sum_{i=1}^n (r \cdot p_i^{info} + (1-r) \cdot q_i^u) \log \frac{r \cdot p_i^{info} + (1-r) \cdot q_i^u}{p_i^{fact}}$$

If E^{info} is the set of distributions that *info* affects, then the overall *observed reliability* on the basis of the verification of *info* with *fact* is: $R^{(info|fact)} = 1 - (\max_{D \in E^{info}} |1 - R_D^{(info|fact)}|)$. Then for each ontological context o_j, at time t when, perhaps, a chunk of *info*, with $O(info) = o_k$, may have been verified with *fact*:

$$R^{t+1}(\alpha, \beta, o_j) = (1-\nu) \times R^t(\alpha, \beta, o_j) + \nu \times R^{(info|fact)} \times \text{Sim}(o_j, o_k)$$

Figure 1: Trust, Honour and Reliability in the informed agent. The figure shows the social model and a sample world model. The social model consists of the four components: Capable$(\cdot)$**,** Trust$(\cdot)$**,** Honour$(\cdot)$ **and** Rel$(\cdot)$ **standing for 'reliability'. The sample world model contains just five distributions with the thick arcs representing the five model building functions J_s^D.**



where Sim measures the semantic distance between two sections of the ontology, and ν is the learning rate. Over time, α notes the ontological context of the various chunks of *info* received from β and over the various ontological contexts calculates the relative frequency, $P^t(o_j)$, of these contexts, $o_j = O(info)$. This leads to a overall expectation of the *reliability* that agent α has for agent β:

$$R^t(\alpha, \beta) = \sum_j P^t(o_j) \times R^t(\alpha, \beta, o_j)$$

7. AGENT ARCHITECTURE

The agent architecture developed so far is summarised in Fig. 1. α's actions are determined by its relationship-building strategy that identifies trading partners, and by its negotiation strategy that manages individual negotiations. The complete apparatus in Fig. 1 is revised when any illocution is received.

The *social model*: Capable$(\cdot)$, Trust$(\cdot)$, Honour$(\cdot)$, and Reliability$(\cdot)$, presents a summary of the interaction history weighted towards recent observations. It provides the foundation for α's relationship building strategy. The estimation of Capable$(\cdot)$ depends on the commitments involved — for example, estimating whether β is capable of raising a sum of money presents a different problem to estimating whether β can supply 100 kilos of sausages — the estimation of Capable$(\cdot)$ is not discussed here. An estimation of Trust$(\cdot)$ is discussed in [14], and herein Honour in Sec. 5 and Reliability in Sec. 6. The world model is described following.

7.1 World Model

Agent α acts by activating a plan s that invokes model building functions J_s^D and uses one of the two entropy inference methods [see Sec. 4] to "fill in the gaps" in probability dis-

tribution D. The *world model* is then aggregated from these distributions. Fig. 1 shows a sample world model consisting of: $\text{Acc}(\alpha, \beta, \delta)$, $\text{Acc}(\beta, \alpha, \delta)$, $p_{b,\beta}$, $\text{Build}(\alpha, \beta, \rho)$ and $p_{c,\beta}$. The components of the world model estimate of α's belief that a proposition is true.

$P^t(\text{Acc}(\alpha, \beta, \delta))$ is α's estimate that δ is an acceptable deal from agent β — this probability expresses the certainty that α has in this proposition. So $P^t(\text{Acc}(\alpha, \beta, \delta_1)) > P^t(\text{Acc}(\alpha, \beta, \delta_2))$ does not mean that δ_1 is an intrinsically better deal than δ_2 — it means that α is more certain that δ_1 is acceptable than δ_2 is acceptable.[6] A framework for estimating the $P^t(\text{Acc}(\alpha, \beta, \delta))$ distribution, derived from subjective, market and general observations, is described in [1].

$P^t(\text{Acc}(\beta, \alpha, \delta))$ is α's estimate that δ is an acceptable deal to β. [1] estimates $P^t(\text{Acc}(\beta, \alpha, \delta))$ by observing the Offer$(\cdot)$, Accept$(\cdot)$ and Reject$(\cdot)$ illocutions received from β.

$p_{b,\beta}$ is estimated by observing β's behaviour. A simple method is to measure the local entropy of the $P^t(\text{Acc}(\beta, \alpha, \delta))$ distribution — that is, the entropy of the normalisation of that distribution for δ restricted to the negotiation region. If that entropy measurement does not fall in time as the negotiation proceeds then this may indicate β's intention to withdraw from the negotiation. [1] describes the estimation of $p_{b,\beta}$.

A natural way to define $\text{Build}(\alpha, \beta, \rho)$ is by combining the four concepts we claimed were its pillars:

$$P(\text{Build}(\alpha, \beta, \rho)) = k(\text{Capable}(\alpha, \beta, \rho), \text{Trust}(\alpha, \beta, \rho), \text{Honour}(\alpha, \beta, \rho), \text{Rel}(\alpha, \beta, O(\rho)))$$

for some function k, where $O(\rho)$ denotes the ontological context of relationship ρ. This characterisation of Build is defined in terms of (commitment, observation) pairs for each of its four components.

A simple negotiation agent aims to strike a deal δ with agent β whilst avoiding the risk of negotiation breakdown. Similarly, an argumentation agent aims to build a trading relationship ρ with agent β whilst avoiding the risk of relationship collapse. To estimate $p_{c,\beta}$ we observe β for odd behaviour. Indications of collapse include: individual negotiations that have a high number of rounds or that take a significant time, reluctance to benefit from attractive terms offered in α's Reward$(\cdot)$s, β offers unrealistic Reward$(\cdot)$s, β issues Threats$(\cdot)$s, or, if the structure of the electronic institution permits general observation of β's trading, that β has taken his business elsewhere.

8. CONCLUSION

We introduced in this paper two new concepts: Honour$(\cdot)$ and Reliability$(\cdot)$. Honour$(\cdot)$ models the fulfilment of the promises derived from the use of argumentative particles such as *appeals*, *rewards* and *threats*. This concept plays a similar role to that that the concept of Trust$(\cdot)$ plays with respect to the fulfilment of contract agreements, as developed in [14]. Reliability$(\cdot)$ models the reliability of the information tabled by an opponent. We have proposed a semantics for these two concepts based on information theory. Also, an agent architecture to support this model is proposed.

[6]We differ from game-theory-based agents in that our agent is not necessarily "utility aware".

9. REFERENCES

[1] Debenham, J. Bargaining with information. In *Proceedings Third International Conference on Autonomous Agents and Multi Agent Systems AAMAS-2004* (July 2004), N. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, Eds., ACM, pp. 664 – 671.

[2] Dorigo, M., and Stützle, T. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

[3] Esteva, M., Padget, J., and Sierra, C. Formalizing a languages for institutions and norms. In *Intelligent Agents VIII*, J. Meyer and M. Tambe, Eds. Springer-Verlag, Berlin, Germany, 2002, pp. 348 – 366.

[4] Faratin, P., Sierra, C., and Jennings, N. Using similarity criteria to make issue trade-offs in automated negotiation. *Journal of Artificial Intelligence 142*, 2 (2003), 205–237.

[5] Giménez, E., Godo, L., Rodríguez-Aguilar, J. A., and Garcia, P. Designing bidding strategies for trading agents in electronic auctions. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS-98)* (1998), pp. 136–143.

[6] Halpern, J. *Reasoning about Uncertainty*. MIT Press, 2003.

[7] Jaynes, E. *Probability Theory — The Logic of Science*. Cambridge University Press, 2003.

[8] Jennings, N., Faratin, P., Lomuscio, A., Parsons, S., Sierra, C., and Wooldridge, M. Automated negotiation: Prospects, methods and challenges. *International Journal of Group Decision and Negotiation 10*, 2 (2001), 199–215.

[9] Kraus, S. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence 94*, 1–2 (1997), 79–97.

[10] Li, Y., Bandar, Z. A., and McLean, D. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering 14(4)* (2003).

[11] MacKay, D. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[12] Paris, J. Common sense and maximum entropy. *Synthese 117*, 1 (1999), 75 – 93.

[13] Rosenschein, J. S., and Zlotkin, G. *Rules of Encounter*. The MIT Press, Cambridge, USA, 1994.

[14] Sierra, C., and Debenham, J. An information-based model for trust. In *Proceedings Fourth International Conference on Autonomous Agents and Multi Agent Systems AAMAS-2005* (Utrecht, The Netherlands, July 2005), F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. Singh, and M. Wooldridge, Eds., ACM Press, New York, pp. 497 – 504.

[15] Sierra, C., Jennings, N., Noriega, P., and Parsons, S. *Proceedings of the 4th International Workshop on Intelligent Agents IV, Agent Theories, Architectures, and Languages*. Springer-Verlag, London, UK, 1997, ch. A Framework for Argumentation-Based Negotiation, pp. 177 – 192.

Using CHI-Scores to Reward Honest Feedback from Repeated Interactions *

Radu Jurca
Ecole Polytechnique Fédérale de Lausanne
Artificial Intelligence Laboratory
CH-1015 Lausanne, Switzerland
radu.jurca@epfl.ch

Boi Faltings
Ecole Polytechnique Fédérale de Lausanne
Artificial Intelligence Laboratory
CH-1015 Lausanne, Switzerland
boi.faltings@epfl.ch

ABSTRACT

Online communities increasingly rely on reputation information to foster cooperation and deter cheating. As rational agents can often benefit from misreporting their observations, explicit incentives must be created to reward honest feedback. Reputation side-payments (e.g., agents get paid for submitting feedback) can be designed to make truth-telling optimal. In this paper, we present a new side-payment scheme adapted for settings where agents repeatedly submit feedback. We rate the *feedback set* of an agent, rather than individual reports. The *CHI-Score* of the feedback set is computed based on a Chi-square independence test that assesses the correlation between the agent's feedback and the feedback submitted by the rest of the community. The mechanism has intuitive appeal and generates significantly lower costs than existing incentive-compatible reporting mechanisms.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms, Design, Economics

Keywords

reputation mechanisms, honest reporting

1. INTRODUCTION

Reputation mechanisms emerge as a promising solution for building trust in online communities lacking objective monitoring and enforcing authorities. The experience of previous interactions (i.e., reputation information) is an indicator of future performance; when feedback is shared within the community, agents can accurately identify malicious partners and thus protect themselves against cheating. The *Feedback Forum* of eBay is a notorious example of how reputation mechanisms make business possible in a decentralized environment where two agents have few a priori reasons to trust each other.

In a service-oriented market, reputation mechanisms can do more than just identify and isolate malicious providers. Perfect service is impossible, or prohibitively expensive to provide. Occasional failures happen against the best intention of service providers; excluding such providers from the market would be unfair and inefficient. Instead, reputation information can be used to approximate the real quality of service (QoS) delivered by a provider, and consequently, to dynamically scale the price paid for service [10]. Such a flexible pricing scheme allows service providers with different characteristics to reach the best compromise between delivered quality (hence revenue) and costs. Jurca and Faltings [11] show how reputation information can become part of the Service Level Agreements (SLAs) and efficiently substitute external monitoring of Quality of Service (QoS). They prove that such *reputation-based* SLAs make it rational for service providers to keep their promises, leading to an efficient and trustworthy market.

In this paper, we address a similar scenario. Autonomous agents (clients) contract services from the different providers available in the market. The contract (i.e., the SLA) specifies the price of service, the promised QoS and a monetary penalty, paid back to the agent, when the delivered QoS is lower than promised (see [2, 11] for a framework and detailed description of such interactions). Assuming that all clients in a given group are treated equally within a certain period of time, their aggregated feedback estimates the real QoS delivered to them as a group. Reputation information (i.e., aggregated feedback) can therefore replace external monitoring of QoS, and serve to compute the penalty due by the providers.

This leaves the problem of obtaining true feedback from self-interested agents. Reporting the truth is not optimal: by submitting false negative reports, the agents can decrease the reputation of the provider and thus decrease the total price (i.e., the price specified by the SLA minus the reputation-based penalty) paid for service. Misreporting incentives can be counterbalanced by explicit rewards for honest feedback. The reputation mechanism can use side-payments to pay for the agents' reports. As no objective information is available in the market, the *reputation* side-payments are computed by correlating one agent's report

*First author is a student. We thank the anonymous reviewers for helpful remarks and comments.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

with reports submitted by peers. When carefully designed, these payments make it optimal for every agent to reveal the true feedback. Moreover, they can be scaled such that reporting costs or other external benefits obtained from lying (e.g., a price decrease) are accounted for. [12, 11].

However, existing side-payment schemes assume that every agent interacts only once with a service provider (in a given time period), and can therefore submit only one feedback. When a client interacts several times with the same provider, coordinated lying in multiple feedback reports can still be profitable.

Take, for example, a web service providing stock quotes. The client requests the price of a specified stock at a given time: e.g., *"Request the price of an IdealCompany share at 11:00AM on Oct 18, 2005"*. The provider promises an answer within a certain deadline: e.g., *information is delivered within 1 minute from the request time* (i.e., before 11:01 AM for the previous request). The daily SLA specifies (a) the cost of a service request, (b) the promised QoS (i.e., the probability that the request is answered before the deadline) and (c) a penalty if the delivered QoS is lower than promised, for the considered day.

One agent a that constantly monitors (e.g., once every ten minutes) the price of a share, will issue $6 \cdot 8 = 48$ service requests during one 8-working-hours day. This agent is therefore entitled to submit 48 binary feedback reports by the end of the day, where every report signals that one of the requested answers was received or not before the deadline. By misreporting only one feedback (negative instead of positive; the other 47 reports remain unchanged) the agent expects:

- to lose amount ΔSP due to the side-payment for that report;
- to decrease the price of every service invocation by Δp

Existing side-payment schemes make sure that ΔSP is always greater than Δp; however, the total price decrease of agent a is $48\Delta p$, as the price decrease also applies to the other 47 service invocations where a reported true feedback. By a similar argument, a will find it profitable to misreport all other positive reports, and she will end up reporting a clean set of 48 negative reports. Eliminating the lying incentives of agent a requires $\Delta SP \geq 48\Delta p$, which translates into a 480% increase of the original reputation side-payments.[1] The cost of reputation management simply becomes too big.

In this paper we present a new mechanism for rewarding truth-telling in settings where agents have several interactions and can submit several reports within a given time-window. The first idea behind our mechanism is to group all reports of an agent from a given time period into a *feedback set*, and pay the agent only once based on her feedback set. Second, we introduce *CHI-Scores* as a basis for computing the side-payments. The CHI-Score for a feedback set is computed based on a *Chi-square independence test* [18] applied to the feedback set itself, and the set containing the feedback of all other agents. The CHI-Score reflects the correlation between the reports of one agent and those of the entire group. Third, we use CHI-Scores not only to pay the reporter, but also to weigh different feedback sets when computing the reputation of the provider. Finally, we explicitly model external lying incentives (e.g., decrease in price) to make sure that the side-payments based on CHI-Scores makes honest reporting optimal.

Like other agreement-based (between the considered report and the general opinion) side-payment schemes (see Section 2) the mechanism we propose is not incentive compatible in general. Truthful reporting is the optimal strategy only when an agent's beliefs regarding the reports of other agents are entirely determined by her direct experience (i.e., the agent does not have any prior information). This is a common assumption for software agents in multi-agent systems and therefore we particulary address our work to such environments. Moreover, the scheme is intuitive, computationally cheap, robust against small numbers of irrational reporters, and generates significantly lower costs than existing schemes (Section 4).

Section 2 discusses related work, Section 3 formally introduces the setting and our mechanism. An experimental evaluation of the mechanism is presented in Section 4, followed by future work and the conclusion.

2. RELATED WORK

The contribution of reputation information in dynamically scaling the prices of goods and services has been established both theoretically and experimentally. Dellarocas [3] studies an "eBay-like" market where sellers can strategically choose whether to ship or not the good purchased by a buyer. He proves the existence of an equilibrium where the reputation of the seller from the last N transactions determines the present price of the good. Experimental studies of eBay transactions [5, 15] confirm that the reputation of a seller affects the selling price of the auctioned goods.

Jurca and Faltings extend the analysis of reputation-based pricing to service oriented environments where providers with different capabilities coexist [10, 11]. Their results are based on two observations. First, all clients in a given group are probably treated equally[2]. Second, service providers cannot change the quality of the delivered service infinitely often. Therefore, the reputation information obtained by aggregating the feedback from the immediate past offers an accurate estimation of the real QoS delivered by the provider. In [10], client's pay the price corresponding to yesterday's reputation. In [11], a reputation-based SLA allows clients to be refunded at the end of a given time period if the provider's reputation is lower than promised. Both mechanism promote trustworthy behavior, as service providers have the incentive to deliver the promised QoS.

The question of honest reporting has been extensively addressed by Miller et al. in [12]. The mechanism uses the feedback of a future client (called the *rater*) to rate (and compute the payment for) a submitted report. The present report is first used to update a probability distribution for the report of the rater. The payment is then computed by a scoring rule[3] that assesses how the actual report of the rater relates to the probability distribution induced by the rated report. Truthful reporting is a Nash equilibrium in all contexts: it is always optimal for an agent to report the truth

[1] for one given side-payment scheme, multiplication by a constant is the only method for increasing the side-payment loss due to misreporting

[2] note that providers can have several customer groups that are treated differently: e.g., silver/gold/platinium clients

[3] see [1] for a detailed description of scoring rules. The logarithmic function is an example of a scoring rule

given that all other agents report the truth. Moreover, the side-payments can be scaled to account for reporting costs and external benefits obtained from cheating.

Prelec [14] considers a similar approach. His mechanism asks every agent two questions: the first, regarding the observed feedback, the second regarding the feedback other agents are likely to submit. Agents are rewarded if the feedback they report is "unexpectedly common" compared with the overall prediction made by the group about the feedback likely to occur. This *information scoring* makes truthful reporting pareto optimal.

[12] discusses the danger associated with using *agreement* for computing side-payments. When the report of an agent is paid according to how well it "agrees" with the general opinion, the optimal reporting strategy is to adhere to the expected majority. Take an eBay buyer cheated by a seller with a large positive *feedback score*. Despite the negative experience, the buyer continues to believe that most of the future buyers will experience good transactions. Therefore, the report that maximizes the agreement with future feedback is the false positive one.

Agreement-based payments can therefore reward honest feedback only when the agent's expectation of future transactions coincides with her direct experience. In the case of one interaction per agent, this is possible when the outcome of successive interactions are correlated. [7] documents such a scenario where the behavior of service providers is modeled by a Markov chain satisfying certain conditions.

When agents get feedback from several direct interactions, agreement-based payment does encourage truth-telling as long as agents do not have external, trusted third-party information. This assumption is common for multi-agent systems, and has been extensively used in social networks to filter out malicious reports. In TRAVOS [16], the weights of peer reports are computed based on a Bayesian model assessing the agreement between previous recommendations and the direct experience. In CREDENCE [17] malicious reporters are filtered based on a distance measure between the agent's own observations and those of the reporter. Similar techniques are described in [20, 6, 19].

A different approach to honest feedback elicitation is presented in [8] and [13]. They rate feedback for each transaction separately, independent of other transactions. The mechanism punishes both the client and the provider if they have conflicting reports regarding the outcome of the transaction in question. [8] proves that (in any equilibrium) the number of false reports collected by such a mechanism is limited by an upper bound. [13] show an implementation of the mechanism for the Peer-to-Peer environment.

3. USING CHI-SCORES TO REWARD HONEST FEEDBACK

We propose a new side-payment scheme to explicitly reward honest feedback, based on the following ideas: first, feedback reports submitted by one agent about the same provider (within a relevant time window) are treated unitarily as one feedback set of the agent. Second, each feedback set is rated against the set of all other relevant reports available in the market for the considered time period. The CHI-Score assigned to a feedback set is computed using a Chi-square independence test applied to the two sets: the rated feedback set and the reference set containing all other

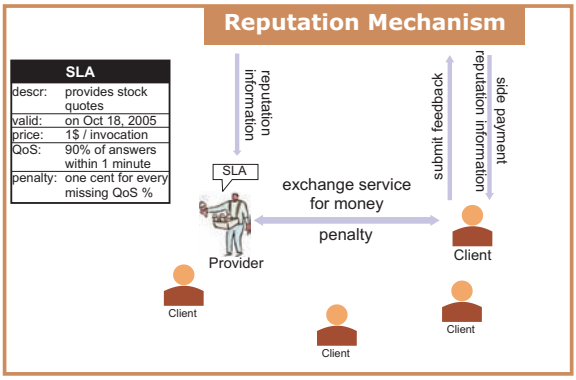


Figure 1: An online market of services.

available feedback reports. Finally, the CHI-Score is used (a) to compute the monetary reward paid to the reporting agent, and (b) to weigh the feedback when computing the reputation information. The external benefits from lying (e.g., price cuts) are factored in so that truth-telling is a Nash equilibrium.

3.1 The Scenario

We consider an online market pictured in Figure 1 where service providers repeatedly offer the same service (having the same functional interface) to the interested clients (autonomous agents), in exchange for money. Agents choose the desired providers based on the Service Level Agreement (SLA) advertised by the latter. The SLA specifies:

- the description of the service
- a period of validity,
- the quality parameters of the delivered service (i.e., the QoS),
- the price for every service invocation,
- the monetary penalty to be refunded when the actual QoS is lower than the advertised one.

A precise definition of such SLAs is presented in [11]. A practical framework supporting such interactions is documented in [2].

We assume that clients have two degrees of satisfaction: they perceive either *high* or *low* quality service. For example, high quality service means that the client obtains the answer before the deadline. QoS can therefore be defined as the probability that one service request is answered successfully (i.e., the client perceives high quality).

The QoS actually delivered to clients depends on the fixed infrastructure (e.g., computing power, bandwidth, etc) and on the strategic choices of the service provider (e.g., number of accepted requests, allocation of resources, etc). The provider can anticipate the implications of his[4] decisions on the perceptions of the clients; therefore we say that the provider can choose (within the limits imposed by the infrastructure) the QoS he is actually going to deliver.

[4]as a convention, we will refer to the provider as "he" and to the client as "she"

A service provider cannot change his decisions infinitely often. We split time into equidistant periods (indexed according to t) such that all services delivered within one period will share the same QoS. We also assume that a large enough group of clients will request the service in any period, and that their observations will be independently identically distributed according to the delivered QoS. Note that a provider can have several customers groups (i.e., silver/gold/platinium) as long as all clients in a given group are treated identically within a given period of time.

An independent reputation mechanism requests binary feedback (1 for a high quality service, 0 otherwise) about every interaction. The reputation of a provider is computed as the average satisfaction rate of the clients for a given period of time. When clients report truthfully, reputation information accurately estimates the real QoS delivered by the provider in that period.

At the end of each period, the reputation mechanism publishes the reputation of every provider, and service providers are expected to refund every client the monetary penalty specified in the SLA. The penalty is positive if the delivered QoS (i.e., the reputation) is lower than advertised, and zero otherwise. When the penalty is large enough, [11] proves that rational service providers keep their promises and the market is efficient.

The feedback submitted during one period has no influence over future periods of time. We can therefore separately analyze the reporting incentives of rational clients for each period. Let us consider the time period t where one provider advertises the SLA, $sla_t = \big[p_t, \bar{q}_t, \lambda(\bar{q}_t, R_t)\big]$ where p_t is the price of one service invocation, $\bar{q}_t$ is the advertised QoS, and $\lambda(\bar{q}_t, R_t)$ is the penalty refunded to the client at the end of the period if the reputation R_t is lower than $\bar{q}_t$. For simplicity, we take a linear penalty function:

$$\lambda(\bar{q}_t, R_t) = \max\big(C \cdot p_t \cdot (\bar{q}_t - R_t), 0\big);$$

where C is a constant. We assume that N_t agents require the service from the provider, each of them issuing M service requests during period t. We use the notation a_i, $i = 1, \ldots, N_t$ to refer to one client agent, and a_{-i} to refer to the group of all agents except a_i.

By the end of the period, every agent will have M binary feedback reports, each describing the quality of one service request. Let $o_i = \{s_1, s_2, \ldots, s_M\}, s_j \in \{0, 1\}$ be the feedback set *observed* by agent a_i. As the order of the reports is not relevant, the feedback set can be represented by one integer value counting the number of positive reports: $o_i = \sum_{j=1}^{M} s_j$.

Client agents do not necessarily report true feedback to the reputation mechanism. We denote by $r_i \in \{0, 1, \ldots, M\}$ the feedback set *reported* by agent a_i. Similarly, $r_{-i} \in \{0, 1, \ldots (N_t - 1)M\}$ denotes the union of the feedback sets of the agents a_{-i}.

3.2 Computing the CHI-Scores

At the end of the time period, the reputation mechanism obtains N feedback sets, each containing M binary reports. Based on this information, the reputation mechanism will (a) compute the reputation of the provider for the past period, and (b) reward agents for their feedback such that it is in their best interest to report the truth. For both task, the reputation mechanism will use the CHI-Scores of the feedback sets.

The CHI-Score of the feedback set r_i belonging to the agent a_i is computed based on a Chi-square independence test applied to the sets r_i and r_{-i}. The Chi-square independence test is a well established statistical tool that helps decide if the results of an inquiry are independent from the sample group. Consider, for example, the vacation preferences (seaside or mountains) of the citizens of Switzerland and Italy. m_S out of the N_S inquired Swiss citizens prefers a vacation in the mountains, compared to m_I out of the N_I inquired Italian citizens. Applied to this data, a Chi-square independence test allows us to decide if the vacation preferences of Swiss citizens are really different from those of the Italian citizens.

We use the Chi-square independence test in a similar manner: for every agent, we try to establish if her feedback set is really different from the one of the rest of the agents. However, instead of outputting a binary decision (i.e., reporting pattern is different or not), we modify the test to output a continuous value: the CHI-Score of the feedback set.

We define, χ^i_{Sc}, the CHI-Score of the feedback set r_i as:

$$\begin{aligned} \chi_i^{Sc}(r_i) &= \chi_i^{Sc} = 1 - chi2cdf(\chi_i^2, 1); \\ \chi_i^2 &= \frac{\big(r_i - r_{-i}/(N_t-1)\big)^2}{r_{-i}/(N_t-1)} \\ &\quad + \frac{\big(M - r_i - M + r_{-i}/(N_t-1)\big)^2}{M - r_{-i}/(N_t-1)} \\ &= \frac{M\big(r_i(N_t-1) - r_{-i}\big)^2}{r_{-i}\big(M(N_t-1) - r_{-i}\big)} \end{aligned} \tag{1}$$

where χ_i^2 is the Chi-square value associated with the Chi-square independence test applied to the sets r_i and r_{-i}, and $chi2cdf(\cdot, 1)$ is the chi-square cumulative distribution function with one degree of freedom.

Intuitively, the CHI-Score of r_i is equal to the probability that honest reports of agent a_i could be even further away from r_{-i} than r_i is. The CHI-Score takes values between 0 and 1; the higher the CHI-Score, the "closer" are the sets r_i and r_{-i} and therefore the lower the probability that a_i's reports are distributed differently than the reports of the rest of the agents. We take the CHI-Score to be a direct measure of the agent's honesty, and therefore use it to compute the agent's side-payment, and the weight of the feedback set when computing the reputation of the provider.

3.3 Incentives for Honest Reporting

The reputation of the provider is computed as a weighted average of the feedback sets submitted by the agents:

$$R_t = \frac{1}{M} \frac{\sum_{j=1}^{N_t} \chi_j^{Sc} r_j}{\sum_{j=1}^{N_t} \chi_j^{Sc}}; \tag{2}$$

PROPOSITION 1. *When agents honestly report feedback, the reputation R_t of the provider accurately approximates the actual QoS delivered in the time period t.*

SKETCH OF PROOF. Let q be the real QoS delivered by the service provider during time period t. For all agents a_i, the number of positive reports in the observed feedback set o_i will be distributed according to the binomial distribution with parameters M and q. As all agents report truthfully, the number of positive reports in the set r_{-i} follows a binomial distribution with parameters $(N_t - 1)M$ and q. The

expected CHI-Score of agent a_i can be computed from the distributions of r_i and r_{-i}, and is equal for all agents. Since all weights in Equation (2) have the same values, the expected value of the reputation R_t, is equal to the expected ratio of positive reports submitted by all agents: i.e., the delivered QoS, q. ■

Proposition 1 proves that the use of CHI-Scores to weigh feedback sets does not bias the value of the reputation information (i.e., reputation accurately estimates delivered QoS). Therefore, the efficiency results of [11] still apply.

The final price agents pay for the M service invocations in period t is:

$$\begin{aligned} p &= M \cdot \big(p_t - \lambda(\bar{q}_t, R_t)\big) \\ &= M \cdot p_t \Big(1 - max\big(C(\bar{q}_t - R_t), 0\big)\Big); \end{aligned} \tag{3}$$

The price directly depends on the reputation of the provider. By reporting only negative feedback, rational clients decrease the reputation of the provider as much as possible, and thus minimize their payments.

Side-payments for feedback reports can make truth-telling rational. The payment proposed by our mechanism is directly proportional to the CHI-Score of the feedback set submitted by an agent. When all other agents report the truth (hence r_{-i} contains true feedback), a_i expects that r_{-i} follows a similar distribution as o_i, her direct observations. Therefore, the expected CHI-Score of r_i is maximized when $r_i = o_i$: any misreporting increases the "distance" between r_i and r_{-i}. The expected reputation side-payment is therefore maximized when a_i reports honestly; this makes truthful reporting a Nash equilibrium.

The external reward obtained from lying (i.e., the price decrease) imposes supplementary constraints on the reputation side-payments. Truthful reporting remains a Nash Equilibrium only when the price decrease obtained by lying is smaller than the expected side-payment loss. Let $SP(o_i, r_i)$ be the expected side-payment obtained by agent a_i when reporting r_i instead of the observed feedback o_i. $\Delta SP(o_i, r_i) = SP(o_i, o_i) - SP(o_i, r_i)$, $r_i \neq o_i$ is the expected side-payment loss due to misreporting. If $\Delta p(o_i, r_i)$ is the corresponding price decrease, we must have:

$$\Delta SP(o_i, r_i) \geq \Delta p(o_i, r_i); \forall r_i \neq o_i; \tag{4}$$

To compute the potential price decrease obtained by lying, let us consider the effect of misreporting on the reputation of the provider. By reporting r_i instead of o_i, the reputation of the provider becomes:

$$R_t^* = \frac{1}{M} \frac{\chi_i^{Sc}(r_i) r_i + \sum_{j \neq i} \chi_j^{Sc} r_j}{\chi_i^{Sc}(r_i) + \sum_{j \neq i} \chi_j^{Sc}}$$

When N_t is large enough, the misreporting of a_i does not affect the CHI-Scores of other agents, and therefore the expected decrease in reputation can be approximated by:

$$R_t - R_t^* \simeq \frac{o_i \chi_i^{Sc}(o_i) - r_i \chi_i^{Sc}(r_i)}{M \cdot N_t \cdot \chi_i^{Sc}(o_i)}$$

Assuming providers advertise the intended QoS, (i.e., $\bar{q}_t \simeq R_t$), the expected price decrease (due to misreporting) for the M service invocations is:

$$\Delta p(o_i, r_i) = C \cdot p_t \frac{o_i \chi_i^{Sc}(o_i) - r_i \chi_i^{Sc}(r_i)}{N_t \cdot \chi_i^{Sc}(o_i)}$$

Computing the expected side-payment loss due to misreporting requires an explicit model for the beliefs of the agents. We assume agents do not have any prior information about the service provider. Having observed o_i successful service invocations during one time period, the best estimate a_i has about the real QoS delivered by the provider is o_i/M.[5]

We denote by $\phi(r_i, r_{-i})$ the actual payment received by a_i when she reports the feedback set r_i, and the rest of the agents report r_{-i}. Let $\phi(r_i, r_{-i}) = K \cdot \chi_i^{Sc}(r_i)$, where K is a constant, greater than 0. The expected payment of a_i can now be computed by considering the probability distribution of r_{-i} given the real observation o_i:

$$SP(o_i, r_i) = \sum_{r_{-i}=0}^{(N_t-1)M} Pr[r_{-i}|o_i] \phi(r_i, r_{-i}); \tag{5}$$

where $Pr[r_{-i}|o_i]$ follows a binomial distribution for $(N_t - 1)M$ trials with probability o_i/M.

PROPOSITION 2. *Truthful reporting is a Nash Equilibrium when:*

$$K \geq \max_{r_i \neq o_i} \frac{\Delta p(o_i, r_i)}{\Delta SP(o_i, r_i | K = 1)};$$

where $\Delta SP(o_i, r_i|K = 1)$ is the expected side-payment loss for reporting r_i instead of o_i, computed for $K = 1$.

SKETCH OF PROOF. The proof has two parts. First, we show that $\Delta SP(o_i, r_i) \geq 0$ for all $o_i \neq r_i$, and $K > 0$. Then, given the lower bound for K, we show that the price decrease obtained by misreporting is always smaller than the loss caused by the side-payments.

Note that $SP(o_i, r_i) = K \cdot SP(o_i, r_i|K = 1)$ where:

$$SP(o_i, r_i|K = 1) = \sum_{r_{-i}=0}^{(N_t-1)M} Pr[r_{-i}|o_i] \chi_i^{Sc}(r_i, r_{-i});$$

is the expected CHI-Score of the report r_i given the observation o_i. $SP(o_i, r_i|K = 1)$ can be seen as the inter-correlation of two signals: $Pr[x|o_i]$ and $\chi_i^{Sc}(x|r_i)$. When all other agents report the truth, both signals have only one maximum (for $x = o_i(N_t - 1)$ and $x = r_i(N_t - 1)$ respectively) and are (approximately) symmetrically decreasing around the maximum point. The maximum correlation is reached when the maximum points of the two signals are aligned, i.e., $o_i = r_i$, and therefore $SP(o_i, o_i|K = 1) \geq SP(o_i, r_i|K = 1)$ for all $o_i \neq r_i$.

When K satisfies the hypothesis, we have:

$$\begin{aligned} \Delta SP(o_i, r_i) &= K\Big(\Delta SP(o_i, r_i|K = 1)\Big) \\ &\geq \Delta p(o_i, r_i) \text{ for all } o_i \neq r_i; \end{aligned}$$

Therefore it is optimal for a_i to report honestly, given that all other agents also report the truth. This makes truthful reporting a Nash Equilibrium. ■

3.4 Extension to N-ary Feedback

Binary feedback about a transaction can sometime cause the loss of valuable information. For agents using the stock-quote web service, late information (i.e., received after the

[5]for $o_i = 0$ and $o_i = M$, the actual estimate of the QoS is actually ε respectively $1 - \varepsilon$ for ε small

deadline) could still be more valuable than no information at all. One can therefore imagine a finer-grained reporting system where the feedback also gives some information about the delay of the answer: one feedback is submitted when the request was answered on time, another when the answer was no more than 1 minute late, etc.

In general, one can consider an N-ary reporting scheme where feedback can take any value from the predefined set $\mathcal{S} = \{s_1, \ldots s_S\}$ of cardinality S. Agents share a common ontology about the elements of $\mathcal{S}$ (i.e., they interpret feedback in the same way). The feedback set observed by an agent becomes a vector $o_i = (o_i^1, \ldots, o_i^S)$ where o_i^j is the number of service requests where a_i perceived the feedback value s_j. Similarly, the reported feedback set of an agent (i.e., r_i), as well as the feedback set of all other agents (i.e., r_{-i}) are vectors with S elements. The QoS and the reputation of the provider become probability distributions over the possible feedback values: $\bar{q}_t = (q_{s_1}, \ldots, q_{s_S}) \in [0,1]^S$ such that $\sum_{j=1}^{S} q_{s_j} = 1$.

The extension of the mechanism to N-ary feedback is straightforward: the CHI-scores are computed in the same way and become:

$$\chi_i^{Sc}(r_i) = \chi_i^{Sc} = 1 - chi2cdf(\chi_i^2, S-1);$$

$$\chi_i^2 = \sum_{j=1}^{S} \frac{\left(r_i^j(N_t-1) - r_{-i}^j\right)^2}{(N_t-1)r_{-i}^j}$$

where the Chi-square value is a summation over the dimensions of the feedback set, and the chi-square cumulative distribution function has $S-1$ degrees of freedom. The qualitative results of Proposition 1 and Proposition 2 remain the same.

4. EXPERIMENTAL EVALUATION

In this section we experimentally evaluate the mechanism described above, using several criteria. First, we study the accuracy of the reputation information in the presence of irrational clients that misreport feedback regardless of the monetary loss they incur. Second, we investigate the robustness of the honest-reporting incentives in the presence of such irrational agents. Finally, we compare the cost of our mechanism against existing side-payment schemes.

We consider the same example presented in the introduction, where a web service delivers stock quotes to interested agents. Every morning, the web service advertises a SLA specifying the price of a service request, the promised QoS (as a probability of delivering the information within a given deadline), and the coefficient C of the linear reputation-based penalty function λ. Clients subscribe to the service before the opening time of the market. Each subscription specifies (a) the code of the desired share and (b) the set of M time points when an information update is requested. An example of a subscription is: *"Request the price of the share X every 15 minutes starting at 9:00 AM and ending at 5:PM"*. Every subscription generates M service invocations.

The real QoS delivered to clients depends on the infrastructure of the provider and on the number of accepted requests. The provider can choose the intended QoS by limiting the total number of subscribers for a given day. When the stock market opens, all requests have already been registered and the delivered QoS remains fixed for the entire day. The reputation information is computed by the reputation mechanism at the end of every day, and all penalties are paid to the clients before the beginning of the next day.

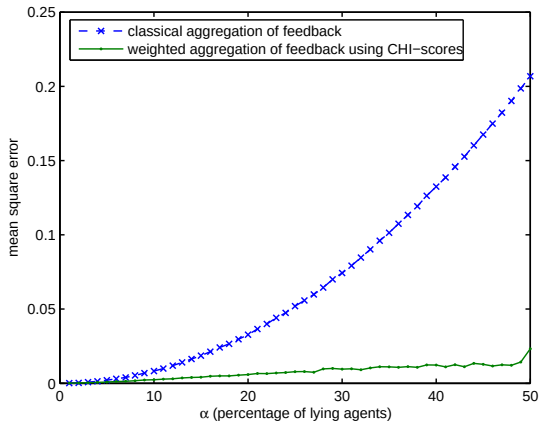


Figure 2: Mean square error of reputation information.

We take one day and one web service. We assume the total number of clients is N, and that all client requests are similar: the price of a share is requested every m minutes. The total number of requests per client during one day is therefore $M = 8hours \times 60min/m$.

As a first experiment, we study the accuracy of the reputation mechanism when irrational clients lie, despite the guarantees provided by Proposition 2. Figure 2 plots the mean square error between the expected and real reputation information when α percent of the agents lie (submit only negative feedback). The performance of our mechanism is compared against a classical (un-weighted) aggregation scheme for $N = 100$ agents, $M = 40$ interactions per agent, and the real QoS equal to 0.9. The average is taken over 100 runs. As it can be seen, using CHI-Scores to weigh feedback sets dramatically improves the robustness of the mechanism against lying agents.

Irrational lying also affects the honest reporting incentives for the rational agents. The most extreme lying strategy (worst-case scenario) is when irrational agents constantly submit negative feedback. Depending on the number of irrational reporters, rational agents find it optimal to slightly down-bias their real report. Figure 3 plots the probability that a rational agent will report something else than the true observation as the number of lying agents increases. For this experiment we took the same delivered quality value (i.e QoS $= 0.9$) but we considered only $M = 20$ interactions per client. The total number of clients varied from $N = 100$ to $N = 400$. The graph reveals an almost instantaneous switch from the truthful strategy to some lying strategy. For up to 2.5% liars, truth-telling is optimal; from that point onward, every supplementary liar makes down-biasing rational for an increasing set of observations (some observations are still reported truthfully, but the probability of such observations appearing decreases rapidly with the number of liars). With 3.5% liars, down-biassing becomes rational for all observations. This low threshold is mostly due to the extreme lying strategy we have considered. In reality, dishonest agents will lie only once every L reports, and consequently, the toler-

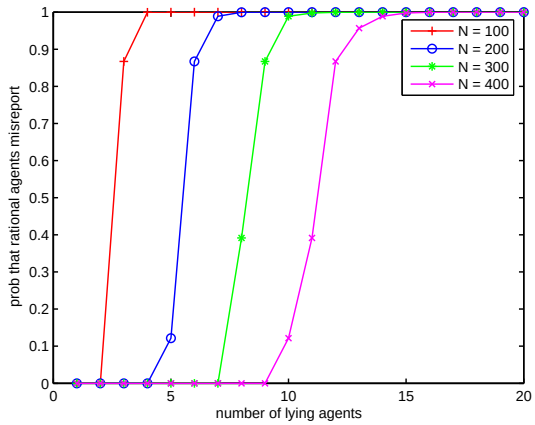


Figure 3: Misreporting incentives in the presence of irrational liars.

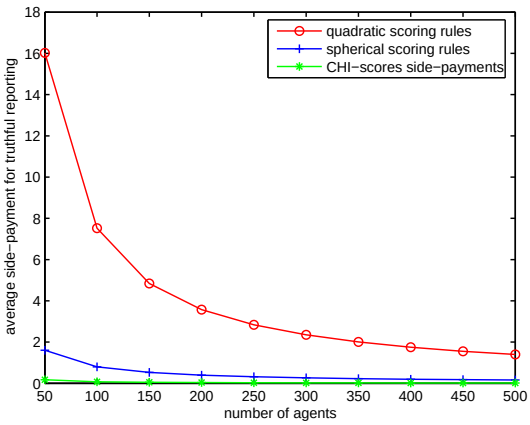


Figure 5: Average side-payment per client per transaction for truthful reporting.

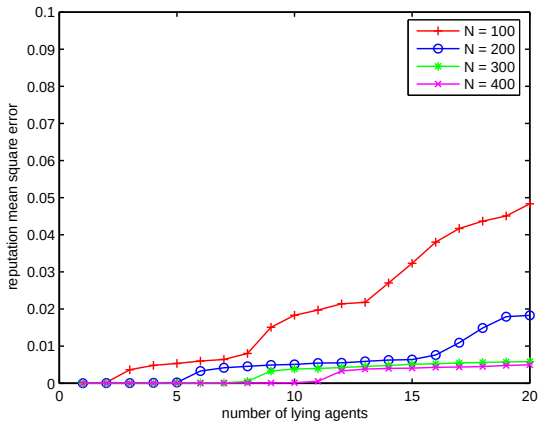


Figure 4: Reputation mean square error when rational agents are aware of the presence of irrational reporters.

ance threshold will be multiplied by L.

Even when truth-telling is not optimal, the submitted reports are not too far from the truth, and thus do not greatly distort reputation information. Figure 4 plots the corresponding mean square error for the reputation value when rational agents are aware of the presence of irrational reporters and consequently adapt their reports.

The reputation side-payments contribute substantially to the global cost of the reputation mechanism. In theory, all side-payment mechanisms can be made budget balanced by asking fixed (or periodically updated) fees from the client agents (i.e., clients also pay for using the reputation mechanism). However, none of the existing online markets charges clients for access to reputation information. The cost of reputation management is absorbed by the market and later distributed to service providers. Moreover, Dellarocas [4] theoretically proves that increasing amounts of truthful feedback helps rational service providers cut down costs (e.g., advertising costs) and thus justifies the providers' contribution to reputation management costs.

It therefore becomes an important objective for mechanism designers to minimize such costs, by minimizing the side-payments. We compare the costs of our mechanism with existing side-payment schemes based on scoring rules [12, 11]. We consider two of the best-known scoring rules: the spherical and quadratical scoring rule. The resulting side-payments are scaled (by addition and multiplication with a constant) to the minimum positive side-payments that counterbalance the price decrease obtained from misreporting. We take the average expected side-payment for truthful reporting, $\bar{SP} = \sum_{o_i} SP(o_i, o_i)$, per client, per transaction. Figure 5 plots the averages for the three mechanisms when $M = 20$ interactions per day per client, the price of service and the penalty constant are unitary (i.e., $p_t = C = 1$) and N varies from 50 to 500 clients per day. The side-payments based on CHI-Scores are one order of magnitude lower than the cheapest of the side-payments based on the considered scoring rules. When one service invocation costs 1 dollar, the average side-payment based on CHI-Scores is 7.5 cents for $N = 50$ clients and 1.3 cents for $N = 500$ clients. In the same conditions, the average side-payment computed using the spherical scoring rule is 79 respectively 15 cents.

5. DISCUSSION AND FUTURE WORK

One of the important assumptions behind our mechanism is that all customers in a certain group receive the same QoS within one period of time. In real systems, however, request arrival rates as well as other real-time factors might trigger non-negligible short term fluctuations in the service quality. Certain clients will thus receive better service than others.

However, most of these fluctuations will look random for the clients. Even though quality observations are no longer independently identically distributed, one client, due to the limited number of observations, will not be able to model the stochastic behavior of the service provider. Thus, the clients' beliefs remain as described in this paper, and the CHI-Scores encourage truthful reporting.

On the other hand, easily-identifiable QoS fluctuations patterns should not exist in a rational market. When ser-

vice providers can identify the clients that are likely to receive higher quality, it makes sense to treat those clients as part of a different, "preferred" group. Providers can thus offer different conditions (QoS and price) to each group, and clients within the same group continue to be treated "approximately equally". As a last resort, the mechanism designer can of course solve the problem by shortening the length of the time period.

A related problem comes from subjective interpretations of quality. Different agents might have different criteria for high quality, and this will affect our scoring mechanism. For this reason, our paper targets specific markets where autonomous agents request (web-)services. Interactions in such environments are usually regulated by SLA's that consider objective measures of QoS (such as response time, availability, etc). Dealing with subjective perceptions is a challenging direction for future work.

Truthful reporting is guaranteed as a Nash equilibrium by Proposition 2. However, honest reporting is not the only equilibrium strategy: e.g., reporting only negative experiences is also a Nash equilibrium. The problem of multiple reporting equilibria has been formally identified in [9]. Fortunately, there is a simple way of enforcing truth-telling by using a small number of trusted, third-party reports.

As future work, we plan to extend this mechanism along several dimensions. First, we plan to increase the robustness of the mechanism against irrational liars. In a future version we envisage to rate the feedback set of one agent against the reports of a fraction (not all) of the remaining agents. By carefully choosing the "reference" set for every agent, we plan to increase the threshold for the percentage of tolerated irrational reports.

Second, we plan to extend the set of contexts where the mechanism can be successfully used, by allowing some limited amounts of external information. As before, we expect to be able to group agents according to their prior information, and rate the feedback set of an agent against the reports submitted by other agents in the same group.

Third, we plan to investigate other distance metrics for scoring set of reports.

Last, but not least, we plan to address the problem of collusion. In the present mechanism, two or more agents can benefit by coordinating their false reports. To solve this problem we plan using the idea in [11] where a small number of trusted reports (i.e., reports that are true with high probability) can deter collusion.

6. CONCLUSION

When client agents can submit several feedback reports within a certain period of time, existing payment schemes that reward honest feedback are either inefficient, or extremely expensive. In this paper we present a novel side-payment scheme based on CHI-Scores. We consider the entire feedback set of an agent, not her individual reports. The CHI-Score of a feedback set results from a Chi-square independence test and reflects the correlation between the reports of one agent and those of the rest of the group. The score is used to weigh feedback sets when computing reputation information, and to scale the reputation side-payments. We prove that such monetary rewards make truth-telling rational, and show how external benefits obtained from lying can be counterbalanced. The mechanism can be extended to finer-grained feedback systems, and generates significantly lower costs. Experimental results show that a small number of irrational liars, does not significantly degrade the performance of the mechanism.

7. REFERENCES

[1] R. Cooke. *Experts in Uncertainity: Opinion and Subjective Probability in Science*. Oxford University Press: New York, 1991.

[2] A. Dan, D. Davis, R. Kearney, A. Keller, R. King, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer, and A. Youseff. Web services on demand: WSLA-driven automated management. *IBM Systems Journal*, 43(1):136–158, 2004.

[3] C. Dellarocas. Sanctioning Reputation Mechanisms in Online Trading Environments with Moral Hazard. MIT Sloan Working Paper #4297-03, 2004.

[4] C. Dellarocas. Strategic Manipulation of Internet Opinion Forums: Implications for Consumers and Firms. Working Paper, 2005.

[5] D. Houser and J. Wooders. Reputation in Auctions: Theory and Evidence from eBay. *Journal of Economics and Management Strategy*, forthcoming, 2005.

[6] R. Ismail and A. Jøsang. The Beta Reputation System. In *Proceedings of the 15th Bled Conf. on E-Commerce*, 2002.

[7] R. Jurca and B. Faltings. An Incentive-Compatible Reputation Mechanism. In *Proceedings of the IEEE Conference on E-Commerce*, Newport Beach, CA, USA, 2003.

[8] R. Jurca and B. Faltings. "CONFESS". Eliciting Honest Feedback without Independent Verification Authorities. In *Sixth International Workshop on Agent Mediated Electronic Commerce (AMEC VI 2004)*, New York, USA, July 19 2004.

[9] R. Jurca and B. Faltings. Enforcing Truthful Strategies in Incentive Compatible Reputation Mechanisms. In *Internet and Network Economics*, volume 3828 of *LNCS*, pages 268 – 277. 2005.

[10] R. Jurca and B. Faltings. Reputation-based Pricing of P2P Services. In *Proceedings of the Wokshop on Economics of P2P Systems*, Philadelphia, USA, 2005.

[11] R. Jurca and B. Faltings. Reputation-based Service Level Agreements for Web Services. In *Service Oriented Computing (ICSOC - 2005)*, volume 3826 of *LNCS*, pages 396 – 409. 2005.

[12] N. Miller, P. Resnick, and R. Zeckhauser. Eliciting Informative Feedback: The Peer-Prediction Method. Forthcoming in Management Science, 2005.

[13] T. G. Papaioannou and G. D. Stamoulis. An Incentives' Mechanism Promoting Truthful Feedback in Peer-to-Peer Systems. In *Proceedings of IEEE/ACM CCGRID 2005*, 2005.

[14] D. Prelec. A bayesian truth serum for subjective data. *Science*, 306(5695):462–466, 2004.

[15] P. Resnick, R. Zeckhauser, J. Swanson, and K. Lockwood. The Value of Reputation on eBay: A Controlled Experiment. *Experimental Economics*, forthcoming, 2005.

[16] L. Teacy, J. Patel, N. Jennings, and M. Luck. Coping with Inaccurate Reputation Sources: Experimental Analysis of a Probabilistic Trust Model. In *Proceedings of AAMAS*, Utrecht, The Netherlands, 2005.

[17] K. Walsh and E. Sirer. Fighting Peer-to-Peer SPAM and Decoys with Object Reputation. In *Proceedings of P2PECON*, Philadelphia, USA, 2005.

[18] N. A. Weiss. *Elementary Statistics*. Addison-Wesley Publishing Company, 1996.

[19] A. Whitby, A. Jøsang, and J. Indulska. Filtering out Unfair Ratings in Bayesian Reputation Systems. In *Proceedings of the 7th Intl. Workshop on Trust in Agent Societies*, 2004.

[20] B. Yu and M. Singh. Detecting Deception in Reputation Management. In *Proceedings of the AAMAS*, Melbourne, Australia, 2003.

Learning Trust Strategies in Reputation Exchange Networks

Karen K. Fullam and K. Suzanne Barber
Laboratory for Intelligent Processes and Systems
The University of Texas at Austin
1 University Station Stop C5000, Austin, TX, 78712
+1-512-471-5350

{kfullam, barber}@lips.utexas.edu

ABSTRACT

An agent's *trust decision strategy* consists of the agent's policies for making trust-related decisions, such as who to trust, how trustworthy to be, what reputations to believe, and when to tell truthful reputations. In reputation exchange networks, learning trust decision strategies is complex, compared to non-reputation-communicating systems. When potential partners may exchange reputation information about an agent, the agent's interactions with one partner are no longer independent from interactions with another; partners may tell each other about their experiences with the agent, influencing future behavior. This research enumerates the types of decisions an agent faces in reputation exchange networks, explains the interdependencies between these decisions, and correlates rewards to each decision. Experimental results using the Agent Reputation and Trust (ART) Testbed demonstrate the success of strategy-learning agents over agents employing naive strategies. The variation in performance of reputation-based learning vs. experience-based learning over different opponents illustrates the need to dynamically determine when to utilize reputations vs. experience in making trust decisions.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *Intelligent Agents, Multiagent Systems*

General Terms

Algorithms, Experimentation.

Keywords

Trust, Reputation, Multi-Agent Systems, ART Testbed.

1. Introduction

When an agent does not have the resources to accomplish its goals alone, it may seek needed resources from other agents in a multi-agent system. These resources are acquired through transactions, which expose at least one party to risk, since a transaction partner might not follow through on the (implicitly or explicitly) agreed upon contract, either by choice or inability. An agent can protect itself against this risk of transaction failure by learning estimates of partner trustworthiness.

Learning trust from transaction experiences [5, 12, 16] is advantageous when agents have opportunities for numerous repeated interactions. When the outcome of interactions are observable, transaction experiences provide a trust-strategy-learning agent with trustworthiness feedback that is certain. Unfortunately, basing trust on transaction experiences means risk exposure is unavoidable; interactions must take place while partners' trustworthiness characteristics are being learned [7].

Reputation exchange is useful for quickly learning trustworthiness characteristics of other agents [14]. Adapted from [8], a reputation is a (not necessarily truthful) communication from one agent to another about the sender's trust in a third subject-agent. A multi-agent system whose communication protocols permit the exchange of reputations [11, 17, 18, 20, 21] is advantageous in systems with large populations and sparse transactions. Further, reputation exchange reduces an agent's risk exposure; an agent risks only the price of reputations it purchases, rather than the value of resources exchanged in a potential transaction. Agents entering a multi-agent system can quickly learn trust models by requesting reputations from more knowledgeable agents.

An agent's *trust decision strategy* consists of the agent's policies for making trust-related decisions with the aim of achieving greatest benefit. The simplest trust decision strategy, in systems without reputation exchange, encompasses decisions regarding both 1) who to trust (to minimize risk of failed transactions), and 2) how trustworthy to be (to exploit others). But the advantage of agents communicating reputation information adds another dimension to an agent's trust decision strategy; in reputation exchange networks, an agent must make additional decisions regarding 1) which reputations to believe (to better choose who to trust) and 2) how truthful to be in communicating reputations to others (to manipulate what others think about who to trust).

Reinforcement learning has been successfully applied to simple trust decision strategies for deciding who to trust—and how trustworthy to be—in agent transactions [3, 13]. This research seeks to apply similar techniques to help agents learn more

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

comprehensive trust decision strategies required in networks of agents exchanging reputation information. However, in reputation exchange networks, the complexity of the learning problem increases dramatically, compared to systems in which agents do not communicate reputation information. The agent must make more trust-related decisions, regarding not only who to trust and how trustworthy to be, but also regarding which reputation information to believe and how truthful to be when communicating reputations to others. Compounding the problem, the effects of these decisions may not be immediately observable, since the agent cannot easily gauge the nature and extent of reputation exchange occurring among its potential partners. Further, when potential partners may exchange reputation information about an agent, the agent's interactions with one partner are no longer independent from interactions with another; partners may tell each other about experiences with the agent, influencing how each behaves with the agent in the future.

Toward the goal of enabling agents to learn comprehensive trust decision strategies in reputation exchange networks, this research first enumerates the types of decisions an agent faces in systems with reputation exchange. Next, using the Agent Reputation and Trust (ART) Testbed [10] as a case study, the interdependencies between these decisions are explained. This research employs a learning approach for each decision; by establishing assumptions to ease interdependencies, rewards are correlated to each decision type. Experimental results demonstrate the success of strategy-learning agents over agents employing naive strategies. The variation in performance of reputation-based learning vs. experience-based learning over different opponents illustrates why agents must be able to dynamically determine when to utilize reputations vs. experience in making trust decisions.

2. Trust Decision Strategies

A trust decision strategy describes an agent's choices regarding four trust-related decision types, defined according to agent role (trustee or truster) and transaction (fundamental or reputation), as shown in Figure 1. Fundamental transactions are the basic transactions driving the need for trust modeling in the multi-agent system. Reputation transactions are distinct from fundamental transactions because the outcome of reputation transactions directly impacts decisions regarding fundamental transactions (this impact is shown later). An agent's trust-based decisions are:

1) *Should I trust?* For each potential fundamental transaction, the agent must decide whether to participate in—and expose itself to the risk of— the transaction.
2) *How trustworthy should I be?* For each agreed upon fundamental transaction, the agent must identify the degree to which cheating or honesty yields the greatest benefit.
3) *Should I believe this reputation?* For each potential reputation transaction, the agent must decide whether to conduct the transaction, then use the reputation to determine whether it should trust regarding fundamental transactions.
4) *How accurate should the reputation I tell be?* For each agreed upon reputation transaction, the agent must identify its level of truthfulness in communicating a reputation. The agent can only control the accuracy of reputations it communicates to the extent of its own trust model accuracy.

		Transaction	
		Reputation	Fundamental
Agent Role	Trustee	How accurate should the reputation be?	How trustworthy should I be?
	Truster	Should I believe this reputation?	Should I trust?

Figure 1. Trust decision types by transaction and agent role.

How the agent makes these decisions depends, in part, on the characteristics of the transaction. If the transaction protocol calls for sequential action between the two partners (for example, payment then delivery), the first partner to act serves as truster, and the second as trustee. If the transaction protocol dictates simultaneous action, both partners concurrently act as both trustee and truster. Future decisions are also impacted by the observability of a transactions outcome. For example, if an agent receives a payment as the transaction outcome, most likely it can tell if the payment is satisfactory. However, an agent may be unable to tell whether a reputation transaction is successful (the received reputation is accurate) if it has little knowledge about the trustworthiness of the reputation's subject agent.

2.1 The Complexity of Learning Trust Decision Strategies

This research proposes a method by which agents can learn trust decision strategies using a q-learning technique. Q-learning [15] is a well-known tool for learning best decision-making strategies by associating actions with expected rewards. Several researchers [3, 4, 9, 13, 19] have employed reinforcement learning techniques to discover strategies for trust decisions. However, these trust decisions only relate to fundamental transactions, as in the Prisoner's Dilemma [2]. Learning trust decision strategies in reputation-transaction systems is significantly more difficult, since in these systems, fundamental transaction decisions are not independent from each other and from reputation transaction decisions. An agent's fundamental transactions with one partner may influence its fundamental transactions with another, if its two partners are able to communicate reputation information. Learning an agent's trust decision strategy is also complex because multiple decisions can affect common results. For example, falsely spreading negative reputations about a trustworthy partner might cause other agents to trust the lying agent while isolating the falsely accused agent. Similarly, falsely spreading positive reputations about untrustworthy partners might hurt those agents receiving the reputations, thus improving the standing of the lying agent. Or, an agent might perform truthfully in hopes of receiving better treatment from partner agents, yet truthful behavior might benefit the agent's partners too much.

Considering the interdependent nature of these trust-related decisions, an agent attempting to learn the best combination of decisions faces a complexity problem. When making decisions about "should I trust?" an agent faces $O\left(2^{ae}\right)$ decision combinations from which to choose, assuming that choosing to trust is a binary decision, a represents the number of agents in the system, and e represents the number of possible transaction categories for which agents might have different expertise. When deciding "how trustworthy should I be?" an agent has n^p decision combinations, where n describes a number of possible, discrete degrees of fundamental-transaction trustworthiness and p describes the number of fundamental transaction requests

received from other agents. Since p can be as large as $(a\text{-}1)e$, the complexity of this decision is $O\left(n^{ae}\right)$.

When deciding "should I believe this reputation?" an agent faces $O\left(2^{a^2e}\right)$ decisions, assuming that choosing to believe a reputation is a binary decision, a (signifying the number of agents in the system) represents the number of reputation providers and the number of subject agents about whom reputations may be provided, and e (the number of possible transaction expertise categories) represents the number of categories for which each subject agent's trustworthiness can be questioned. When deciding "should I tell an accurate reputation?" an agent has m^q decision combinations, where m describes a number of possible, discrete degrees of reputation-transaction trustworthiness and q describes the number of reputation transaction requests received from other agents. Since q can be as large as $(a\text{-}1)ae$, the complexity of this decision is $O\left(m^{a^2e}\right)$.

Each decision of a given type is interdependent, but further, each decision type is interdependent. Therefore, in a given discrete timestep, in which an agent has the opportunity to interact with each other agent, as both truster and trustee, for both fundamental and reputation transactions, the agent's trust decision strategy has a complexity of $O\left(2^{ae+a^2e}n^{ae}m^{a^2e}\right)$. It is not feasible for an agent to learn its trust decision strategy without introducing some assumptions to relax interdependencies between decisions. The following section demonstrates how an agent can introduce these assumptions yet still learn a useful trust decision strategy.

3. Case Study: The ART Testbed

The Agent Reputation and Trust (ART) Testbed [1] is used as a case study for learning trust decision strategies. A brief overview of the ART Testbed game is described here. Readers are referred to [10] for more detail; all ART Testbed notation described here is consistent with that reference.

In the ART Testbed's artwork appraisal domain, agents act as appraisers who are hired by clients to deliver *appraisals* about paintings, each for a fixed client fee f. Agents compete for clients, who are assigned based on agents' past appraisal accuracy. In attempting to produce accurate appraisals, appraiser agents may purchase *opinions* (valuations of the painting by other agents) for a fixed fee, c_p. Opinion providers choose an amount, c_g, to pay (to the system) to generate an opinion, symbolizing the effort to perform a valuation; the amount c_g corresponds to the accuracy of the generated opinion. Appraiser agents may also purchase *reputations* (for a fixed fee c_r) from each other to help assess the trustworthiness of opinion providers. Agents compete to achieve the highest bank balance.

Designed as a tool for experimentation and competition, the ART Testbed has earned the attention of numerous researchers in the international trust research community. The Testbed design allows variation of agent strategies and game parameters for easy experimentation, as well as competition against and experiment replication by other researchers. Transactions are conducted sequentially, permitting the separate examination of truster and trustee decisions. Opinion purchases correlate to fundamental transactions, the outcomes of which (opinion accuracy) are observable at the end of each timestep, when true painting values are revealed. The outcomes of reputation purchases are not observable, though reputation receivers may estimate the accuracy of the reputations they receive against previously built trust models. Most importantly, the game's complexity illustrates the interdependencies of trust-related decisions.

The four trust-related decisions translate to the ART Testbed domain as shown in Figure 2. As an Opinion Requester, an agent must decide which opinions to purchase. As an Opinion Provider, an agent must decide how much payment, c_g, to invest in the opinions it communicates. As a Reputation Requester, an agent must decide which reputations to purchase. As a Reputation Provider, an agent must decide how truthful to be, compared to its own trust models, when communicating reputations.

		Transaction	
		Reputation	Fundamental
Agent Role	Provider	How accurate a reputation should I provide?	How accurate an opinion should I provide?
	Requester	Should I purchase this reputation?	Should I purchase this opinion?

Figure 2. Trust decision types in the ART Testbed domain.

Several assumptions are made to the ART Testbed game to remove details unnecessary to this research purpose. First, agent transaction protocols are simplified: Opinion Providers do not communicate opinion certainties, and Opinion and Reputation Providers always satisfy requests, though provided opinions and reputations may be untruthful. Further, agents do not generate opinions for their own clients, and they weight all received opinions equally when calculating their final appraisals. To eliminate non-agent-controlled variances in opinion accuracy, all agents are assigned the same expertise levels for all painting eras (fundamental transaction types).

3.1 Identifying Decision Interdependencies

Figure 3 shows a diagram of interdependencies between the ART Testbed's four trust-related decisions (darkened diamonds in the figure) and resulting observable feedback (costs and earnings, darkened boxes in the figure), enumerated here:

1) *Opinion Requester:* The Opinion Requester's decision regarding which opinions to purchase determines its opinion costs and the accuracy of its resulting final appraisals, which, in turn, impact its client revenue.
2) *Opinion Provider:* The Opinion Provider's decision regarding the accuracy of appraisals to provide determines its opinion order costs (c_g). This decision also affects Opinion Requesters' accuracy of final appraisals and client revenue. Since appraiser agents compete for client revenue, effects on Opinion Requesters' client revenue also impact the client revenue of Opinion Providers. Finally, the Opinion Provider's opinion accuracy influences the number of future requests from Opinion Requesters, which determines the Opinion Provider's future opinion revenue.
3) *Reputation Requester:* The Reputation Requester's decision regarding which reputations to purchase impacts its

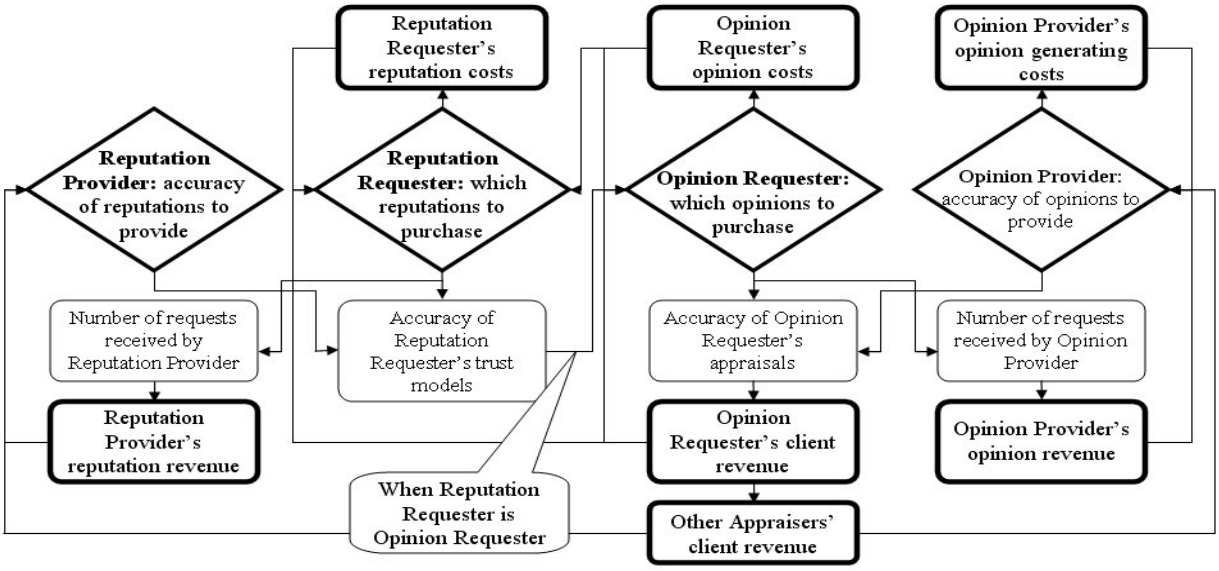


Figure 3. Trust decision interdependencies in the ART Testbed domain.

own reputation costs, as well as its ability to determine which opinions to request, which (as mentioned for the Opinion Requester's role) impacts both the Reputation Requester's opinion costs and client revenue.

4) *Reputation Provider:* The Reputation Provider's decision regarding the accuracy of reputations to provide influences the Reputation Requesters' ability to determine which opinions to request, which affects the Reputation Requester's, and ultimately the Reputation Provider's, client revenue. Also, the Reputation Provider's reputation accuracy influences the number of future requests from Reputation Requesters, which determines the Reputation Provider's future reputation revenue.

Multiple decisions, over all four agent roles, influence multiple feedback elements, making the correlation of specific decisions to their resulting rewards difficult. The following section introduces dependency-eliminating assumptions and proposes methods for calculating rewards associated with each decision.

3.2 Trust Decision Rewards

To conduct q-learning for each trust-related decision, a reward must be assigned to each decision. Since decisions are interdependent, and observable costs and revenues are aggregated, two assumptions are presented to ease these interdependencies and divide observable earnings into per-decision rewards:

Assumption 1: Individual decisions of a single decision type are assumed independent of each other (i.e. all of an agent's opinion-requesting decisions are independent of each other, regardless of opinion provider or era).

Assumption 2: Client revenue is only influenced by the Opinion Requesting decision (thus agents' client revenues are independent of each other).

3.2.1 Opinion Requester Rewards

The Opinion Requester's choice regarding the purchase of a single opinion is binary: either request or not request the opinion. The choice to not request an opinion is assumed to yield a reward of zero, but calculating a reward ($Q_{OR}(true)$) associated with choosing to request an opinion is more complicated. Utilizing the assumptions above, a reward associated with each choice to request an opinion can be estimated by attributing a portion of the Opinion Requester's resulting client revenue first to each computed appraisal, then to each opinion composing that appraisal. First, the total client revenue, v_{total}, attributed to all client appraisals, is calculated as the number of Opinion Requester's clients in the next timestep, n_c, times the earned, per-client fee, f:

$$v_{total} = n_c f \ .$$

Next, the estimated client revenue portion, v_c, attributed to a single appraisal, for client c, is calculated as:

$$v_c = v_{total}\left(\frac{\frac{1}{\varepsilon_{p_c^*}}}{\sum_{i \in C}\left(\frac{1}{\varepsilon_{p_i^*}}\right)}\right),$$

where C represents the set of the Opinion Requester's client appraisals and $\varepsilon_{p_c^*}$ represents the normalized error of the client's painting appraisal, p_c^*, from the painting's true value, t_c:

$$\varepsilon_{p_c^*} = \left|\frac{p_c^* - t_c}{t_c}\right| .$$

An estimated client revenue portion, $v_{c,a}$, is attributed to each opinion (by opinion provider a for client c's painting):

$$v_{c,a} = v_c\left(\frac{\frac{1}{\varepsilon_{p_{c,a}}}}{\sum_{i \in A}\left(\frac{1}{\varepsilon_{p_{c,i}}}\right)}\right),$$

where A represents the set of Opinion Providers for painting c and $\varepsilon_{p_{c,a}}$ represents the normalized error of a single opinion, $p_{c,a}$, from the painting's true value, t_c:

$$\varepsilon_{p_{c,a}} = \left| \frac{p_{c,a} - t_c}{t_c} \right|.$$

Finally, the reward, $w_{OR,c,a}$, associated with requesting the opinion from Opinion Provider a about client c's painting is calculated as the estimated client revenue portion, $v_{c,a}$, minus the cost, c_p, of requesting the opinion:

$$w_{OR,c,a} = v_{c,a} - c_p .$$

This reward is averaged for all requested opinions from a given Opinion Provider a and era e, to which paintings c belong.

3.2.2 Opinion Provider Rewards

The Opinion Provider must choose the accuracy of the opinion it provides to an Opinion Requester. More precisely, the Opinion Provider must choose the amount, c_g, to pay to generate an opinion, where c_g is related to opinion accuracy. First, it is assumed that the Opinion Provider makes the same opinion accuracy decision for all opinion requests from the same Opinion Requester about paintings from the same era. Thus, using the independence assumptions, the reward, $w_{OP,a,e}$, associated with choosing an amount, c_g, to pay to generate an opinion, is calculated as:

$$w_{OP,a,e} = n_{p,a,e}\left(c_p - c_{g,a,e}\right),$$

Where $n_{p,a,e}$ represents the number of opinion requests received in the next timestep from agent a about paintings from era e, c_p represents the per-opinion cost received for providing opinions, and $c_{g,a,e}$ represents the chosen amount paid to generate each opinion for requester a about paintings from era e.

3.2.3 Reputation Provider Rewards

The Reputation Provider must choose the accuracy of the reputation it provides to a Reputation Requester. For this research, it is assumed that all agents have a mutual understanding that reputation values correspond to q-values ($Q_{OR}(true)$) for deciding in the affirmative to request opinions. Since the ART Testbed only permits the exchange of reputations as values between zero and one, q-values are converted from a scale between $-f$-c_p (the assumed worst-case penalty for requesting an opinion) and f-c_p (the assumed best-case reward for requesting an opinion). Therefore, the conversion from $Q_{OR,s,e}(true)$ (the Reputation Provider's actual q-value for requesting opinions from opinion provider (subject-agent) s for expertise era e) to $r_{s,e}$ (the scaled, believed reputation about s for e) is given by:

$$r_{s,e} = \frac{Q_{OR,s,e}\left(true\right) + f + c_p}{2f}.$$

Reputation accuracy is parameterized using the δ variable to describe the relationship between the Reputation Provider's believed reputation, $r_{s,e}$, and the actual reputation, $r^*_{a,s,e}$, it provides to Reputation Requester a. Intuitively, $\delta_{a,s,e}$ represents the percent difference between the believed reputation $r_{s,e}$ and the limits of permissible reputation values, where negative δ values imply pessimistic reputations and positive δ values imply optimistic reputations. Formally, $r^*_{a,s,e}$ is calculated as:

$$r^*_{a,s,e} = \begin{cases} r_{s,e}\left(\delta_{a,s,e} + 1\right), \text{if } -1 \le \delta_{a,s,e} < 0 \\ r_{s,e}\left(1 - \delta_{a,s,e}\right) + \delta_{a,s,e}, \text{if } 0 \le \delta_{a,s,e} \le 1 \end{cases}.$$

Using the independence assumptions, the reward, $w_{RP,a,s,e}$, associated with providing a reputation whose accuracy correlates to $\delta_{a,s,e}$ is given by:

$$w_{RP,a,s,e} = n_{r,s,a,e} c_r ,$$

where c_r is the fixed reputation purchase cost and $n_{r,s,a,e}$ is the number of reputations requested in the next timestep by a about s for era e. Since the maximum possible value of $n_{r,s,a,e}$ is 1 (in a single timestep, a Reputation Requester can only request one reputation from a given provider about a given subject-agent and expertise era), an $n_{r,s,a,e}$ value of zero or one is recorded in each timestep for a chosen δ value, until the timestep in which a new reputation provider decision must be made for the given requester, subject-agent, and era.

3.2.4 Reputation Requester Rewards

The Reputation Requester's decision to purchase reputations is binary. As in the case of the opinion-requesting decision, the reward of not requesting a reputation is assumed to be zero. The reward associated with requesting a reputation is based on the indirect benefit of making savvy opinion requests based on quality reputations, and is estimated as follows. The error of a reputation, $\varepsilon_{r^*_{a',s,e}}$, is estimated by averaging the error between the reputation and computed rewards for requested opinions in C_e (the set of opinions whose paintings belong to this reputation's era e):

$$\varepsilon_{r^*_{a',s,e}} = \frac{\sum_{i \in C_e} \left| r^*_{a',s,e} - w_{OR,i,s} \right|}{\left| C_e \right|},$$

where a' represents the provider of the reputation about opinion provider (subject-agent) s.

An estimated client revenue portion, $v_{r,a',s,e}$, is attributed to each reputation (by reputation provider a' about subject-agent s and era e) in this way: a portion of the client revenue ($w_{OR,c,s}$, attributed to requesting an opinion c) is attributed to each requested reputation according to the reputation's estimated error (A' is the set of reputation providers about s for e). These client revenue portions are averaged over all opinions whose paintings belong to era e:

$$v_{r,a',s,e} = \frac{\sum_{i \in C_e} \left(w_{OR,i,s} \left(\frac{\frac{1}{\varepsilon_{r^*_{a',s,e}}}}{\sum_{j \in A'} \left(\frac{1}{\varepsilon_{r^*_{j,s,e}}} \right)} \right) \right)}{\left| C_e \right|}.$$

The reward, $w_{RR,a',s,e}$, attributed to requesting a given reputation is simply its calculated client revenue portion minus its cost:

$$w_{RR,a',s,e} = v_{r,a',s,e} - c_r .$$

The Reputation Requester must make a second decision which brings to light a significant problem [6]: how should reputation information influence the agent's existing experience-based trust model (the Opinion Requester's $Q_{OR}(true)$ values)? Since a Reputation Requester purchases reputations for the purposes of making better opinion requesting decisions, an agent in the reputation requesting role should be jointly considered in the opinion requesting role. This research employs a weighting parameter λ to perform a weighted average between aggregated reputations and the existing $Q_{OR,s,e}(true)$ value. When λ is 0, no weight is given to reputations, and opinion request decisions are made using only previous experience according to Section 3.2.1. Conversely, when λ is 1, the Opinion Requester's $Q_{OR}(true)$ values are given no weight, instead, the Opinion Requester makes opinion requests based on an average of q-values received as reputations. Reputation request rewards are multiplied by λ since reputations are only responsible for λ fraction of the opinion requesting decision. Note that when an agent serves as a Reputation Provider, it provides reputations from its experience-based $Q_{OR}(true)$ values, excluding any reputations it may have received.

4. Experiments

Once rewards are associated with each trust-related decision as in Section 3.2, q-learning is performed. Experiments are conducted to compare the resulting trust-strategy-learning technique against several standard strategies.

4.1 Experimental Setup

The ART Testbed parameters [10] for these experiments include client fee (f) of 100, opinion cost (c_p) of 10, and reputation cost (c_r) of 1. The accuracy of generated opinions is governed by the parameters s^* (related to appraiser expertise, the standard deviation of opinion error distribution) and α (determines the impact of the opinion-generating cost, c_g). For these experiments, s^* equals 0.1 for all agents regarding all eras, and α is set to 5.0. In each experiment, the simulation assigns client paintings from three eras. The parameter q, which determines the influence of previous client shares on current client shares, equals 0.5. Agents are permitted to select c_g values in one-unit increments between zero and ten (since ten is the opinion purchase price c_p, investments larger than that amount are impractical).

Table 1 displays the seven strategy variations compared. The Naïve-Honest strategy requests opinions from all other agents in the system and provides very accurate opinions, investing $c_g = 10$ to generate each opinion. This strategy is naïve in that reputations are not utilized, therefore none are requested, and default, optimistic reputations of 1.0 are provided to requesters. The Naïve-Cheating strategy is similar to the Naïve-Honest strategy except that it provides very inaccurate opinions (investing $c_g = 0$ in the opinions it provides) and pessimistic reputations (0.0). The Op-Learning-Honest strategy learns opinion requesting decisions from experience (Section 3.2.1) and opinion providing decisions as in Section 3.2.2, while requesting reputations from all agents and providing truthful reputations. The Op-Learning-Cheating strategy behaves similarly to Op-Learning-Honest, but consistently provides pessimistic reputations of 0.0. The Random strategy randomly makes decisions for each of the four decision types. The two Learning strategies demonstrate the learning techniques described in this paper, one with a λ value of 0.0 (learning opinion requesting decisions based on experience only), and the other using λ of 1.0 (learning opinion requesting decisions based on reputations only). All Learning agents employ the same q-learning parameters for each learned decision: learning rate of 0.5, discount factor of 0.0, and temperature, or degree of exploration, of 0.5.

Table 1. Agent strategies.

Strategy Name	Opinion Req.	Opinion Prov.	Reputation Req.	Reputation Prov.
Naïve Honest	request from all	accurate (c_g=10)	none	1.0 (δ= 1)
Naïve Cheating	request from all	inacc. (c_g=0)	none	0 (δ= -1)
Op-Learning Honest	learn by experience	learn	request from all	truthful (δ= 0)
Op-Learning Cheating	learn by experience	learn	request from all	0 (δ= -1)
Random	random	random	random	random
Learning (λ = 0)	learn by experience	learn	learn	learn
Learning (λ = 1)	learn from reputations	learn	learn	learn

In each of ten experiments, a single Learning agent (λ=0 or λ=1) competes against five agents, all of a single strategy: Naïve-Honest, Naïve-Cheating, Op-Learning-Honest, Op-Learning-Cheating, or Random. Each experiment of six agents is run for ten thousand timesteps (for three independent runs) to demonstrate strategy convergence. Each agent's performance is gauged in terms of converged, per-timestep earnings.

4.2 Experimental Results

Results from all ten experiments are shown in Figure 4. In each experiment, per-timestep earnings for each agent converged within approximately the first one hundred timesteps. In all cases, the Learning agent's five opponents (all of like strategy) achieved per-timestep earnings which were statistically similar to each other; these earnings were averaged to arrive at the opponent's average per-timestep earnings (white bars) shown in Figure 4. The opponents' earnings are compared against the Learning agent's earnings (black bars). All differences between Learning agents and their opponents are statistically significant ($\alpha = 0.1$).

In the first pair of experiments, a Learning agent (λ=0 or λ=1) competes in a system with five Naïve-Honest agents. The Learning agent increases its bank balance at nearly twice the rate of each Naïve-Honest agent. Because the Learning agent purchases fewer reputations when λ=0 than when λ=1, the λ=0 agent achieves slightly higher per-timestep earnings than the λ=1 case, and its opponents' earnings are slightly lower (failing to utilize reputations does not penalize the Learning agent, because its opponents are honest). The Learning agent's behavior converges to always requesting opinions from the Naïve-Honest agents, while investing little in the opinions it provides and, in the λ=0 case, rarely requesting reputations.

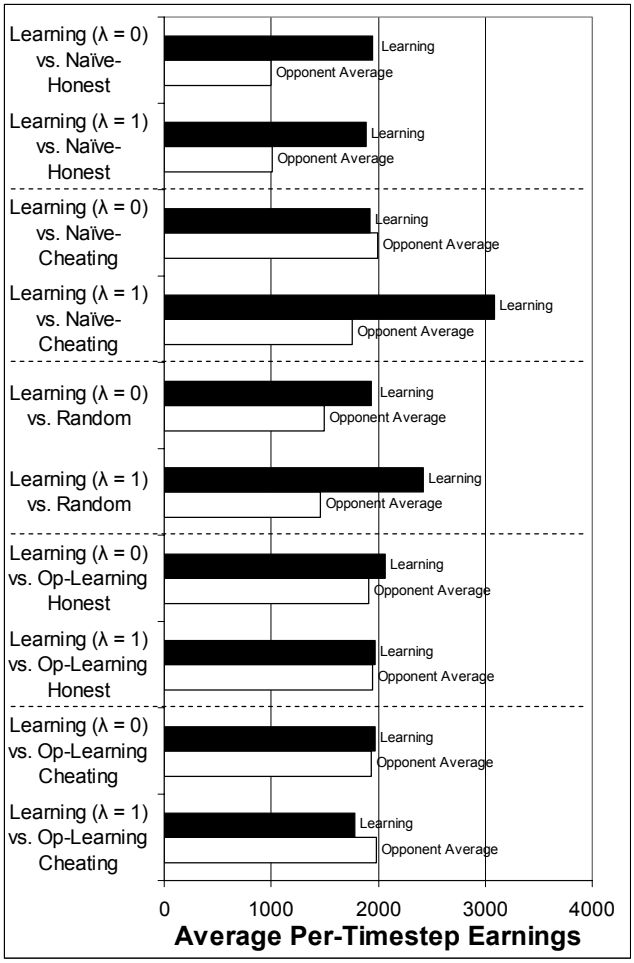


Figure 4. Average, per-timestep earnings for learning strategies (λ=0 and λ=1) versus opponents.

In the second pair of experiments, a Learning agent (λ=0 or λ=1) competes against five Naïve-Cheating agents. The Learning agent with λ=0 maintains per-timestep earnings slightly less than those of the Naïve-Cheating agents. The Learning agent with λ=1 maintains much higher earnings than the Naïve-Cheating agents; it makes use of its opponents' pessimistic reputations, an accurate reflection of the opponents' tendency to cheat.

The third pair of experiments demonstrate that the Learning agent achieves higher earnings than its five Random strategy opponents in both the λ=0 and λ=1 cases. The λ=1 Learning agent achieves higher per-timestep earnings than the λ=0 agent, but Random agents' earnings are the same between the two cases. The Learning agent succeeds by employing a mixed strategy for each of its four decision types.

The fourth pair of experiments compares the Learning agent against five Opinion-Learning-Honest agents. In both cases (λ=0 and λ=1), the Learning agent maintains greater earnings than its opponents. All agents learn to always request opinions and to provide opinions with accuracy that is both beneficial to competitors (to ensure repeated transactions) yet not too costly to generate. Because their opponents naively always request reputations, the Learning agents provide randomly-accurate reputations. As in the Naïve-Honest experiments, the Learning agent yields higher earnings when λ=0 than when λ=1 because the agent purchases fewer reputations in the λ=0 case.

In the last pair of experiments, the Learning agent competes against five Opinion-Learning-Cheating agents. When λ=1, the Learning agent earns lower earnings than its opponents, because it is misled by its opponents' inaccurate reputations. When λ=0, the Learning agent achieves earnings higher than its opponents, relying on the experience-based learning, rather than reputations, to decide which opinions to purchase.

The experiments demonstrate that in only two cases (Learning λ=0 vs. Naïve-Cheating and Learning λ=1 vs. Op-Learning Cheating), the Learning agent earns less than its opponents. However, in both cases, adjusting the λ value (to λ=1 against Naïve-Cheating and to λ=0 against Op-Learning Cheating) allows the Learning agent to outperform its opponents. Further, generalizations can be made about when certain λ values are beneficial: learning from experience (λ=0) achieves higher earnings when opponents tend to be trustworthy, while learning from reputations (λ=1) is beneficial when reputations accurately reflect opponent trustworthiness.

5. Conclusions

Experimental results demonstrate that learning trust decision strategies, either from experience or reputations, yields higher earnings than several static strategies in the ART Testbed domain Further, the variation in earnings of reputation-based learning (λ=1) vs. experience-based learning (λ=0) over different opponents demonstrates successful agents must be able to dynamically determine when to utilize reputations vs. experience in making trust decisions.

This research yields a domain independent contribution by enumerating decisions—and their resulting complexity—that make up a trust decision strategy (who to trust, how trustworthy to be, what reputations to believe, and how truthful to be in telling reputations). Other contributions of this work are adaptable to various types of problems in which trust must be evaluated and reputations may be exchanged. The work identifies interdependencies (which exist since potential partners can influence each other by exchanging reputations about an agent) between those decisions. By introducing assumptions to ease these interdependencies, rewards are attributed to each decision to facilitate reinforcement learning of trust decision strategies. Reward allocation can be adapted to other domains by adjusting transaction costs and earnings associated with domain-introduced interdependencies.

Approaching the complexity of learning trust decision strategies requires the introduction of numerous assumptions. However, these assumptions pave the way for future investigation, which will seek to relax the independence assumptions of Section 3.2 to improve the overall rewards a learning agent achieves. The authors plan to use learning techniques to help an agent learn the best λ, the degree of reliance on experience- versus reputation-based trust, relative to each opinion-requesting decision. The performance of the strategy-learning techniques described in this research requires further analysis against more advanced opponent agents with dynamic strategies. Also, more agent combinations must be tested—including agent self-play—since

trust strategies may yield different results as the composition of the system (strategies and numbers of opponents) change. Since, at the time of writing, the ART Testbed is still awaiting its first competition, sophisticated strategies have yet to be uncovered for comparison. However, the experiments conducted in this research will serve as a benchmark for future progress.

6. Acknowledgment

This research is sponsored in part by the Defense Advanced Research Project Agency (DARPA) Taskable Agent Software Kit (TASK) program, F30602-00-2-0588 and by a subcontract from The University of Texas Applied Research Labs Office of Naval Research Grant N00014-05-1-0857, entitled Improvised Explosive Device. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

7. References

[1] ART Testbed Team. Agent Reputation and Trust Testbed. http://www.lips.utexas.edu/art-testbed/, 2005.

[2] Axelrod, R. The Evolution of Cooperation. New York: Basic Books, 1984.

[3] Banerjee, B., R. Mukherjee, and S. Sen. "Learning Mutual Trust," Proceedings of the Workshop on Deception, Fraud and Trust in Agent Societies at AGENTS-00, pp. 9-14, 2000.

[4] Banerjee, B. and J. Peng. "Countering Deception in Multiagent Reinforcement Learning," Proceedings of the Workshop on Trust, Privacy, Deception and Fraud in Agent Societies at AAMAS-03, Melbourne, Australia, pp. 1-5, 2003.

[5] Barber, K. S. and J. Kim. "Belief Revision Process based on Trust: Agent Evaluating Reputation of Information Sources," Trust in Cyber-societies: Integrating the Human and Artificial Perspectives, vol. 2246, Lecture Notes in Computer Science, Falcone, R., Singh, M., and Tan, Y.-H., Eds.: Springer, pp. 73-82, 2002.

[6] Barber, K. S. and J. Kim. "Soft Security: Isolating Unreliable Agents," Proceedings of The Workshop on Deception, Fraud, and Trust in Agent Societies at AAMAS-02, Bologna, Italy, pp. 8-17, 2002.

[7] Barber, K. S., K. Fullam, and J. Kim. "Challenges for Trust, Fraud, and Deception Research in Multi-agent Systems," Trust, Reputation, and Security: Theories and Practice, R. Falcone, K. S. Barber, L. Korba and M. Singh, Eds., Springer: pp. 8-14, 2003.

[8] Barber, K. S. and K. Fullam. "Applying Reputation Models to Continuous Belief Revision," Proceedings of The Workshop on Deception, Fraud and Trust in Agent Societies at AAMAS-03, Melbourne, Australia, pp. 6-15, 2003.

[9] Crandall, J.W. and M.A. Goodrich. "Establishing Reputation Using Social Commitment in Repeated Games," Proceedings of the Workshop on Learning and Evolution in Agent Based Systems at AAMAS-04, New York, pp. 12-17, 2004.

[10] Fullam, K., T. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. Rosenschein, L. Vercouter, and M. Voss. "A Specification of the Agent Reputation and Trust (ART) Testbed: Experimentation and Competition for Trust in Agent Societies," Proceedings of The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005), Utrecht, pp. 512-518, 2005.

[11] Huynh, T.D., N.R. Jennings, and N. Shadbolt. "On Handling Inaccurate Witness Reports," Proceedings of 8th International Workshop on Trust in Agent Societies at AAMAS-05, Utrecht, pp. 63-77, 2005.

[12] Jonker, C.M. and J. Treur. "Formal Analysis of Models for the Dynamics of Trust Based on Experiences," Proceedings of The 9th European Workshop on Modeling Autonomous Agents in a Multi-Agent World: Multi-Agent System Engineering (MAAMAW-99), pp. 221-231, 1999.

[13] Littman, M. and P. Stone. "Leading Best-Response Strategies in Repeated Games," Proceedings of the Workshop on Economic Agents, Models, and Mechanisms, at IJCAI-01, Seattle, Washington, 2001.

[14] Mui, L., A. Halberstadt, and M. Mohtashemi. "Notions of Reputation in Multi-Agents Systems: A Review," Proceedings of The First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002), Bologna, Italy, pp. 280-287, 2002.

[15] Sandholm, T. and R. Crites. "On Multi-agent Q-Learning in a Semi-Competitive Domain," Proceedings of the Workshop on Adaptation and Learning in Multiagent Systems at IJCAI-95, Montreal, Canada, pp. 71-77, 1995.

[16] Schillo, M., P. Funk, and M. Rovatsos. "Using Trust for Detecting Deceitful Agents in Artificial Societies," Proceedings of The Applied Artificial Intelligence Journal, Special Issue on Deception, Fraud and Trust in Agent Societies, pp. 825-848, 2000.

[17] Shi, J., G. Bochmann, and C. Adams. "Dealing with Recommendations in a Statistical Trust Model," Proceedings of the Workshop on Trust in Agent Societies at AAMAS-05, Utrecht, pp. 144-155, 2005.

[18] Teacy, L., J. Patel, N.R. Jennings, and M. Luck. "Coping with Inaccurate Reputation Sources: Experimental Analysis of a Probabilistic Trust Model," Proceedings of The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005), Utrecht, pp. 997-1004, 2005.

[19] Weinberg, M. and J. Rosenschein. "Best-Response Multiagent Learning in Non-Stationary Environments," Proceedings of The Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2004), New York, pp. 506-513, 2004.

[20] Yolum, P. and M.P. Singh. "Self-Organizing Referral Networks: A Process View of Trust and Authority," Engineering Self-organizing Systems, Lecture Notes in Artificial Intelligence, pp. 195-211, 2003.

[21] Yu, B. and M.P. Singh. "An Evidential Model of Distributed Reputation Management," Proceedings of The First International Joint Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy, pp. 294-301, 2002.

Monopolizing Markets by Exploiting Trust

Sandip Sen
University of Tulsa
sandip@utulsa.edu

Dipyaman Banerjee
University of Tulsa
dipu@utulsa.edu

ABSTRACT

We investigate trust-based relationships in electronic supply chains where trust is a measure of consistency in meeting negotiated contract deadlines. We consider scenarios where contractors assign contracts to contractees who are significantly more successful in meeting deadlines. The task deadlines are drawn from a known distribution. We present a probabilistic analysis that enables contractees to strategically bid on only certain tasks to earn the trust of their contractors. Once a contractee achieves a high level of trust, it can then exploit that trust to increasingly corner a larger portion of the market share of all tasks. We present a trust exploitation scheme that monopolizes the market against greedy contractees who bid on all announced tasks. We also show that such market monopolization is not possible in the presence of a trusted, but non-exploiting, contractee. The exploiter can, however, effectively "starve" the non-exploiting trusted agent.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Coherence and coordination, Multiagent systems, Intelligent agents*

General Terms

Algorithm, Performance, Experimentation

Keywords

trust, monopoly, markets, supply chains

1. INTRODUCTION

Supply chains form distributed, dynamic networks of agile, adaptive entities. As a result, supply chain management where rational agents represent interests of individual entities and organizations have been an area of active research [16, 21, 23]. From the multiagent systems perspective, a supply chain is often modeled using a decentralized network of agents where each agent can do a part of a task. Typical approaches try to optimize the profitability of entities with emphasis on pricing and scheduling [7].

Researchers in business and management science, however, have recognized that a key component of decision-making in real-world supply chains is the consideration of trust between the contracting organizations [12, 24]. Though the use of information technology and computer systems to facilitate trust-building between organizations have been proposed in mid-80s [14], little work has been done in this area.

We believe an agent based supply chain can effectively capture the needs and constraints faced by real-life B2B systems, and hence, can become an integral part of fielded applications only when such higher level issues like trust are adequately addressed. In our previous work we focused on the aspect of developing stable, long term, trusted partnerships that allow organizations to have more accurate predictions of the time and cost requirements to meet market demands [18]. This work measured trust as the likelihood of completing contracted tasks by their deadlines. Agents that were sufficiently more trusted were always awarded contracts over others. Task deadlines were drawn from a known distribution, and strategic agents bid only on tasks for which they were sufficiently more confident of meeting deadlines. Over time, such an agent would be recognized as more trustworthy by the contractor(s) and would be preferred over other agents who bid for all tasks. We refer to such trust-building agents as *trust-building (TB)* agents.

In this paper, we investigate how a trusted contractee agent can exploit the trust of its contractor(s) to eliminate the competition and monopolize market share. From a self-interested viewpoint, trust is like any other resource that can be leveraged to gain competitive advantage. We propose that once an agent has build sufficient trust, the next logical step would be to exploit that to incrementally take over the entire market. This process mirrors the development of a monopolist in a marketplace though practical scenarios are more complex and involve aspects beyond trust. We refer to agents who exploit their trust as *trust exploiting (TE)* agents.

We further evaluate the performance of such a monopolistic agent in the presence of a non-exploiting TB agent who has no intention to monopolize. The presence of the TB agent presents a major impediment to the TE agent's goal of monopolizing the market directly. We motivate, implement, and evaluate two variations of TE agent that tries to corner as much of the market share as possible even if

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

monopolization is precluded.

We use a contracting framework for connecting the suppliers to the manufacturers: manufacturers announce contracts for tasks with given deadlines; suppliers bid on these tasks; and the contract is allocated to one of the bidders based on its past performance history. We use a simplified contracting model where contractors prefer to assign tasks to contractees who have significantly higher success rates in meeting deadlines. We assume that the trust preferences of the contractor, the task deadline distribution and the performance distribution of the contractees are known. We then develop a precise bidding strategy for trust-building contractees. The motivation is to bid only on those tasks for which they have a high likelihood of meeting deadlines. However, not bidding on tasks also reduces the success rate of completing tasks. We provide a probabilistic analysis to handle this tradeoff. We then augmented this strategy to incorporate strategic shifts in the bidding range, once trust has been established, to monopolize the market. Further, we investigate the different scenarios which may occur when another trusted agent is present in the market. Finally we provide experimental verification of our formal analysis.

We emphasize that we are not pursuing, in this paper, a game-theoretic approach to equilibrium analysis in the context of trust-based contracting. While such an analysis is critical and necessary under the assumption of all bidders being rational, we are more interested in approximately modeling real world situations. In typical market environments, participants have a varying degree of knowledge, longevity, resources, strategic reasoning capability, etc. In particular, we model market situations where almost of all of the contractors use a myopic or greedy approach to bidding for tasks. This myopic behavior can stem from short market lifetime, financial exigencies, resource constraints, lack of knowledge or reasoning capability, etc. The research question we are pursuing is how should a rational contractor, with long-term market presence, approximate knowledge about market conditions, and no limiting resource constraint, etc. choose its bidding strategy to maximize its long-term utility. While we follow intuitive guidelines while developing and exploiting trust, the technical contribution of this work lies in formally developing precise bidding strategies with accurate performance predictions followed by experimental verification. Whereas trust and reputation in agent based systems is a very active research area in multiagent systems [5, 6, 10, 15, 20, 22, 25], to date there exists very little work on formal derivation of strategies to build or exploit trust.

2. CONTRACTING MODEL

We assume a trust-based model for task allocations by a contractor to bidders in a marketplace.

2.1 Task and performance distributions

We assume a task distribution $\mathcal{T}$ from which the deadline of different tasks are drawn. $\mathcal{P}_j$, defined over the closed interval $[l, h]$, represents the distribution from which the actual time taken by supplier j to process tasks is drawn. For continuous distributions, this means $\int_l^h \mathcal{P}_j(x)dx = 1$ and $\int_l^h T(y)dy = 1$. The corresponding discrete distributions, where T and $\mathcal{P}_j$ represent probability mass functions, are $\sum_{x=h}^{l} \mathcal{P}_j(x) = 1$ and $\sum_{y=h}^{l} T(y) = 1$ respectively.

2.2 Use of trust in selecting contractees

Each contractor maintains a trust rating, t_c for each possible contractee, c, which is the proportion of time that contractee could meet an assigned task deadline. For any two bidders, $\underline{b}, \overline{b} \in B_i$, we say $\overline{b}$ γ-dominates $\underline{b}$ (equivalently, $\underline{b}$ is γ-dominated by $\overline{b}$) if

$$\gamma * t_{\underline{b}} < t_{\overline{b}}, \tag{1}$$

where $\gamma > 1$ is a trust constant, chosen by the contractor. The basic motivation or incentive for the contractor to use a trust measure for selecting suppliers is to ensure reliability. If a supplier fails to deliver the required materials or products by the negotiated deadline, the manufacturer may not be able to meet its own production schedules and delivery deadlines and hence may face stiff penalties.

Each task announcement, T_i, contains an associated deadline $d(T_i)$ drawn from the distribution $\mathcal{T}$. Each supplier decides to bid or not bid on an announced task. As we are concentrating only on the trust evaluation of the bidders in awarding tasks, a bid from a bidder j can just be a default message signaling js interest in the task. The contractor of a task assigns it to a contractee randomly chosen from the set of non-γ-dominated bidders in the set of bidders for the task, B_i.

Note that this is a deliberate simplification of the real-world contracting process. In particular, we have left out the consideration of the price in the bid while awarding contracts! We believe that real-world stable supply chains incorporate trust considerations as a key metric for awarding contracts. We acknowledge that even in such situations, trust is one of several key parameters, including price, that determines contract awards. In this study, however, we wanted to focus exclusively on trust considerations as there has been very little work in agent based systems on the use of trust in supply chain contracting. Another way of situating this work is to assume that the contractor is choosing, based on trust, between all bidders with the same price. The issues we are studying can also be analyzed effectively with only one manufacturer and several suppliers and contractees, and hence we do not use a full supply chain. The techniques presented, however, readily transfers to more elaborate supply chains.

2.3 Goal of contractee

Let there be N contractees. The goal of a contractee j is to maximize its success in procuring and delivering contracts. The objective criterion to be maximized, the *success rate*, is[1]:

$$\int_{bl}^{bh} r(j, y)T(y) \left(\int_l^y \mathcal{P}_j(x)dx \right) dy, \tag{2}$$

where $r(j, y)$ is the probability that a task of length y is assigned to supplier j by the contractor (depends on the set of bidders for the task and their respective trust values), the integration within the parenthesis represents the likelihood of meeting the deadline y and bl, bh are respectively the minimum and maximum deadlines for tasks on which j will bid. The *success rate* represents the expected number of assigned tasks successfully delivered by their deadline.

[1]Most of the analysis presented in the following uses notations corresponding to continuous distributions. The equivalent expressions for discrete distributions can be similarly derived.

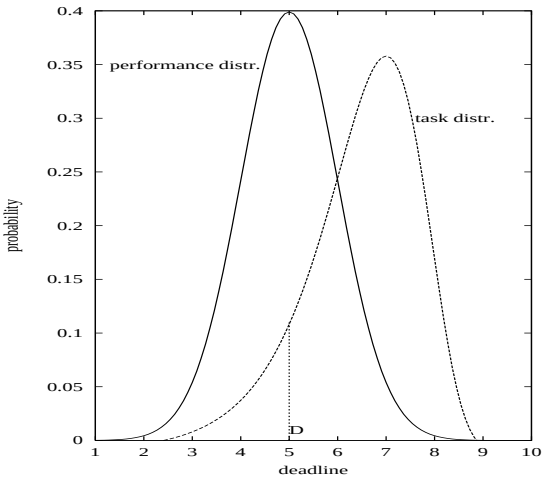


Figure 1: Example continuous task deadline and performance distributions.

3. BUILDING TRUST

We assume that the contracting model is common knowledge between the contractor and the suppliers and that each supplier j is cognizant of the task deadline distribution $\mathcal{T}$, its performance distribution $\mathcal{P}_j$ and the trust constant γ. We now present two types of bidding strategies for a contractee j in response to an announcement for a task t_i with deadline $d(T_i)$:

Greedy (G): This strategy bids for a task whenever there is a non-zero probability of meeting the deadline, i.e., $\int_l^{d(T_i)} \mathcal{P}_j(x)dx > 0$.

Trust-building (TB): This strategy focuses on building trust, which is measured as the likelihood of meeting the deadline of an assigned task. This strategy is elaborated in the rest of this section.

The number of contractees using the G and TB strategy are known and given by N_G and N_T respectively. Let $N = N_G + N_T$.

The key consideration for a TB agent is the choice of the minimum deadline threshold, D, for bidding such that it is viewed by the contractor to be sufficiently more trustworthy, in the sense of γ-domination, than a G contractee. We assume that performance distribution $\mathcal{P}$ is the same for all agents. The Figure 1 presents a typical situation with task and performance probability distributions and the deadline D below which TB agents will not bid.

The average expected success likelihood or *trustworthiness* of an agent j who wins, i.e., is awarded, all tasks in the region $[bl, bh]$ is given by

$$\bar{P}(bl, bh) = \frac{1}{\mathcal{T}_{bl,bh}} \int_{bl}^{bh} T(y) \left(\int_l^y \mathcal{P}_j(x)dx \right) dy, \quad (3)$$

where $\mathcal{T}_{x,y} = \int_x^y T(z)dz$, is the cumulative probability of tasks arriving with deadlines in the region $[x, y]$. The above expression is obtained using Equation 2 with $r(j, y) = 1$, i.e., agent j wins the contract with certainty. If an agent wins only a fraction f of tasks in that region, the corresponding average success likelihood is $f\bar{P}(bl, bh)$.

To facilitate the presentation of the analysis of strategic bidding for tasks, we consider the steady state case, where TB agents always win the contract when they bid (as they have been recognized to be more trustworthy than greedy contractees), and at other times one of the G agents win the contract. First we note that the TB agents should be more inclined to bid on tasks at the higher end of the task deadline distribution as they are more likely to be able to meet the corresponding deadlines. Hence, the upper limit of the range of task deadlines a TB agent will bid, bh_{TB} is equal to h, the maximum task deadline. Let the minimum deadline for which a TB agent bids at steady state be D_{ss}. Therefore, at steady state, the TB agents will win all the tasks for deadlines in the range $[D_{ss}, h]$ and G agents will win all contracts in the region $[l, D_{ss})$. From Equation 1, the following inequality holds if the TB agents were to γ-dominate the greedy agents at steady state:

$$\bar{P}(D_{ss}, h) > \gamma \bar{P}(l, D_{ss}). \quad (4)$$

Equations 3 and 4 can be used to calculate D_{ss}.

The assumption implicit in Equation 4, that the TB agents win whenever they bid, is not valid initially, when they are yet to be recognized as γ-dominant. Hence D_{ss} is not the appropriate choice for the initial minimum task deadline to bid for, D_I, by TB agents. To calculate D_I, we assume that tasks are initially assigned randomly between all bidders. So, while all the tasks with deadlines in the range $[l, D_I)$ will be assigned to G contractees, tasks in the region $[D_I, h]$ will be assigned to G versus TB bidders in the ratio of N_G to N_{TB}. The choice of D_I should be such that it allows a TB agent to have at least γ times higher trust than G agents when tasks are being assigned randomly between bidders:

$$\frac{N_{TB}}{N} \bar{P}(D_I, h) > \gamma \left(\frac{\mathcal{T}_{l,D_I} \bar{P}(l, D_I) + \mathcal{T}_{D_I,h} \frac{N_G}{N} \bar{P}(D_I, h)}{\mathcal{T}_{l,h}} \right). \quad (5)$$

The left hand side of the inequality represents the proportion of tasks expected to be successfully delivered by TB agents when tasks are randomly assigned between all bidders and the TB agents bid only in the interval $[D_I, h]$. The term within the parenthesis on the RHS of the inequality denotes the proportion of tasks successfully delivered by G agents in this period. The terms $\mathcal{T}_{l,D_I}$ and $\mathcal{T}_{D_I,h}$ are used to normalize the trustworthiness in the regions $[l, D_I]$ and $[D_I, h]$ respectively. To calculate D_I we simplify Equation 5

$$\bar{P}(D_I, h) > \frac{N\gamma\mathcal{T}_{l,D_I}}{N_{TB} - \gamma N_G \mathcal{T}_{D_I,h}} \bar{P}(l, D_I), \quad (6)$$

where the simplification uses the fact that $\mathcal{T}_{l,h} = 1$ and that

$$N_{TB} > \gamma N_G \mathcal{T}_{D_I,h}. \quad (7)$$

The inequality in Equation 6 can be satisfied for a range of D_I values. The TB agent uses the minimum value in the range which also satisfies the inequality in Equation 7.

In this paper we use equal number of TB and G agents and hence Equation 6 simplifies to

$$\bar{P}(D_I, h) > \frac{2\gamma\mathcal{T}_{l,D_I}}{1 - \gamma\mathcal{T}_{D_I,h}} \bar{P}(l, D_I), \quad (8)$$

and Equation 7 simplifies to $\mathcal{T}_{D_I,h} < \frac{1}{\gamma}$.

The goal of the contractee is to maximize its success rate as described in Section 2.3. Hence, a contractee must evaluate whether it will have a better success rate, in the long

run, if it decides to be a TB or a G agent. If all agents were of type G, each contractee will be awarded tasks with equal probability, i.e., $\forall j, r(j,y) = \frac{1}{N}$. With N_{TB} TB agents, the probability of such an agent winning a contract when it bids, and after it becomes γ-dominant, is $\frac{1}{N_{TB}}$. Hence, in addition to satisfying the inequality in 6 and 7, the D_I value chosen must satisfy the following inequality

$$\int_{D_I}^{h} T(y) \left(\int_{l}^{y} \mathcal{P}_j(x)dx \right) dy > \frac{N_{TB}}{N} \int_{l}^{h} T(y) \left(\int_{l}^{y} \mathcal{P}_j(x)dx \right) dy. \quad (9)$$

Given T and $\mathcal{P}$ distributions, we designate by γ_{max} the maximal γ value for which there exists a D_I that satisfies all these conditions.

3.1 Bidding strategy of Trust Exploiting Agents

We now present the strategy of the *trust exploiting (TE)* agent. Whereas the TB agent picks a D_I and bids only for tasks above it, TE will progressively bid over larger parts of the task distribution, i.e., for smaller task deadlines, until it monopolizes the market. The key to understanding how such a monopoly can be achieved is to recognize that as a TE agent shifts its deadline to bid, D_I, to the left, i.e., for tasks with shorter deadlines, its trust level drops but so does that of greedy agents who now win only on even more riskier tasks. The following discussion assumes that the TE agents first enter a market with G agents and have achieved γ-domination by bidding for tasks in the range $[D_I, h]$. At this stage, the trust ratings of TE and G agents are $t_{TE} = \bar{P}(D_I, h)$ and $t_G = \bar{P}(l, D_I)$ respectively. Actually, t_{TE} should be much more than the value required to γ-dominate t_G because D_I was calculated assuming half the tasks in the safer task deadline range $[D_I, h]$ go to G agents, which is not true after TE agents achieve γ-domination. Hence it is possible for TE agents to use a lower task deadline, D_{new} to bid on and still maintain γ-domination over the G agents, i.e., $D_{new} < D_I$ and $\bar{P}(D_{new}, h) > \gamma \bar{P}(l, D_{new})$. After sticking with this new deadline for sometime to stabilize new trust values, the TE agents can again expand its bidding range, and continue to do so until it bids over the entire range, effectively monopolizing all task contracts! This process of expanding the bid range after achieving γ-domination over the G agents is described in Algorithm 1.

Algorithm 1 Selecting increasingly larger bidding range to achieve market monopoly.

while $D_I > l$ **do**
  Find D_{new} such that $\bar{P}(D_{new}, h) > \gamma \bar{P}(l, D_{new})$.
  $D_I \leftarrow D_{new}$
  Bid in the range $[D_I, h]$ for a sufficient period to settle on new trust values
end while

The situation is more challenging for the TE agent if it enters the market when a TB agent is already γ-dominating G agents. The first task for the TE agent to gain a foothold is to γ-dominate the TB agent. The corresponding bidding deadline for tasks, D_{TE} can be calculated from the inequality, $\bar{P}(D_{TE}, h) > \gamma \bar{P}(D_I, D_{TE})$ (we assume here that newcomers to the marketplace are given some tasks to evaluate their trustworthiness). Once it has achieved that domination, there are two possibilities for further exploitation:

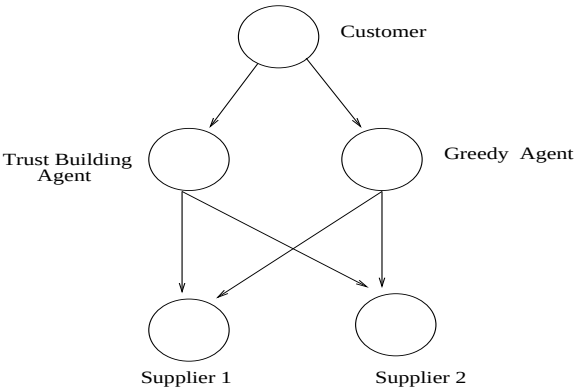


Figure 2: A simple supply chain.

TE1: TE1 expands to bid for all tasks. As a result it will lose γ-domination over TB. When this happens, it switches back to original D_{TE}. Thus the TE1 agent switches its low-end bid deadline between D_{TE} and l.

TE2: TE2 expands to coincide its bidding deadline with that of the TB agent. It may eventually lose γ-domination over TB agent, depending on the task deadline distribution. When this happens, it switches back to original D_{TE}. Thus the TE2 agent switches its low-end bid deadline between D_{TE} and D_I.

A TE2 agent can, depending on the task distribution, "starve" the TB agent out of the market. If the TB agent then leaves the market, we have the scenario of a TE agent against G agents, which will eventually lead to the TE agent monopolizing the market. In other situations, a TE2 agent may outperform a TE1 agent when facing the same TB and G competitors, but not each other.

4. EXPERIMENTAL FRAMEWORK

We use a simple two-level supply chain consisting of a single contractor at the top level, two to three contractees in the second level, and two suppliers under each such contractee at the third level. A representative instance is presented in Figure 2. We ran our experiments in two discrete phases. In the first phase we compared the behavior of two agents bidding, one TB and one TE agent when they play separately against the same G agent, for a contract offered by the customer, the contractor. In the next phase we allow all three agents to operate simultaneously and compared their performances. The deadlines for the contracts are generated randomly from a discretized triangular distribution. We used triangular distribution as an approximation of likely real-life distributions because of easy calculations of the cumulative probabilities. There are two suppliers under each of these two contractees, and the contractee agents must procure supplies from these downstream suppliers before it can process an assigned task.

Since the total time taken by the contractee is the sum of the time taken by its supplier to produce the necessary supplies plus the time it takes to use these supplies to process the tasks, the corresponding task distributions are chosen so that the resultant performance distribution range matches that of its task distribution. We have used identical triangular distributions for the contractee and the suppliers

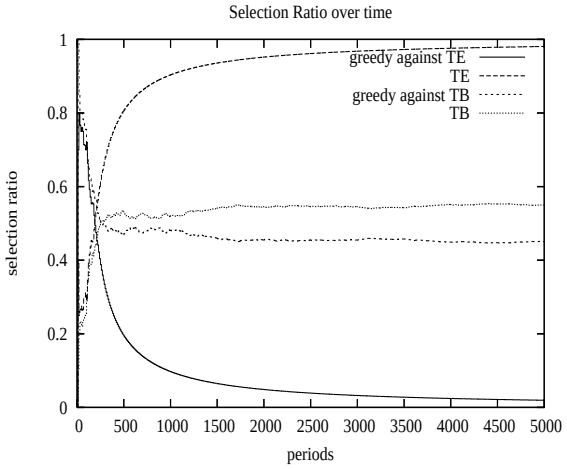


Figure 3: Selection Ratio ($T_p = 8$, $\gamma = 1.29$).

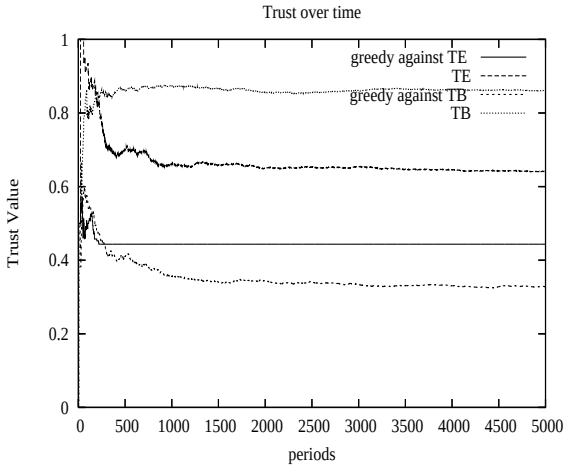


Figure 4: Trust over time ($T_p = 8$, $\gamma = 1.29$).

defined over the range [0,6] with the highest point of the distribution at the midpoint of the range. The resultant performance distribution of the contractee then ranges over [0,12].

In our experiments, we iteratively generated new task deadlines from the task distribution. In a particular iteration, the TB agent decides to bid if the deadline is greater than D_I. The G agent bids on all tasks and TE agent bids if the deadline is greater than its current D_I value.

In the first 100 iterations the customer randomly selects between the agents with bids to estimate the trustworthiness of the contractees. Thereafter a contractee for a task is chosen using the selection criteria specified in Section 2.2. We also assume that the customer selects any new bidder entering the marketplace with a small probability, ϵ, to estimate its performance. After the new agent is selected 50 times using this ϵ-exploration, the customer reverts back to its previous selection strategy using the computed trust values based on these exploration phase. This exploration phase is used only when the TE agent enters the market after TB agent is γ-dominating the G agents.

When a contractee is awarded a task, it generates a task announcement to procure necessary supplies from the suppliers. The time taken by the suppliers followed by that of the selected contractee are then generated from their performance distributions using the standard inverse transform method and are added to calculate the contract fulfillment or delivery time. The agent succeeds if the delivery time is less than or equal to the contract deadline, and fails otherwise.

5. EXPERIMENTAL RESULTS

In our first experiment, we compare the selection ratio and the trust value of the greedy and trust-based contractees over 5000 iterations, where the selection ratio is the proportion of time the agent is selected. The experiment is repeated on three different task distributions: T_p= 4, 6 and 8 and for $\gamma = 1.29$. The results from the $T_p = 8$ scenario is shown in Figure 3 (plots for other values used for T_p are similar and are omitted due to space constraints).

From Figure 3 we observe that though TB agents are selected more often than the greedy agents, it can never get

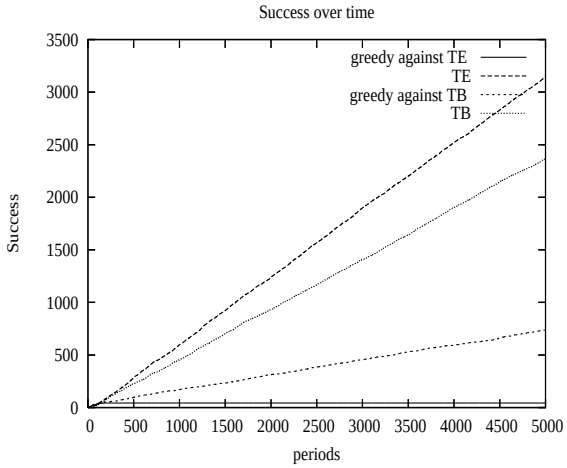


Figure 5: Success over time ($T_p = 8$, $\gamma = 1.29$).

more than 60% of the task as it never bids on a task whose deadline is less than D_I. On the other hand, the selection ratio of the TE agent asymptotically reaches 1 as it starts taking over the market once it has gained sufficient trust. To ensure that, it observes the market continuously and after winning successive 20 bids it sets its new D_I at 2 forcing the G agent to operate only on the high-risk tasks having deadline from 0 to 1. This causes the greedy agent to fail more often and thus loose its trust even more. The trust value of TE agent also decreases and is less than that of TB as shown in Figure 4. But the rate of decrease of trust for the TE agent is much slower than that of the G agent which enables the former to maintain γ-domination. As a result it continues winning all the tasks it bids, and after 20 wins it shifts D_I to 0 and takes over the entire market. From this point onward the G agent does not win any bid, and as a result its trust value remains constant (Figure 4).

We observe from Figure 5 that the TE agent receives more success compared to the TB agent when they play separately against the greedy agent. Also note that, the G agent wins no contracts once TE takes over the market.Due to sampling

biases, a sequence of tasks with short deadlines may be generated which the TE agent may fail to meet, thereby loosing γ-domination over G agents. To counter this problem, the monopolist TE agent monitors the situation and whenever it looses enough bids to loose γ-domination over the greedy agent, it reverts back to its initial D_I value to rebuild the lost trust. Once it gains back the trust, it uses Algorithm 1 to re-monopolize the market.

In our next set of experiments, only the TB and G contractee agents are in the system for the first 400 iterations. Once the TB contractee gain γ-domination over G, a TE contractee is introduced. Here again we assume that the customer uses ϵ-exploration to evaluate the new entrant. We used two different versions of TE agent: TE_1 and TE_2. Both of these agents choose a higher task deadline threshold D_{TE} to achieve γ-dominance over not only the G agent, but also the TB agent. Once it gains the required trust, TE_1 tries to monopolize the market by setting its D_I to 0 and oscillates back and forth to maintain its trust domination. From the results (see Figure 6) we observe that the success of TE_1 depends on the nature of the task distribution; it does well for the distribution with $T_p = 4$, but for task distributions with $T_p = 6$ and higher, TB outperforms TE_1. This is because, in order to γ-dominate the TB agent, the TE_1 bids on a very small range of the task spectrum for such distributions when tasks with longer deadline is common. In such situations, it is not able to win sufficient number of tasks to outperform the TB agent.

TE_2 behaves less greedily than TE_1: after it gains γ-domination over the TB agent, instead of monopolizing the market it sets its D_I at the same position as the TB agent. The TB agent, therefore, loses some of its market share to the TE_2 agent. From Figures 6 and 7 we observe that the success rates of TE_2 are better than that of TE_1. The TE_2 agent, however, loses out to the TB agent for task distributions with $T_p = 8$ and above. In such situations, eventually TE_2 loses γ-domination over TB and moves its bidding deadline to D_{TE}, and after regaining trust switches it back to D_I of TB agent, and the cycle repeats. TE_2 also retains its γ-domination over the greedy agent through this entire process. But this oscillation does not happen for all task distributions, e.g., for $T_p = 4$ and 6, TE_2 never needs loses γ-domination over TB even when it continues to play in the range $[D_I, h]$. This is because, when TE_2 becomes γ-dominant over TB while playing in the range $[D_{TE}, h]$, the TB agent is only awarded tasks with deadline in the range $[D_I, D_{TE})$. For $T_p = 4$ and 6, not only is $\bar{P}(D_{TE}, h) > \gamma\bar{P}(D_I, D_{TE})$ required for establishing γ-domination, but $\bar{P}(D_I, h) > \gamma\bar{P}(D_I, D_{TE})$, i.e., the TE_2 agent continues to maintain γ-domination even after bidding over the range on which the TB agent was bidding when the TE agent enters the market. As a result we observe that in these situations (see the middle and left plots in Figure 7), the TB agent receives no contracts after some time, i.e., it is effectively "starved". Note that the TE_2 agent cannot monopolize all the contracts, and hence the G agent continues to receive the more riskier contracts on which none of the trust-based agents are bidding. Ironically, the presence of the TB agent benefits the G agent: TB starves, but prevents TE from starving G as well!

6. RELATED WORK

More recently the problem of supply chain management has drawn the attention of multiagent systems (MAS) researchers. In MAS, a supply chain is conceptualized as a group of collaborative autonomous software agents [11]. It is argued that managers can better coordinate and schedule processes by distributing the organization-wide business management system into autonomous problem solving agents. This approach is more appropriate when the participating enterprises are geographically distributed and manually controlling $B2B$ trading is not possible. Tackling coordination in supply chains using partial constraint satisfaction problems by mediating agents is investigated by [2].

Swaminathan *et al.* has provided a framework for efficient supply chain formation [21]. MAS researchers have paid attention to the emergence of the optimal supply chain configuration. Walsh *et al.* has shown the optimal dynamic task allocation in a supply chain using combinatorial auctions [23]. Given a task, composed of a group of subtasks, they provide a mechanism for dynamic formation of a supply chain to produce maximum profit. Collins and Gini have provided a testbed, MAGNET, for multiagent contracting for supply chain formation using multiple criteria [7]. A few years back, the Trading Agent Competition (TAC) [16] introduced a realistic supply chain framework where the challenges include the design of strategies for effectively handling factory schedules, efficiently contracting with the suppliers, competitively and profitably bidding to the customers, reducing the inventory cost and penalty for late delivery or order cancellation.

In recent research, trust is established as a key factor to build profitable and long term B2B and B2C collaborations in the Industry [12]. Trust plays an even more critical and important role in the domain of Electronic commerce [3]. Several trust management systems have been proposed to handle the development of trust and its impact in such systems [9].

Use of trust as a basis for interaction strategies has been widely used in multiagent systems. Marsh was one of the first to attempt a computational model of trust [13]. Sen *et al.* have developed a comprehensive reciprocity-based reputation framework for learning to trust like-minded agents and outperform exploitative agents in fully distributed environments like P2P networks [17, 19, 1]. Castelfranchi and colleagues have argued for the necessity of trust in social interactions between agents with complex mental attitudes [5]. Brainov and Sandholhm [4] have shown that trust based contracting can significantly increase market efficiency measured by social welfare, trade volume and agent utilities. Yu and Singh [26] have proposed a mechanism for combining reputation from multiple sources to obtain trust ratings. Glass and Grosz present the SPIRE framework [8], where agents who fulfill social obligations are treated favorably by other members of the society. Sen *et al.* have developed a comprehensive reciprocity-based reputation framework for using trust in

7. CONCLUSIONS AND FUTURE WORK

In this paper, we have argued for the use of trust models to award contracts in the context of supply chains. We work with a particular trust-based contracting framework where contractees with significantly higher historical success rates in meeting contract deadlines are preferentially selected over less "punctual" agents. Our proposed trust-building agents bid less often for tasks, but win and suc-

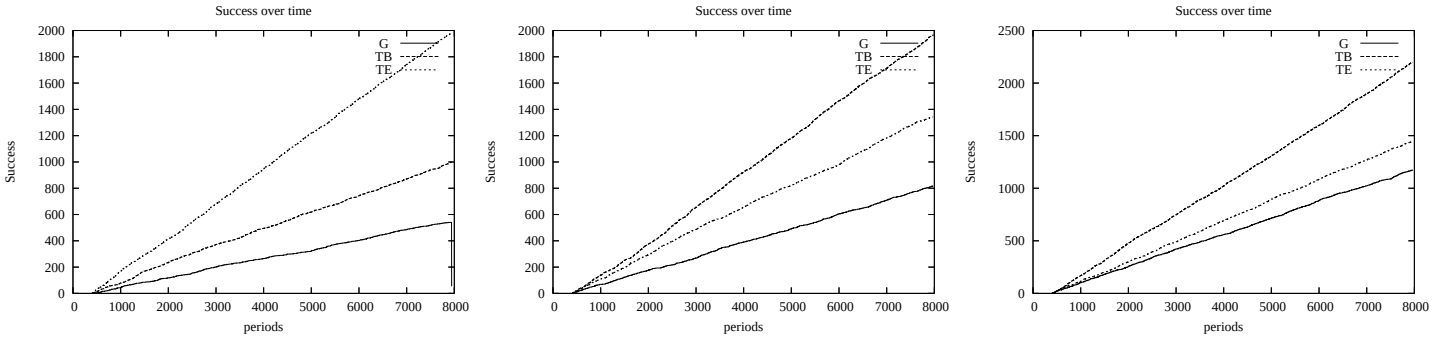


Figure 6: Success of G, TB, and TE_1 agents over time for different task distributions: T_p = 4 (left), 6 (middle), 8 (right); $\gamma = 1.15$.

cessfully meet the deadline of tasks they bid for more often than "greedy" agents who bid for all tasks.

Given task deadline and performance, i.e., task processing distributions, we use a probabilistic analysis to analytically derive the task deadline threshold below which a trust-building contractee will not bid for a task. We expand on this analysis to develop exploitative agents who incrementally take over the entire market from greedy agents. We also propose variations of exploitative agents when competing against both greedy and non-exploiting trust-building agents. We provide experimental verification on a small supply chain to demonstrate the competitive advantage of these trust-exploiting strategies.

While the basic motivations and outlines for trust-exploitation strategies presented in this paper are intuitive, the contribution of our paper lie in formally specifying the trust-exploitation problem and deriving precise strategies for exploiting trust. We present implementable strategies with predictable performance given task deadline and performance distributions and the trust-based contract allocation criterion. We first derived a trust-based strategy that trades off short-term loss (not bidding on certain tasks) for long-term profits (higher success rates after achieving γ-domination). Following this we introduced variants of aggressive trust-exploitation schemes that not only establishes trust as the previous scheme, but then leverages it strategically to monopolize the market. We show that though such market monopolization is not possible if non-exploitative trust-based agents are present, the latter can be effectively dominated under certain task deadline distributions.

In this paper, the trust constant, γ, and the contract allocation procedure are assumed to be common knowledge. The task distribution is assumed to be constant and known *a priori*. Similarly, the performance distribution is fixed and is assumed to be identical for all contractees. Some or all of those assumptions may be violated in a real environment. For example, the contractor may not be consistent in either maintaining or enforcing the trust-based contract award scheme, task distribution may vary over time, agents may be heterogeneous in their capabilities, etc. We plan to relax some of this approaches in future work. In particular, we plan to explore various adaptive mechanisms to cope with dynamic, uncertain environments.

Acknowledgments This work has been supported in part by an NSF award IIS-0209208.

8. REFERENCES

[1] D. Banerjee, S. Saha, P. Dasgupta, and S. Sen. Reciprocal resource sharing in p2p environments. In *Proceedings of the Fourth Intenational Joint Conference on Autonomous Agents and Multiagent Systems*, pages 853–859, New York, NY, 2005. ACM Pres.

[2] J. C. Beck and M. S. Fox. Supply chain coordination via mediated constraint relaxation. In *Proc. of the 1st Canadian Workshop on Distributed Artificial Intelligence*, 1994.

[3] A. Bhattacherjee. Individual trust in online firms: Scale development and initial test. *Journal of Management Information Systems*, 19(1):211–241, 2002.

[4] S. Brainov and T. Sandholm. Contracting with uncertain level of trus. *Computational Intelligence*, 18(4):501–514, November 2002.

[5] C. Castelfranchi and R. Falcone. Principles of trust for MAS: Cognitive autonomy, social importance, and quantification. In *Proceedings of the Third International Conference on Multiagent Systems*, pages 72–79, Los Alamitos, CA, 1998. IEEE Computer Society.

[6] C. Castelfranchi and Y.-H. Tan. The role of trust and deception in virtual societies. In *Proceedings of the 34th Hawaii International Conference on System Sciences*, Los Alamitos, CA, 2001. IEEE Computer

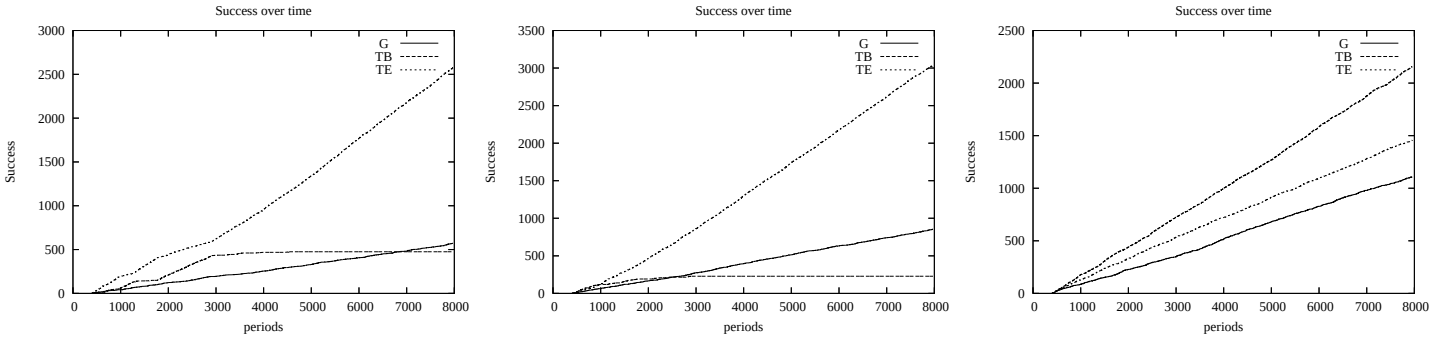


Figure 7: Success of G, TB, and TE_2 agents over time for different task distributions: $T_p = 4$ (left), 6 (middle), 8 (right); $\gamma = 1.15$.

Society.

[7] J. Collins, W. Ketter, and M. Gini. A multi-agent negotiation testbed for contracting tasks with temporal and precedence constraints. *International Journal of Electronic Commerce*, 7(1):35–57, 2002.

[8] A. Glass and B. Grosz. Socially conscious decision-making. *Autonomous Agents and Multi-Agent Systems*, 6(3):317–339, May 2003.

[9] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 3(4), 2000.

[10] N. Griffiths and M. Luck. Coalition formation through motivation and trust. In *Proceedings of the Second Intenational Joint Conference on Autonomous Agents and Multiagent Systems*, pages 17–24, New York, NY, 2003. ACM Pres.

[11] N. R. Jennings, P. Faratin, T. J. Norman, P. O'Brien, and B. Odgers. Autonomous agents for business process management. *International Journal of Applied Artificial Intelligence*, 14(2):145–189, 2000.

[12] Y. L. Lederer-Antonucci, P. S. Greenberg, M. zur Muehlen, and G. Ralph. Establishing trust in a business-to-business collaboration: Results from an international simulation. In *Proc. of the IRMA 2003 Conference*, pages 922–924, 2003.

[13] S. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, April 1994.

[14] R. Miles and C. Snow. Organizations: New concepts for new firms. *California Management Review*, 28(3):62–73, 1986.

[15] M. Schillo, P. Funk, and M. Rovatsos. Using trust for detecting deceiptful agents in artificial societies. *Applied Artificial Intelligence*, 14:825–848, 2000.

[16] Tac, supply chain management. url: http://www.sics.se/tac/.

[17] S. Sen. Believing others: Pros and cons. *Artificial Intelligence*, 142(2):179–203, 2002.

[18] S. Sen, S. Saha, and D. Banerjee. Trust-based contracting in supply chains. In *Proceedings of the Fourth Intenational Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1251–1252, New York, NY, 2005. ACM Pres.

[19] S. Sen, S. Saha, and P. S. Dutta. Helping based on future expectations. In *Proceedings of the Second Intenational Joint Conference on Autonomous Agents and Multiagent Systems*, pages 289–296, New York, NY, 2003. ACM Pres.

[20] S. Sen and N. Sajja. Robustness of reputation-based trust: Boolean case. In *Proceedings of the First Intenational Joint Conference on Autonomous Agents and Multiagent Systems*, pages 288–293, New York, NY, 2002. ACM Pres.

[21] J. M. Swaminathan, S. F. Smith, and N. M. Sadeh. Modeling supply chain dynamics: A multiagent approach. *Decision Sciences*, 29(3):607–632, 1998.

[22] Y. Tan and W. Thoen. An outline of a trust model for electronic commerce. *Applied Artificial Intelligence*, 114(8):849–862, 2000.

[23] W. E. Walsh and M. P. Wellman. Decentralized supply chain formation: A market protocol and competitive equilibrium analysis. *Journal of Artificial Intelligence Research*, 19:513–567, 2003.

[24] B. Welty and I. Becerra-Fernandez. Managing trust and commitment in collaborative supply chain relationships. *Communications of the ACM*, 44(6):67–73, 2001.

[25] P. Yolum and M. P. Singh. Engineering self-organizing referral networks for trustworthy service selection. *IEEE Transactions on System, Man, and Cybernetics*, 2005.

[26] B. Yu and M. P. Singh. Distributed reputation management for electronic commerce. *Compuational Intelligence*, 18(4):535–549, 2002.

Developing a Composite Trust Model for Multi-agent Systems

Xiaoqing Zheng, Zhaohui Wu, Huajun Chen, Yuxin Mao
College of Computer Science, Zhejiang University, Hangzhou 310027, China
+86 - 0571 - 87953703
{zxqingcn, wzh, huajunsir, maoyx}zju.edu.cn

ABSTRACT
The Semantic Web, conceived as a collection of agents, brings new opportunities and challenges to trust research. Enabling trust to ensure more effective and efficient interaction is at the heart of the Semantic Web vision. We propose a composite trust model based on statistical decision theory and Bayesian sequential analysis to balance the costs and benefits during the process of trust evaluating and combine a variety of sources of information to assist users with making the correct decisions in selecting the appropriate service providers according to their preferences. The model proposed by this paper gives trust a strict mathematical interpretation in terms of probability theory.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *Intelligent agents, Multiagent systems*; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation – *Probabilistic computation.*

General Terms
Theory, Security, Algorithms.

Keywords
Trust Model, Semantic Web, Bayesian Sequential Analysis.

1. INTRODUCTION
In our model, the quality of a provider agent can be considered to be an unknown numerical quantity, and will represent it by θ and it is possible to treat θ as a random quantity with a probability distribution. Consider the situation of an agent A try to make an estimate of trust value for provider agent B. A holds a prior information (subjective) of B, represented by distribution $\pi(\theta)$, and request A's acquaintance to give opinions on B's quality. After A receives the assessments of B's quality from his acquaintances, A takes these statements as sample about θ. Outcome of these sample is a random variable and will be denoted $\boldsymbol{X}$. A particular realization of $\boldsymbol{X}$ will be denoted x and $\boldsymbol{X}$ will be assumed to be a random variable, with density $f(x|\theta)$. Then, we can compute "posterior distribution" of θ given x, denoted $\pi(\theta|x)$. $\pi(\theta|x)$ reflects the update beliefs about θ after (posterior to) observing the sample x. We take the posterior distribution of θ, $\pi(\theta|x)$, as the estimate of B's trust. If we want to take further investigation on B's quality for more accuracy, $\pi(\theta|x)$ will be used as prior distribution for the next stage of investigation instead of original $\pi(\theta)$.

When several provider agents exist, A need to decide which one should be selected. At that time, the preferences of agent A's owner should be considered properly to make this decision. Therefore, *utility function* should be constructed for agent A's owner, which represented by $U_A(r)$, to express his preferences, where r represents rewards of the consequences of a decision. Supposing that $\pi(\theta|x)$ is the posterior distribution of agent B, the expected utility of function $U_A(r)$ over $\pi(\theta|x)$, denoted $E^{\pi(\theta|x)}[U_A(r)]$, is possible gain of consequence of selecting B.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

2. COMPOSITE TRUST MODEL
By treating an agent as a node, the "knows" relationship as an edge, a directed graph emerges. To facilitate the model description, agents and their environment are to be defined. To clarify the idea of our trust model, we begin with a simple illustration. Consider the scenario that agent A is evaluating trust value of B and C for being business. The set of all consultant agents that A requests for this evaluation as well as A, B, C can be considered to be a unique society of agents. In our example (see Figure 1) it is called a "closed society of agents" with respect to A.

2.1 Closed Trust Model
We will treat the service quality, θ, as a continuous variable here, and suppose that the service charges of B and C are 250 and 210 units respectively (SC_B = 250, SC_C = 210, where SC denotes *service charge*). As shown in Figure 1, the agent A feels that θ_B, the service quality of B, has a normal prior density, $N(0.64, (0.08)^2)$. We also suppose that the prior density of θ_C is $N(0.5, (0.15)^2)$ here. The probability distribution of $\boldsymbol{X}$ that represents the assessments of service quality from consultant agents will, of course, depend upon the unknown state of nature θ. Therefore, we assume that $\boldsymbol{X}$ is another continuous random variable with density $f(x|\theta) \sim N(\theta, (0.05)^2)$. We also assume that the agent A's owner has a utility function $U_A(r) = 208.4 + 459.8r - 17.95r^2 - 99.84r^3$.

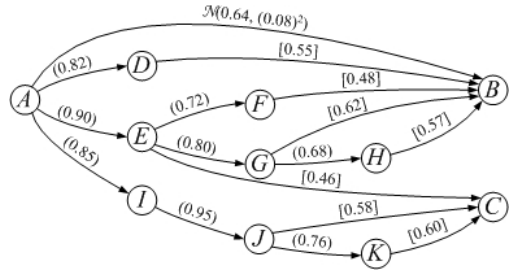


Figure 1. A "closed society of agents" with respect to agent *A*

Assume $X \sim N(\theta, \sigma^2)$, where θ is unknown but σ^2 is known. Let be prior information, $\pi(\theta)$, a $N(\mu, \tau^2)$ density, where μ and τ^2 are known. Note that the marginal distribution of $\boldsymbol{X}$, $m(x)$, is $N(\mu, \sigma^2 + \tau^2)$ and the posterior distribution θ of give x is $N(\mu(x), \rho)$, where

$$\mu(x) = \frac{\sigma^2}{\sigma^2 + \tau^2}\mu + \frac{\tau^2}{\sigma^2 + \tau^2}x \qquad \rho = \frac{\tau^2\sigma^2}{\tau^2 + \sigma^2} \qquad \textbf{(1)}$$

We take the assessments of *B*'s quality from the consultant agents as sample about θ_B and combine these sample information (x) and prior information into posterior distribution of θ_B given x. In order to answer the question of which sample information should be used with higher priority, we propose following order rules: (1) The sample from the agent with shorter referral distance should be used first; (2) If the samples come from the agents that have the same referral distance, which with larger reliability factor is prior to that with smaller one.

Now, we can evaluate trust value of *B* for *A* by using above formulas and information. Firstly, we use the sample information from *D* through the path $A \rightarrow D \rightarrow B$, and the posterior distribution $\pi_B(\theta|x = 0.55) \sim N(0.5753, (0.0424)^2)$, where

$$\frac{0.05^2}{0.08^2 + 0.05^2} \times 0.64 + \frac{0.08^2}{0.08^2 + 0.05^2} \times 0.55 = 0.5753$$

$$\frac{0.08^2 \times 0.05^2}{0.08^2 + 0.05^2} = 0.0424^2$$

However, the reliability factor has not been considered during the process of calculating this posterior distribution. We used following formula to rectify above $\pi_B(\theta|x = 0.55)$, where p_{old}, p and p_{new} represent the parameter of prior distribution, the posterior distribution before rectification and the posterior distribution after rectification respectively (p is mean or variant for normal distribution), and R is reliability factor.

$$p_{new} = p_{old} + (p - p_{old}) \times R \qquad \textbf{(2)}$$

Hence, after rectification, $\mu_B = 0.64 + (0.5753 - 0.64) \times 0.82 = 0.5869$ and $\tau_B = 0.08 + (0.0424 - 0.08) \times 0.82 = 0.0492$. Then the posterior distribution $\pi_B(\theta|x = 0.55)$ is $N(0.5869, (0.0492)^2)$. The residual calculating process of *B* and the whole the process of *C* are shown in Table 1 and 2. Note that we employ multiplying to merge two or more than two reliability factors. The reason behind using multiplying is that if the statement is true only if the agents who propagate this statement all tell the truth and it is considered to be independent for any two agents to lie or not to lie. Expected utility of *B* is:

$$\begin{aligned}\text{utility of } B &= \int_{-\infty}^{+\infty} U_A(r)\pi_B(\theta \mid x)d\theta - SC_B \\ &= \int_{-\infty}^{+\infty}(208.4 + 459.8\theta - 17.95\theta^2 - 99.84\theta^3) \times \\ &\quad \frac{1}{\sqrt{2\pi} \times 0.0309} e^{-\frac{(\theta - 0.5696)^2}{2 \times 0.0309^2}} d\theta - 250 \\ &= 195.82\end{aligned}$$

$$\begin{aligned}\text{and utility of } C &= \int_{-\infty}^{+\infty} U_A(r)\pi_C(\theta \mid x)d\theta - SC_C \\ &= 225.34\end{aligned}$$

hence (225.34 > 195.82), *C* is more appropriate than *B* in the eyes of *A* .

2.2 Open Trust Model

Above discussion is under the condition that the "closed society of agents" must be defined at first, but it is nearly impossible for inherent open and dynamic Web. Our idea is that at every stage of the procedure (after every given observation) one should compare the (posterior) utility of making an immediate decision with the "expected" (preprosterior) utility that will be obtained if more observations are taken. If it is cheaper to stop and make a decision, that is what should be done.

The goal of preposterior analysis is to choose the way of investigation which minimizes overall cost. This overall cost consists of the decision loss (*opportunity cost*) and the cost of conducting observation (*consultant fee*). Note that these two quantities are in opposition to each other, we propose an approach to balance these two costs.

Table 1. The process of evaluating *B*'s trust

Step	Path	Statement	Reliability Factor	Prior Distribution		Posterior Distribution		Utility
				μ	τ	μ	τ	
1	$A \rightarrow D \rightarrow B$	0.55	0.8200	0.6400	0.0800	0.5869	0.0492	201.39
2-1	$A \rightarrow E \rightarrow G \rightarrow B$	0.62	0.7200	0.5869	0.0492	0.5986	0.0390	205.46
2-2	$A \rightarrow E \rightarrow F \rightarrow B$	0.48	0.6480	0.5986	0.0390	0.5695	0.0337	195.75
3	$A \rightarrow E \rightarrow G \rightarrow H \rightarrow B$	0.57	0.4896	0.5695	0.0337	0.5696	0.0309	195.82

Table 2. The process of evaluating *C*'s trust

Step	Path	Statement	Reliability Factor	Prior Distribution		Posterior Distribution		Utility
				μ	τ	μ	τ	
1	$A \rightarrow E \rightarrow C$	0.46	0.9000	0.5000	0.1500	0.4676	0.0577	198.72
2	$A \rightarrow I \rightarrow J \rightarrow C$	0.58	0.8075	0.4676	0.0577	0.5194	0.0416	218.06
3	$A \rightarrow I \rightarrow J \rightarrow K \rightarrow C$	0.60	0.6137	0.5194	0.0416	0.5396	0.0357	225.34

Table 3. The process of Bayesian sequential analysis

Stage	Agent *B*				Agent *C*				Consultant Fee	Utility		Decision
	Prior Distribution		Marginal Distribution		Prior Distribution		Marginal Distribution			Immediate Decision	Further Investigation	
	μ_B	τ_B	$\mu_{x\|B}$	$\tau_{x\|B}$	μ_C	τ_C	$\mu_{x\|C}$	$\tau_{x\|C}$				
1	0.6400	0.0800	0.6400	0.0943	0.5000	0.1500	–	–	3	217.78	219.24	Continue
2	0.5869	0.0492	0.5869	0.0701	0.4676	0.0577	–	–	3	201.39	201.80	Continue
3	0.5695	0.0337	–	–	0.5194	0.0416	0.5194	0.0650	2	218.06	216.09	Stop

We continue above example used in the illustration of the closed trust model. As shown in Figure 2, we begin at the stage 1 when *A* only holds the prior information of *B* and has no any information about *C* (even the existence of *C*, but it is more likely that an agent with the prior distribution of $N(0.5, (0.15)^2)$ and the expected service charge of 210 is near in the network). Agent *A* either can make an immediate decision (to select *B*) or can send request to his acquaintances for their opinions by extending the tree of Figure 2 down to the next layer.

Suppose that the cost of consultant services is determined by how much agents will be requested at the next stage and the consultant fee is the constant of 1 for each times (for example, at the stage 1, agent *A* can ask Agent *D*, *E* and *I* for their opinions, so the consultant fee of the stage 1 will be 3).

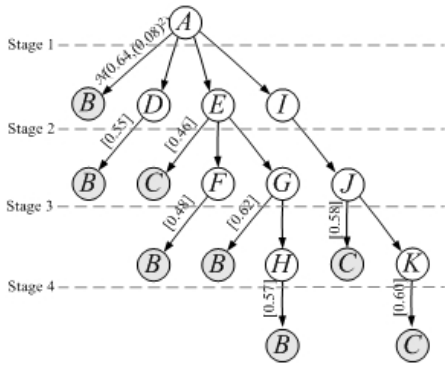


Figure 2. The process of trust evaluating

The utility of an immediate decision is the larger of

$$\int_{-\infty}^{+\infty} U_A(r)\pi_B(\theta)d\theta - SC_B = 217.78$$

here, $\pi_B(\theta) \sim N(0.64, (0.08)^2)$.

and $\int_{-\infty}^{+\infty} U_A(r)\pi_C(\theta)d\theta - SC_C = 207.54$

here, $\pi_C(\theta) \sim N(0.5, (0.15)^2)$.

Hence the utility of an immediate decision is 217.78.

If the request message is sent and x observed, the posterior density $\pi_B(\theta|x)$, is $N(\mu(x), \rho)$, where

$$\mu(x) = \frac{0.05^2}{0.08^2 + 0.05^2} \times (0.64) + \frac{0.08^2}{0.08^2 + 0.05^2}(x) \cong 0.1798 + (0.7191)x$$

$$\rho = \frac{0.08^2 \times 0.05^2}{0.08^2 + 0.05^2} \cong 0.0018$$

However, that we do not know which x will occur, but we know the relevant distribution for $\boldsymbol{X}$ is the "predictive" or marginal distribution, $m_B(x)$, which in this situation is $N(0.64, (0.08)^2 + (0.05)^2)$. Note that if $x < 0.5914$ is observed, the expected utility of $\pi_B(\theta|x)$ is less than 207.54, so we prefer to select *C* instead of *B*. Hence expected utility of not making immediate decision is

$$\int_{-\infty}^{0.5914} 207.54\, m_C(x)dx +$$

$$\int_{0.5914}^{+\infty} (\int_{-\infty}^{+\infty} U_A(r)\pi_B(\theta \mid x)d\theta - SC_B)m_B(x)dx - 3 = 219.24$$

This just is the opportunity cost. Because 219.24 > 217.78, and then further investigation would be well worth the money, in other words, *A* should send request to his acquaintances for their opinions. The residual process of Bayesian sequential analysis are shown in Table 3 and remember that the further exploiting should be terminated immediately along the path on which a cycle is detected. As shown in Table 3, at the stage 3, the expected utility of *C* begins to larger than that of *B*, and because 218.06 > 216.09, making an immediate decision is more profitable. Therefore, *A* should stop investigating and select *C* as a decision. The advantage of sequential analysis should be clear now. It allows one to gather exactly the correct amount of data needed for a decision of the desired accuracy.

3. ACKNOWLEDGMENT

The work is funded by subprogram of China 973 project (NO. 2003CB317006), National Science Fund for Distinguished Young Scholars of China NSF program (NO. NSFC60533040), and also Program for New Century Excellent Talents in University of Ministry of Education of China (NO. NCET-04-0545).

4. REFERENCES

[1] James, O. B. Statistical Decision Theory and Bayesian Analysis. Springer-Verlag, New York Inc, 1985.

[2] O'Hara, K. Alani, H. Kalfoglou, Y. and Shadbolt, N. Trust Strategies for the Semantic Web. In *Proceedings of Workshop on Trust, Security, and Reputation on the Semantic Web at International Semantic Web Conference (ISWCI '04)*, Hiroshima, Japan, 2004.

[3] Richardson, M. Agrawal, R. and Domingos, P. Trust Management for the Semantic Web. In *Proceedings of the International Semantic Web Conference (ISWCI '03)*, Sanibel Island, Florida, USA, 2003, 351-368.

Trust-based Agent Community for Collaborative Recommendation*

Jianshu Weng, Chunyan Miao, Angela Goh
School of Computer Engineering
Nanyang Technological University,
Singapore

Zhiqi Shen, Robert Gay
School of Electrical and Electronic Engineering
Nanyang Technological University,
Singapore

{WENG0004,ASCYMiao,ASESGoh,ZQShen,EKLGay}@ntu.edu.sg

ABSTRACT

There exist a number of similarity-based recommendation communities, within which similar users' opinions are collected by users' agents to make predictions of their opinions on a new item. Similarity-based recommendation communities suffer from some significant limitations, such as scalability and susceptibility to the noise. In this paper, we propose a trust-based community to overcome these limitations. The trust-based recommendation community incorporates trust into the domain of item recommendation. Experimental results based on a real dataset show that trust-based community manages to outperform its similarity-based counterpart in terms of prediction accuracy, coverage, and robustness in the presence of noise.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Information filtering

General Terms: Algorithm, Performance

Keywords: Trust Metric, Recommendations

1. INTRODUCTION

The past few years have seen the emergence of the online recommendation communities, such as Epinions[1] and Movielens[2]. Users join the community to collect others' opinions (usually in the form of ratings), which are then used to make predictions of new items' ratings, which recommend the users whether to browse the new items.

Traditionally, similarity-based approach is applied to make predictions in communities, i.e. only opinions from agents[3] with similar preferences are taken into account when making the predictions. The similarity between a pair of agents is generally measured according to the "agreements" between individual items over all the items co-rated by the two agents [6]. However, the similarity-based communities have major limitations, for example:

(1) Agents usually rate few items [2]. It is thus not unusual that agents do not co-rate the minimum number of items required to compute the similarity. For example, it is required that agents co-rate at least two items to calculate the Pearson correlation coefficient, which is a widely used similarity metric. Hence, the number of similar agents is small. In some cases, agent even cannot find similar agents. As a consequence, a large number of agents are not able to make predictions.

(2) By intentionally rating only a small number of items, malicious agents are easier to achieve high similarity since the number of co-rated items is small. For example, Pearson correlation gives a similarity of either 1, -1 or 0 when the number of co-rated items is 2. Consequently, malicious agents' noisy ratings[4] can easily outweigh others' ratings, leading to the biased predictions on given items.

(3) Typically, there is a centralized server that collects the opinions and makes predictions for the agents. The computation burden of the centralized server increases quickly with the increase of the number of agents.

To overcome the aforementioned limitations, we propose a trust-based community by incorporating *trust* into the domain of item recommendation. This paper adapts Barber's definition[1] which defines *trust* to be *the expectation of technically competent role performance*. More specifically, *trust* is interpreted as an agent's expectation of another agent's competence in providing opinions to reduce its uncertainty in predicting new items' ratings. The main components of the trust-based community are as follows:

(1) Agents in trust-based communities take opinions of *trustworthy* agents into account when making predictions. A trust metric is designed to help an agent to quantify the degrees of trust it should place on other agents. Unlike the similarity metric, the trust metric is computable on most agents. It is even computable on pairs of agents who only co-rate one item. A new approach is also proposed to make predictions based on trustworthy agents' opinions.

(2) Each agent collects others' opinions actively when it encounters new items, and then makes predictions of the new items' ratings with local computations. At the

*This work was partially supported by IMSS grant from the Agency for Science, Technology and Research, Singapore.

[1]Epinions: http://www.epinions.com/

[2]Movielens: http://movielens.umn.edu/

[3]In the rest of this paper, *agent* is used to denote the user who is the owner of the agent.

[4]Presence of noisy ratings means that agents might give unfairly positive or negative opinions for the purpose of biasing other agents' predictions for or against certain items.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Ag_s's opinions	Ag_t's opinions					Total
	1	...	j	...	Z	
1	n_{11}	...	n_{1j}	...	n_{1Z}	R_1
$\vdots$			$\vdots$			$\vdots$
i	n_{i1}	...	n_{ij}	...	n_{iZ}	R_i
$\vdots$			$\vdots$			$\vdots$
Z	n_{Z1}	...	n_{Zj}	...	n_{ZZ}	R_Z
Total	C_1	...	C_j	...	C_Z	N

Table 1: Ag_s's experience with Ag_t

same time, each agent contributes to the community by sharing its opinions when it is contacted by others for opinions. Hence, each agent is in fact a peer servent, and the community is basically a P2P overlay network. The P2P-based architecture avoids the need for centralized server.

The outline of the rest of this paper is as follows. The main components of the trust-based community are described in Section 2. Experimental results are presented in Section 3. Finally, Section 4 concludes the paper.

2. COMPONENTS OF THE TRUST-BASED COMMUNITY

There are two sets of entities within the communities.

(1) a set of uniquely identifiable *items* $O = \{O_1, O_2, \ldots, O_j, \ldots, O_m\}$,

(2) a set of uniquely identifiable *agents* $Ag = \{Ag_1, Ag_2, \ldots, Ag_i, \ldots, Ag_n\}$, who have their opinions $r_{i,j}$ on the *items*.

Here $r_{i,j}$ denotes Ag_i's *opinion* (in the form of rating) on O_j after browsing O_j. Agents' predictions of items' ratings are also given in the form of ratings. Ag_i's prediction on O_j's rating is denoted as $Pr_{i,j}$. When contacted by other agents, agents' opinions on the items, but not the predictions, are shared.

As discussed earlier, each agent is treated equally as a peer servent. Without loss of generality, the rest of this paper is presented from the perspective of a particular agent Ag_s.

2.1 Design of the Trust Metric

The proposed trust metric is experience-based, i.e., the trustworthiness of a given agent, say Ag_t, is calculated by Ag_s based on its previous experience with Ag_t[5]. Ag_s compiles its experience with Ag_t in a table as shown in Table 1. In Table 1, Z denotes the number of possible ratings, and N is the total number of items co-rated by Ag_s and Ag_t. Each cell n_{ij} records the number of co-rated items on which Ag_t's opinions are j while Ag_s's opinions (after browsing the items) are i. Each cell is initiated with dummy experience number as $1/Z$. After Ag_t shares opinions as j for the first time, the dummy experience in all the cells of column j are cleared first, then cells in column j are updated based on real experience. The dummy experience is introduced to eliminate the possible appearance of the undefined $0/0$ when calculating the trustworthiness.

$R_i = \sum_{j=1}^{Z} n_{ij}$, i.e. the number of co-rated items on which Ag_s's opinions are i; $C_j = \sum_{i=1}^{Z} n_{ij}$, i.e. the number of co-rated items on which Ag_t's opinions are j. $\sum_i R_i = \sum_j C_j = N$. After compiling the experience with Ag_t, agent Ag_s calculates Ag_t's *trustworthiness* $Tr_{s,t}$ as:

$$Tr_{s,t} = \frac{(1-\sum_i (R_i/N)^2) - (1 - \frac{1}{N}\sum_i \sum_j (n_{ij}^2/C_j))}{1-\sum_i (R_i/N)^2} \quad (1)$$

$$= \frac{N \sum_i \sum_j \frac{n_{ij}^2}{C_j} - \sum_i R_i^2}{N^2 - \sum_i R_i^2} \quad (2)$$

[5] Ag_s has experience with Ag_t means Ag_t's opinions have ever been collected by Ag_s to make predictions.

As discussed before, $Tr_{s,t}$ is interpreted as Ag_s's expectation of Ag_t's competence in providing opinions to reduce its uncertainty in predicting new items' ratings. The reduction of uncertainty can be measured as the reduction of the incorrect predictions given Ag_t's opinions. In the right hand side of Eq. (1), the first and second term in the numerator are the proportions of the incorrect predictions made without Ag_t's opinions and with Ag_t's opinions respectively. Thus, Eq. (1) quantifies agent Ag_t's trustworthiness by measuring the reduction of incorrect prediction given Ag_t's opinions. Eq. (1) is further simplified to Eq. (2).

$Tr_{s,t}$ takes values in $[0, 1]$. $Tr_{s,t} = 0$ implies that Ag_t does not reduce Ag_s's uncertainty at all, whereas $Tr_{s,t} = 1$ implies Ag_s's perfect predictability of the new items' ratings given Ag_t's opinions. Every time after a new prediction has been made with Ag_t's opinions, Ag_s updates its evaluation of Ag_t's trustworthiness.

2.2 Opinions Collection

Ag_s maintains a set of *direct neighbors*, whom Ag_s contacts directly for opinions when it encounters a new item. Each direct neighbor, when contacted by Ag_s, shares its opinion if it has rating on the target item. Moreover, each direct neighbor returns all its direct neighbors as well as their trustworthiness as *referrals* to Ag_s. Instead of contacting all the referrals, Ag_s applies a probabilistic strategy in determining whether to contact a particular referral at the next step. Suppose Ag_d returns a number of referrals. Ag_s will contact a particular referral Ag_m at the next step with a probability $Tr_{d,m}$, and will not contact Ag_m with a probability $1 - Tr_{d,m}$. $Tr_{d,m}$ is Ag_d's evaluation of Ag_m's trustworthiness. Also, Ag_s will not contact an agent if it has been contacted before for the same item. Then each contacted agent carries out the similar actions as Ag_s's direct neighbors do. The only difference is that it returns referrals only when it has no opinions to share.

A path starting from agent Ag_s connecting all the contacted agents (including the direct neighbors) and ending at the agent who has opinion on the target item is called a *referral chain*. An upper limit (U) is applied on the length of a referral chain to avoid endless opinion collection.

Opinions collection usually discovers more than one agent that can share opinion on a target item. Ag_s can use the proposed trust metric to determine those agents' trustworthiness if it has previous experience with them. Trust propagation is necessary to help Ag_s reason about the trustworthiness of the agents who share opinions but Ag_s has no previous experience with. The trust propagation mechanism proposed in [4] is employed in this work. Suppose in a given referral chain starting from Ag_s, Ag_m is one of Ag_d's referrals to Ag_s. Ag_s derives Ag_m's trustworthiness through *trust propagation* as:

$$Tr_{s,m} = Tr_{s,d} * Tr_{d,m} + (1 - Tr_{s,d}) * (1 - Tr_{d,m}) \quad (3)$$

where $Tr_{s,d}$ is Ag_s's evaluation of Ag_d's trustworthiness, and $Tr_{d,m}$ is Ag_d's evaluation of Ag_m's trustworthiness. Then (3) is computed recursively by Ag_s over all the agents along the referral chain to derive the trustworthiness of the agent at the end of the referral chain.

2.3 Rating Prediction

The new items' ratings are basically derived as weighted sum of the collected opinions, with each opinion-sharing agent's trustworthiness as weight. The collected opinions are adjusted before being used to make predictions if Ag_s has previous experience with the corresponding opinion-sharing

agents. Suppose agent Ag_x shared opinion on the item O_n as $r_{x,n} = v$ and Ag_s has previous experience with Ag_x, then Ag_x's opinion $r_{x,n} = v$ is adjusted to $r'_{x,n} = i$ with a probability n_{iv}/C_v. If Ag_s has no previous experience with Ag_x, Ag_x's opinion is used directly without adjustment.

It is possible that process of opinion collection does not collect any opinion. In this case, Ag_s selects a *pseudo-opinion* as $r'_{x,n} = i$ from each direct neighbor Ag_x with a probability R_i/N (R_i and N are based on Ag_s's experience with each direct neighbor Ag_x).

Ag_s then predicts the new item O_n's rating $Pr_{s,n}$ as:

$$Pr_{s,n} = \frac{\sum_{Ag_x \in \mathcal{A}_{s,n}} Tr_{s,x} * r'_{x,n}}{\sum_{Ag_x \in \mathcal{A}_{s,n}} Tr_{s,x}} \quad (4)$$

Here $\mathcal{A}_{s,n}$ is the set of opinion-sharing agents (including those share pseudo-opinions). $r'_{x,n}$ is the adjusted opinion or pseudo-opinion shared by agent Ag_x. Each agent in $\mathcal{A}_{s,n}$ is assigned a weight which equals to its trustworthiness $Tr_{s,x}$.

3. EXPERIMENTAL RESULTS

A trust-based movie recommendation community has been simulated based on the MovieLens dataset[6]. A similarity-based community based on same dateset has also been setup to study the improvement introduced by the trust-based community. The Pearson correlation efficient is employed as the similarity metric. Agents with negative similarity are ignored when making prediction. Opinion collection with "similarity propagation" was also implemented for the similarity-based community.

The first experiment studied the improvement in terms of coverage without the presence of noisy ratings. Coverage was measured as the percentage of the dataset that the agents were able to predict rating for. The trust-based community manages to achieve a coverage very close to 100%, whereas the maximum coverage achievable with similarity-based community is less than 80%.

The second experiment studied the improvement in terms of prediction accuracy without the presence of noisy ratings. Prediction accuracy was measured as Mean Absolute Error (MAE). MAE is derived by calculating each agent's mean prediction error between the predicted ratings and real opinions, and then averaging all agents' mean errors. The values of U (limit of the length of the referral chains) and D (number of direct neighbors) are tuned to be: $U \in [3, 8]$, and $D \in \{2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 60, 70, 80\}$. Figure 1 compares the MAE achieved by the trust-based and the similarity-based community respectively when $U = 3$. It is observed that the trust-based community generally achieves higher prediction accuracy (i.e. lower MAE) than its similarity-based counterpart with different settings of D. The same trends are observable with other values of U. Due to the space limitation, comparisons of the MAE with other values of U are omitted here.

The third experiment was to study the community's robustness in the presence of noisy ratings. $D = 10$ and $U = 3$ in the third experiment. The robustness was measured in terms of *power of attack* (POA). Let O_t be the targeted item of the noisy ratings, and Ω be the set of agents who are predicting the item O_t's rating. The POA for a particular type of noisy rating ATK is given by:

$$POA(\Omega, O_t, ATK) = \frac{\sum_{a \in \Omega}(|r_{target} - r_{a,t}| - |r_{target} - r''_{a,t}|)}{|\Omega|}.$$

[6]MovieLens dataset is publicly available at http://www.cs.umn.edu/research/GroupLens/data/.

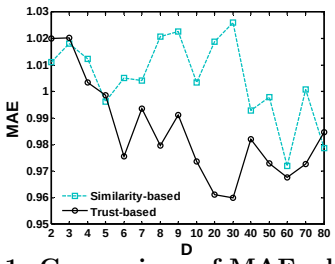


Figure 1: Comparison of MAE when $U = 3$

where $r''_{a,t}$ and $r_{a,t}$ are predictions made with and without the presence of noisy ratings respectively. r_{target} is the target rating, e.g $r_{target} = 5$, or 1 for *push* and *nuke* respectively. A positive value of POA means that the predictions have been forced toward the target rating. Otherwise, POA gives a negative value. Robustness in the presence of different ratios of noisy ratings is shown in Table 2. It is observed that the trust-based community manages to achieve smaller POA than its similarity-based counterpart. It is thus shown that the trust-based community is more robust to the presence of noisy ratings than similarity-based community.

Noisy rating ratios	POA (Push)		POA (Nuke)	
	Trust-based	Similarity-based	Trust-based	Similarity-based
25%	0.0688	0.0410	0.0722	0.0238
50%	0.1105	0.1880	0.1430	0.2145
75%	0.1375	0.2020	0.5305	0.6408
100%	0.1788	0.2211	0.6730	0.7617

Table 2: POA w.r.t. different malicious noisy ratios

4. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a trust-based recommendation community, in which a trust metric has been designed to quantify the degrees of trust an agent should place on others, and a new prediction method has also been proposed.

Work has been reported to introduce trust into the domain of item recommendation to overcome the limitations listed in Section 1, e.g. [3, 5]. However, all of these in existing work do not make clear how an agent can determine the degrees of trust it should place on others. This paper goes beyond existing work in that it proposes a trust metric, with which an agent can practically quantify the degrees of trust it places on others.

Currently, direct neighbors maintained by an agent are not changed during its whole life cycle. It is possible that some direct neighbors become less trustworthy. In this case, those less trustworthy direct neighbors should be replaced by some indirect neighbors that are more trustworthy. The performance of the trust-based community with neighbor replacement will be further investigated.

5. REFERENCES

[1] B. Barber. *The logic and limits of trust*. Rutgers University Press, 1983.

[2] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval Journal*, 4(2):133–151, 2001.

[3] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *Proceedings of International Conference on Cooperative Information Systems 2004*, 2004.

[4] L. Mui. *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 2003.

[5] M. Papagelis, D. Plexousakis, and T. Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. In *Proceedings of iTrust 2005*, pages 224–239, 2005.

[6] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

Learning to Commit in Repeated Games

Stéphane Airiau
Mathematical & Computer Sciences Department
University of Tulsa, USA
stephane@utulsa.edu

Sandip Sen
Mathematical & Computer Sciences Department
University of Tulsa, USA
sandip@utulsa.edu

ABSTRACT

Learning to converge to an efficient, i.e., Pareto-optimal Nash equilibrium of the repeated game is an open problem in multiagent learning. Our goal is to facilitate the learning of efficient outcomes in repeated plays of incomplete information games when only opponent's actions but not its payoffs are observable. We use a two-stage protocol that allows a player to unilaterally commit to an action, allowing the other player to choose an action knowing the action chosen by the committed player. The motivation behind commitment is to promote trust between the players and prevent them from mutually harmful choices made to preclude worst-case outcomes. Our agents learn whether commitment is beneficial or not. Interestingly, the decision to commit can be thought of as expanding the action space and our proposed protocol can be incorporated by any learning strategies used for playing repeated games. We show the improvement of the outcome efficiency of standard learning algorithms when using our proposed commitment protocol. We propose convergence to pareto optimal Nash equilibrium of repeated games as desirable learning outcomes. The performance evaluation in this paper uses a similarly motivated metric that measures the percentage of Nash equilibria for repated games that dominate the observed outcome.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms

Keywords

Repeated Game, Learning, Commitment

1. INTRODUCTION

A rational agent, playing in iterated stage games, should maximize long-term utility. In a two-player, general-sum game, this means that the players need to systematically explore the joint action space before settling on an action combination. Reinforcement learning schemes, and in particular, Q-learning [6], have been widely used in single-agent learning situations. In the context of multi-player games, if one agent plays a stationary strategy, the stochastic game becomes an MDP and techniques like Q-learning can be used to learn to play an optimal response against such a static opponent. When two agents learn to play concurrently, however, the stationary assumption does not hold any longer, and Q-learning is not guaranteed to converge in self play. In such cases, researchers have focused on convergence to Nash equilibrium (NE) in self-play, where each player is playing a best response to the opponent strategy and does not have any incentive to deviate from his strategy.

Convergence is a desirable property in multiagent systems, but converging just to any NE is not the preferred outcome since NE is not guaranteed to be Pareto optimal (an outcome is Pareto optimal if no agent can improve its payoff without decreasing its opponent's payoff). For example, the widely studied Prisoner's Dilemma game (PD) has a single pure strategy NE that is defect-defect, which is dominated by the cooperate-cooperate outcome. A Pareto Optimal outcome may not be appealing to players if that outcome is also not a NE, i.e., there might be incentives for one agent to deviate and obtain higher payoff. For example, each agent has the incentive to deviate from the cooperate-cooperate Pareto optima in PD.

In repeated games, folk theorems[3] ensure that, when players are "patient enough", any payoff dominating a reservation payoff can be sustained by a NE. Hence, in repeated games, there are Pareto Optimal outcomes that are also NE outcomes. It is evident that the primary goal of a rational agent, learning or otherwise, is to maximize utility. Though we, as system designers, want convergence and corresponding system stability, those considerations are necessarily secondary for a rational agent. The question then is what kind of outcomes are preferable for agents engaged in repeated interactions with an uncertain horizon, i.e., without knowledge of how many future interactions will happen. We believe that the goal of learning agents in repeated self-play with an uncertain horizon should be to reach a Pareto-optimal Nash equilibria (PONE).

We are interested in developing mechanisms by which agents can produce PONE outcomes. [4] provides a solution under complete knowledge. This assumption is unrealistic in most cases: opponent valuation is in general intrinsic and private. Moreover, payoff communication opens the door for deceptive behavior. Hence, we believe that not observing the opponent payoff is a more realistic assumption. We are interested in two-person, general-sum games where each agent only gets to observe its own payoff and the action played by the opponent, but opponent's payoff is unknown. Under

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

these conditions, it may be difficult to guarantee convergence to a PONE. In order to compare the performance of different algorithms that are trying to converge to a PONE, we introduce a new metric: given an outcome of the game, the metric relates to the relative number of states dominating the current outcome.

2. COMMITMENT

We now present our proposed commitment protocol that can be added onto any stage game playing algorithm. The motivation behind the protocol is for agents to improve payoffs by building trust via up-front commitment to "cooperating" moves that can be mutually beneficial, e.g., a cooperate move in PD. If the opponent myopically chooses an exploitative action, e.g., a defect move in PD, the initiating agent would be less likely to repeat such cooperation commitments, leading to outcomes that are less desirable to both parties than mutual cooperation. But if the opponent resists the temptation to exploit and responds cooperatively, then such mutually beneficial cooperation can be sustained.

We build on the simultaneous revelation protocol [5]. Agents repeatedly play an $n \times n$ bimatrix game. At each iteration of the game, each player first announces whether or not it wants to commit to an action. If both players want to commit at the same time, one is chosen randomly. If no player decides to commit, then both players simultaneously announce their action, as in the traditional simultaneous play protocol. When one player commits to an action, the other can choose any action given its opponent's action. Each agent can observe which agent actually revealed, and which action the opponent played. In this paper, agents play best response action to opponent's committed action.

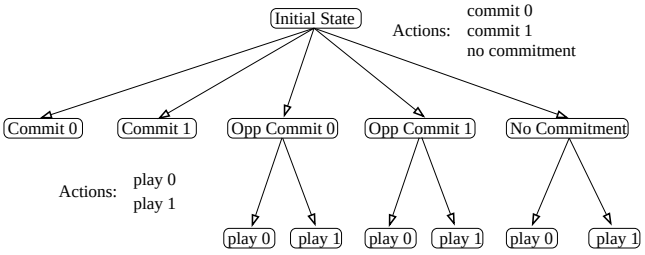


Figure 1: Game tree for a two-action game.

Such games can be represented by game trees, e.g., Figure 1 presents the tree for a two-action game. In the initial state, the agents have $n+1$ actions: it can plan to commit to any of the n actions of the game, or decide not to commit. The transition from the root of the tree depends on the decision of the opponent. The *commit* states are reached when the player commits and the opponent does not, or when both players are willing to commit, but the player wins the toss. From the *commit* state, no further decision is needed, and the payoff received depends on the play of the opponent. When the player decides not to commit, the transition can lead to any one of the n states where the opponent commits or to the state where no players is willing to commit. In both cases, the player has n actions available. From the *opp commit* states, the transition depends only on the current players' decision. From the state where there is no commitment, the transition also depends on the opponent decision. Any multiagent learning algorithm can be used to estimate the utility of different actions, including the commitment actions, from repeated play against an opponent.

Compared to NE outcome of a traditional protocol, the NE outcome with the commit protocol may differ. We hypothesize that, under rational play, the outcome of a game played with the commit protocol is not strictly dominated by the outcome of the game played with the traditional protocol. Assume that players are in a NE of the stage game and are provided the opportunity to commit. A player i commits only when it is beneficial, hence getting a higher payoff. If the other player j is improving due to the commitment, both players improve their respective payoffs. Else, j's payoffs is worse. In this case, j may improve by committing, which might decrease i's payoff. If on average both players' payoffs decrease, the players will ultimately learn not to reveal. When i commits and j cannot improve its payoff by committing, e.g. committing to any action yields a lesser payoff, the players reached a different equilibrium (i improves and j is worse off but there is no dominance). In any case, players should only benefit from the commit protocol.

3. RESULTS

We compared the use of the commit protocol with the traditional protocol of simultaneous play on various set of matrices. Any traditional algorithm for game playing can be used in combination with the commit protocol. We chose to use WoLF-PHC[1] (Win or Learn Fast - policy hill climbing) [1] as the learning algorithm. The algorithm learns mixed strategy and is guaranteed to converge to a NE in a 2-person, 2-actions repeated game.

3.1 Metric

To compare the equilibrium outcomes, we can use the concept of dominance. However, when there is no dominance between the outcomes, additional metrics are needed. Investigating the sum of the payoff of the player (a measure of the social welfare), or the product of the payoff (a measure of fairness) provides insight to the equilibrium properties of the learning algorithms. Another approach is to consider the number of equilibria that dominate the current equilibrium: the fewer outcomes that dominate the current outcome, the closer this outcome is to a Pareto Optimum. The folk theorems [3] ensure that when an outcome dominates the minimax outcome, it can be sustained by a NE of the repeated game. For an outcome x, let $d(x)$ denotes the area containing all points that dominates x in the payoff space. If $d(x) = 0$ and x dominates the minimax outcome, then x is a PONE.

Definition 1. Performance metric of an equilibrium outcome x: $\delta(x) = \frac{d(x)}{d(x_{mm})}$ where x_{mm} is the minimax outcome.

$\delta(x)$ represents the proportion of NE outcomes of the repeated games that dominates x. The smaller $\delta(x)$, the better the outcome x is with respect to convergence to a PONE. When one outcome x dominates an outcome y, $\delta(x) < \delta(y)$. The opposite is not true: when there is no dominance between x and y, $\delta(x)$ may be less, equal, or greater than $\delta(y)$.

3.2 Testbed of 2x2 conflicted games

We first use a neutral but extensive testbed of games introduced by Brams in [2]: the testbed is composed of all possible conflicting situations that can occur in a two-action two-player game with a total preference order over the four outcomes of the game. This testbed represents a wide variety of situations, including often-studied games like PD, the chicken game, battle of the sexes, etc. There is no game where agents can simultaneously obtain their most preferred outcome, which implies that each game represents a

[1]WoLF-PHC settings: $\alpha(t) = \frac{1}{10+\frac{t}{100}}$, $\delta_W = \frac{1}{10+t}$, $\delta_L = 4\delta_W$. The games were played over 10,000 iterations, and results were averaged over 40 runs.

conflicting situation. There are 57 structurally different 2x2 conflict games (no two games are identical by renaming the actions or the players). In five games, the equilibrium reached by WoLF(commit) strictly dominates the equilibrium reached with WoLF. Three of them are games where the NE is dominated. The remaining two are the games where the NE is a mixed strategy NE dominated by a pure strategy. In nine other games, the outcome obtained by WoLF(commit) is different than the outcome of the NE of the stage game played with the traditional protocol, but there is no dominance (one player gains and the other looses). We found that the use of the commit protocol fails to produce a Pareto optimal solution in only two games, one of which is the prisoner's dilemma game.

3.3 Results on randomly generated matrices

As shown in the previous experiments, the structure of some games can be exploited by the commit protocol to improve the payoff of both players. To evaluate the effectiveness of the protocol on a more general set of matrices, we ran experiments on randomly generated matrices as in [5]. We generated 1000 matrices of sizes 3x3, 5x5 and 7x7. Each matrix entry is sampled from a uniform distribution in $[0, 1]$. We compare the outcome of WoLF(commit) and WoLF. In the top plot of Figure 2, we plot different areas: the average area containing all the outcome of NE (i.e. dominating the minimax outcome), the area that dominates the outcome of the traditional and the commitment protocol. We first observe that the minimax outcome is dominated by more outcomes for larger games, i.e. the space of NE is larger. When we compare with the area that dominates the outcome found by WoLF we find that the outcome with the protocol with commitment is dominated by less outcomes, and the difference increases with the game size. In the bottom plot of Figure 2, we plot our δ metric that provides the percentage of sustainable NE of the repeated game that dominates the outcome of the algorithm. The chart indicates that the outcome obtained with protocol with commitment is dominated by at most 10% of the possible NE, when the outcome of the traditional simultaneous game is dominated by 3 times more NE. This suggests that the commitment protocol produces more efficient equilibrium than the traditional simultaneous game protocol.

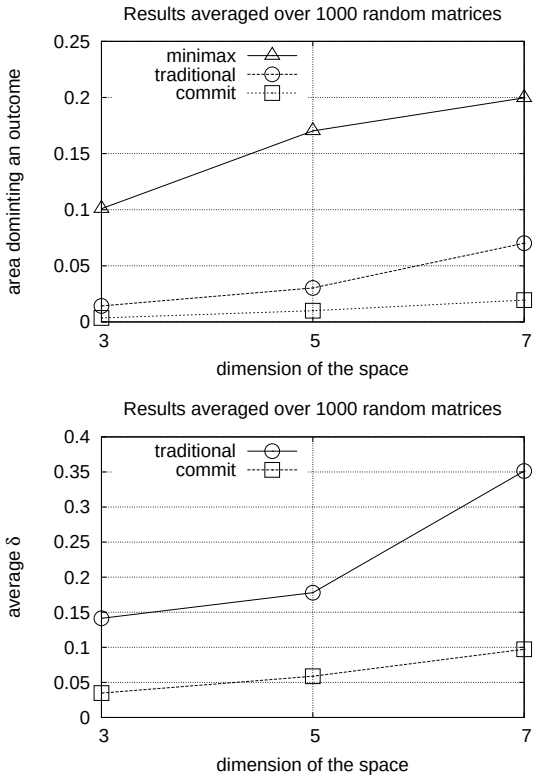


Figure 2: Results over randomly generated matrices.

4. CONCLUSION AND FUTURE WORK

In this paper, we built on a previous algorithm from [5] with the goal of producing PONE outcomes in repeated single-stage games. We propose a metric that can be used to measure the quality of an outcome: it represents the relative number of Nash equilibria of the repeated game that dominate the outcome reached. Under the assumption that the opponent payoff matrix is unknown, it might be difficult to ensure convergence to a PONE. Our proposed metric is helpful in comparing the relative efficiency of different outcomes.

We experiment with two-player two-action general-sum conflict games where both agents have the opportunity to commit to an action and allow the other agent to respond to it. The opportunity of revealing its action should not be seen as making a concession to the opponent, but rather as a means to explore the possibility of mutually beneficial outcomes. Any learning algorithm can be augmented to incorporate the commit protocol, which improves the payoffs in most cases: we empirically show that our protocol improve the payoffs obtained by WoLF-PHC in a variety of games. The experiments also show shortcomings of the current commitment protocol in that it fails to reach PONE outcomes: the primary reason for this is that a player responds to a commitment with a myopic best response.

We assume that a player does not know the payoff matrix of the opponent, which makes it difficult to estimate whether the equilibrium reached is acceptable for both players. In particular, there are situations where not playing a best response to a committed action can be beneficial for both players. To find a non-myopic equilibrium, an agent should not be too greedy! Currently, the agents are learning only their own payoff, and learn to play a best response to a committed action. We are working on learning action-utility estimates that incorporates an estimate of the preference of the opponent in the game tree presented in Figure 1. We expect that the agents will be able to more consistently discover states beneficial for both learners, and thereby converge to PONE outcomes.

Acknowledgment: US National Science Foundation award IIS-0209208 partially supported this work.

5. REFERENCES

[1] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.

[2] S. J. Brams. *Theory of Moves*. Cambridge University Press, Cambridge: UK, 1994.

[3] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, Cambridge, MA, 1991.

[4] M. L. Littman and P. Stone. A polynomial-time nash equilibrium algorithm for repeated games. *Decision Support Systems*, 39:55–66, 2005.

[5] S. Sen, S. Airiau, and R. Mukherjee. Towards a pareto-optimal solution in general-sum games. In *Proceedings of the Second International Joint Conference On Autonomous Agents and Multiagent Systems*, pages 153–160, 2003.

[6] C. J. C. H. Watkins and P. D. Dayan. Q-learning. *Machine Learning*, 3:279 – 292, 1992.

Effect of deceptive referrals on system stability

Ikpeme Erete
ikpeme-erete@utulsa.edu
University of Tulsa

Teddy Candale
teddy-candale@utulsa.edu
University of Tulsa

Sandip Sen
sandip@utulsa.edu
University of Tulsa

ABSTRACT

We study the problem of agents attempting to find quality service providers in a distributed environment. While referrals from other agents can be used to locate high-quality providers, referrers may be malicious and provide incorrect referrals to reduce traffic to their preferred service providers. Whereas apparently it would seem that such deceptive referrals can disrupt system stability, we observe that homogeneous groups of deceptive referrals converge faster to stable agent distributions over service providers compared to homogeneous groups of truthful referrers. We conjecture that deceptive referrers can unwittingly reduce the entropy, a measure of volatility, of the system as the recipient of a bad referral may not be inclined to move even if it is not satisfied with its current service providers. Additionally, we observe that mixed groups of truthful and deceptive referrers converge faster to stable distributions compared to homogeneous group of truthful referrers. These results highlight the unexpected positive effect of deceptive agents in stabilizing a population.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Coherence and coordination, Multiagent systems, Intelligent agents*

General Terms

Performance, Experimentation

Keywords

Referral system, deceptive referral, satisficing distribution

1. INTRODUCTION

We investigate the usefulness of referrals from other agents in locating service providers with satisfactory performance where provider performance depends on its workload. In particular, we are interested in quick convergence of such systems to stable agent distributions over the providers where each agent is satisfied with the performance of the provider it is currently using. In previous work, we determined that truthful referral helps to increase the speed of convergence as opposed to systems without referrals. But agents may also deliberately provide misleading referrals to, among other reasons, prevent "crowding" at their favorite providers. This paper aims at determining the effect such deception would have on the system convergence and stability.

While ideally speaking agents may aspire for optimal satisfaction levels from service providers selected for performing an assigned task, dynamic, partially known, and open environments can render the realization of this ideal behavior improbable. Possible sources of inefficiencies include noisy, variable feedback about provider performance as the environment is at best partially observable, e.g. all factors affecting performance are not directly observable. Besides, an agent is unable to accurately assess the impact of its own decisions, including choice of service providers and making referrals, on its environment. In this paper, we evaluate the effects of deceptive referrals and if it can, counter-intuitively, mitigate the lack of accurate information available to agents and can maintain or even improve convergence rates to stable configurations.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. EXPERIMENTAL FRAMEWORK

Our environment is formally defined as follows:
$\mathcal{E} =< \mathcal{A}, \mathcal{R}, perf, L, S, \Gamma >$ where:

- $\mathcal{A} = \{a_k\}_{k=1..K}$ is the set of agents
- $\mathcal{R} = \{r_n\}_{n=1..N}$ is the set of providers.
- $f: \mathcal{R} \times \mathbb{R}_+ \rightarrow [0,\ 1]$, intrinsic performance function of a provider given total load.
- $L : \mathcal{A} \rightarrow \mathbb{R}_+$, load function for the agents.
- $S: \mathcal{A} \times [0, 1] \rightarrow [0,\ 1]$, satisfaction function of agents.
- $\Gamma = \{\gamma_1, \ \ldots, \ \gamma_K\}$, the set of satisfaction thresholds of agents.

On each day, agent a_k is assigned a load $L(a_k)$. Initially, each agents know the set of service providers that can handle their tasks but not the intrinsic capabilities, $f(r_n, \cdot)$, of any provider r_n or their respective load capacity. The above formulation is consistent with our previous work on referrals [1].

Our primary objective is for every agent in the system to obtain a satisfaction above its satisfaction threshold, thereby ensuring a stable system. With no incentive to move, the system no longer oscillates and we attain an equilibrium state in finite time. Such a distribution is formally characterized as follows:

DEFINITION 1 (Γ-ACCEPTABLE DISTRIBUTION). *A distribution $D = \{\mathcal{A}_n\}_{n=1..N}$ is said to be* **Γ-acceptable distribution** *iff*

$$\forall n,\, a_k \in \mathcal{A}_n \Longrightarrow S\left(a_k,\, f\left(r_n,\, \sum_{a \in \mathcal{A}_n} L(a)\right)\right) \geq \gamma_k$$

3. INFLUENCE OF INERTIA ON CONVERGENCE SPEED

In uncontrolled systems, agents receive highly variable satisfactions. A system at a distribution close to Γ-acceptable distribution will have the tendency to evolve to a worse distribution and vice versa. Intuitively, a distribution where almost everyone is satisfied contains very few under-used or over-used providers and the rest are occupied by the right number of agents. Those under-used providers, $\mathcal{R}_u$, are very attractive. Consequently, agents will be inclined to move to them, which leads the system to a distribution where providers in $\mathcal{R}_u$ will be overcrowded. Indeed, the performance of the system oscillates between desirable values and highly undesirable ones. We believe this problematic effect can be mitigated by increasing the inertia in the system, where the inertia of an agent is its inclination to persist with a service provider it is not satisfied with and is globally measurable by an inverse function of the number of agents moving at any given time, K_{move}.

The work by Candale and Sen [1] showed that when a high number of agents switch resources simultaneously the performance of the system cannot stabilize and varies between high and low values. As a result, agents receive variable feedback from the environment leading to inaccurate learning. They also showed that when the number of agents moving simultaneously remains low, the system stabilizes quickly. We experiment with the following set of agent types:

AgentTRnD: While deciding on which service provider to use an agent can use its past experience and referrals from selected agents. A referral consists of a name of a provider and an estimation of its quality. While picking an agent to ask for referral it selects an agent, with a probability proportional to the quality of past referrals from that agent, from a set of agents whose expected quality of referral is greater than its satisfaction threshold. When an agent, a_k, fails to get an acceptable referral from other agents or from its knowledge of providers, it resorts to systematic exploration of providers with an exploration probability, α_k. The consequence of excessive exploration is a reduction in the inertia of the system and can lead to system instability.

AgentTRwD: This agent is identical to the previous one except the fact that agents always deceive. They refer the same provider as *AgentTRnD*, but alter the true estimation, *es*. More precisely, if *es* is greater than their satisfaction threshold then they report a lower estimate, and otherwise report a higher estimate.

4. EXPERIMENTAL RESULTS

The environment is setup with the exact number of service providers needed to satisfy all the agents in it.We ran experiments with environments having a population of 100 agents each and results averaged over 50 runs. The environments comprised of purely homogeneous agents, truthful and deceptive, and a mixed population of both agents in equal ratio. Agents have identical properties such as coefficient of exploration, satisfaction threshold, and the load function.

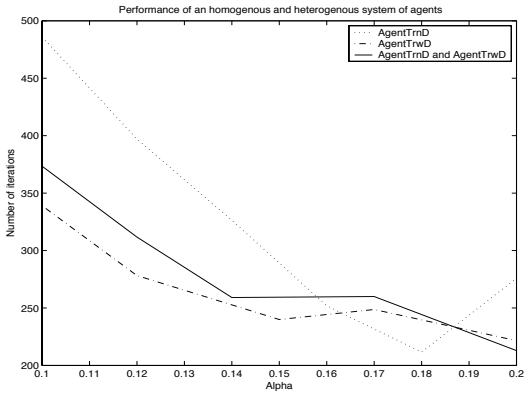


Figure 1: Speed of convergence with varying α: 100 Agents, 50 providers and $\gamma_k = 0.7$.

4.1 Influence of alpha (α)

We observe the speed of convergence of our system by varying coefficient of exploration, α (see Figure 1). We present the following observations:

1. For all agent populations the speed of convergence improves when α increases. For small values of α system convergence is delayed by a slow resource discovery process [1]. For $\alpha > 0.2$, however, exploratory actions become too frequent and increase convergence times. The effect of referrals is more pronounced for smaller values of α. With infrequent exploration, agents have to depend more on referrals to find satisfactory resources.

2. A significant, counter-intuitive result is that the presence of deceptive agents in the population (homogeneous or mixed) leads to better speed of convergence than in homogeneous population of truthful agents. Truthful referrals affect agents using the referred providers, as they will see their satisfactions decrease and then be encouraged to switch resources, which leads to system instability as illustrated in Figure 2. Deception, however, helps an agent maintain the load of its favorite provider at a low level by discouraging probable newcomers. Another key reason is that with deceptive referrals from other agents, it is more likely that an unsatisfied agent may be temporarily led to believe that it will not receive better satisfaction at

other resources. Thus it may choose to not change its provider, thereby decreasing system volatility and ultimately facilitating convergence. Unsatisfied agents, however, will not be persuaded not to change for long, and the system converges steadily.

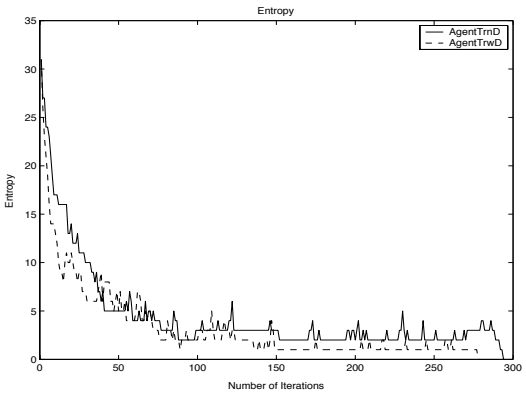


Figure 2: Entropy of truthful and deceptive agents: 100 agents, 50 resources.

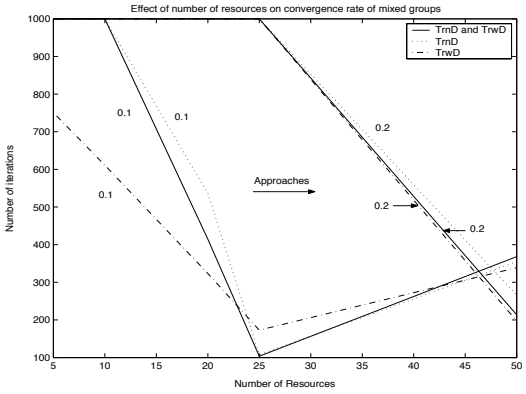


Figure 3: Speed of convergence with varying resources: 100 Agents, $\gamma_k = 0.7,\ \alpha = 0.1,\ , 0.2$.

4.2 Influence of the number of resources

We also investigated the speed of convergence by varying the number of resources (see Figure 3). We observe that deception has a positive effect on the speed of convergence when the numbers of resources is small. With smaller number of resources, it is more likely that agents are dissatisfied with their service providers, leading to greater K_{move}. Deceptive referrals help reduce that for reasons cited above.

5. RELATED WORK

Our domain was inspired by Yu and Singh [4] and Sen,Arora and Roychowdhury [3]. In both work they showed the short term cost of processing referral request and that information can negatively impact agent coordination to find a balanced distribution among resources respectively.

We believe that a more comprehensive understanding of system behavior can be obtained by studying the number of simultaneously moving agents, K_{move}. Our approach provides more detailed characterization of the system, but is consistent with general conclusions from Rustogi & Singh's [2] study as inertia can be used as a mechanism for controlling K_{move}. Rustogi & Singh [2] also claim that performance is remarkably improved when accepting imprecision.

6. CONCLUSION AND FUTURE WORK

We investigated the effect of the presence of deceptive agents in a population of referrer agents trying to coordinate their selection of service providers. Surprisingly, deception does not harm the system but, in general, helps to enhance the speed of convergence. Agents, by the use of deception, can keep the load of the provider they are currently using at a reasonable level. Besides, deception promotes system stability as it can discourage even unsatisfied agents from moving for some time.

In our future work, we plan on investigating if deceptive agent can still improve system convergence of a truthful referral system with differing satisfaction level and load function. We hope to find ways to achieve an improved convergence in such systems.

Acknowledgments: This work has been supported in part by an NSF award IIS-0209208.

7. REFERENCES

[1] T. Candale and S. Sen. Effect of referrals on convergence to satisficing distributions. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 347–354,New York, NY, USA,2005. ACM Press.

[2] S. K. Rustogi and M. P. Singh. Be patient and tolerate imprecision: How autonomous agents can coordinate effectively. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 512–517, 1999.

[3] S. Sen, N. Arora, and S. Roychowdhury. Effects of local information on group behavior. In *Proceedings of the Second International Conference on Multiagent Systems*, pages 322–329, Menlo Park, CA, 1996. AAAI Press.

[4] B. Yu and M. P. Singh. Searching social networks. In *Proceedings of the Second Intenational Joint Conference on Autonomous Agents and Multiagent Systems*, pages 65–72, New York, NY, 2003. ACM Pres.

Agent programming

Goal-Oriented Modularity in Agent Programming

M. Birna van Riemsdijk[1] Mehdi Dastani[1] John-Jules Ch. Meyer[1] Frank S. de Boer[1,2]
Utrecht University[1]
CWI, Amsterdam[2]
The Netherlands
[birna | mehdi | jj | frankb]@cs.uu.nl

ABSTRACT

Modularization is widely recognized as a central issue in software engineering. In this paper we address the issue of modularization in cognitive agent programming languages. We discuss existing approaches to modularity in cognitive agent programming. Then, we propose a new kind of modularity, i.e., goal-oriented modularity, which takes the goals of an agent as the basis for modularization. Further, we present a formal semantics of goal-oriented modularity in the context of the 3APL agent programming language.

Categories and Subject Descriptors

I.2.5 [**Artificial Intelligence**]: Programming Languages and Software; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, languages and structures*; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages; D.3.1 [**Programming Languages**]: Formal Definitions and Theory; D.2 [**Software Engineering**]

General Terms

Theory, Languages

Keywords

Agent programming languages, modularity, semantics, declarative goals

1. INTRODUCTION

Modularization is widely recognized as a central issue in software engineering [11, 9, 2]. A system which is composed of modules, i.e., relatively independent units of functionality, is called modular. A programming language which adheres to this principle of modularization supports the decomposition of a system into modules. The principle lies, e.g., at the basis of procedural programming and object-oriented programming, in which respectively (libraries of) procedures and classes form the modules.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

An important advantage of modularity in a programming language is that it can increase the *understandability* of the programs written in the language. The reason is that modules can be separately understood, i.e., the programmer does not have to oversee the entire workings of the system when considering a certain module. Further, modularization enables *reuse* of software, since modules can potentially be used in different programs or parts of a single program. Finally, we mention an important principle which any kind of modularity should stick to, namely the principle of *information hiding*. This means that information about a module should be private to the module, i.e., not accessible from outside, unless it is specifically declared as public. The amount of information declared public should typically be relatively small. The idea behind information hiding is that a module can be changed (or at least the non-public part of a module), without affecting other modules.

In this paper we address the issue of modularization in *cognitive agent programming languages*. Cognitive agents are agents endowed with high-level mental attitudes such as beliefs, goals, plans, etc. Several programming languages and platforms have been introduced to program these agents, such as 3APL [10, 6], AgentSpeak(L) [14, 3], JACK™ [20], Jadex [13], etc. Some of these languages incorporate support for modularization, which we discuss in some detail in section 2.1.

Our contribution is the proposal of a new kind of modularity, i.e., *goal-oriented modularity*, which is, as we will argue, particularly suited for agent programming languages (section 2). Further, we present a *formalization* of goal-oriented modularity in the context of the 3APL programming language (section 3). We conclude the paper with directions for future research in section 4.

2. GOAL-ORIENTED MODULARITY

In this section, we explain the general idea of goal-oriented modularity. First, we discuss which kinds of modularity are present in today's cognitive agent programming languages and platforms (section 2.1). Then, we explain the general idea of goal-oriented modularity (section 2.2).

As for programming in general, modularization is also an important issue for agent programming. One could argue that the agent paradigm provides inherent support for modularity, since a complex problem can be broken down and solved by a team of autonomous agents. Constructing a team of agents to solve a problem rather than creating a single more complex agent, might however not always be the appropriate approach. The team approach will likely gener-

ate significant communication overhead, and having several independent agents can make it difficult to handle problems that require global reasoning [5, 4]. Following [5, 4], we thus argue that modularization is important also at the level of individual agents.

2.1 Related Work

With regard to cognitive agent programming languages such as 3APL and AgentSpeak(L), one could argue that these languages support modularization: an agent is typically composed of a number of components such as a belief base, a plan or intention base, a plan library, a goal or event base, etc. These components however do not provide the appropriate modularization, since their workings are closely intertwined and therefore they cannot be considered as relatively independent units of functionality. Cognitive agent programming languages thus need more support for structuring the internals of an agent, in order for them to deserve the predicate "modular".

One possible approach to addressing this issue of modularity of cognitive agent programming languages has been proposed by Busetta et al. in [5]. In that paper, the notion of *capability* was introduced and its implementation in the JACK cognitive agent programming language was described.

JACK extends the Java™ programming language in several ways, such as by introducing constructs for declaring cognitive agent notions like beliefs, events, plans, etc. Events are used to model messages being received, new goals being adopted, and information being received from the environment. Plans are used to handle events, i.e., if an event is posted, the reasoning engine tries to find an appropriate plan which has this event as its so-called triggering condition.

A capability in JACK is a cluster of components of a cognitive agent, i.e., it encapsulates beliefs, events (either posted or handled by the capability), and plans. Examples of capabilities given in [5] are a buyer and seller capability, clustering functionality for buyer and seller agents, respectively. A capability can import another capability, in which case the latter becomes a sub-capability of the former. Using the importation mechanism for capabilities, the beliefs of a capability can also be used by its super-capability, that is, if they are explicitly imported by the latter. Also, the beliefs of a capability can be used by its sub-capabilities if they are explicitly declared as exported (and if they are also imported by the sub-capabilities). For events, a similar mechanism exists, by means of which events posted from one capability can also be handled by plans of its sub- and possibly super-capabilities.

The notion of capability as used in JACK has been extended in the context of the Jadex platform [4]. Jadex is a cognitive reasoning engine which is built on top of the Jade [1] agent platform. A Jadex agent has beliefs, goals, plans, and events. Like capabilities in JACK, a Jadex capability clusters a set of beliefs, goals, plans, and events. Its most important difference with the notion of capability as used in JACK, is the fact that a general import/export mechanism is introduced for all kinds of elements of a capability, i.e., the mechanism is the same for beliefs, events, etc.

Another approach which could be viewed as addressing the issue of modularity in cognitive agent programming languages, has been proposed in the context of 3APL in [7]. In that paper, a formalization of the notion of a role is given. Similar to capabilities, a role clusters beliefs, goals, plans, and reasoning rules. The usage of roles at run-time however differs from that of capabilities. In the cited paper, a role can be enacted and deacted at run-time, which is specified at the level of the 3APL reasoning engine or deliberation cycle. If a role is enacted, the agent pursues the goals of the role, using the plans and reasoning rules of the role.[1] Further, the agent has a single belief base, and if a role is enacted, the beliefs associated with the role are added to the agent's beliefs. This is in contrast with JACK and Jadex where beliefs are distributed over capabilities, and can only be used in other capabilities if explicitly imported and exported. Also, only one role at the time can be active. This is in contrast with the way capabilities are used, since a JACK or Jadex agent can in principle use any of its capabilities at all times.

2.2 Our Proposal

While the approaches to modularization as described in section 2.1 are interesting in their own right, we propose an alternative which we argue to be particularly suited for cognitive agent programming languages. As the name suggests, goal-oriented modularity takes the goals of an agent as the basis for modularization. The idea is, that *modules encapsulate the information on how to achieve a goal*, or a set of (related) goals. That is, modules contain information about the plans that can be used to achieve a (set of) goal(s). At run-time, the agent can then dispatch a goal to a module, which, broadly speaking, tries to achieve the dispatched goal using the information about plans contained in the module.

This mechanism of dispatching a goal to a module can be used for an agent's top-level goals[2], but also for the *subgoals* as occurring in the plans of an agent. Plans are often built from actions which can be executed directly, and subgoals which represent a state that is to be achieved before the agent can continue the execution of the rest of the plan. An agent can for example have the plan to take the bus into town, to achieve the goal of having bought a birthday cake, and then to eat the cake.[3] This goal of buying a birthday cake will have to be fulfilled by executing in turn an appropriate plan of for example which shops to go to, paying for the cake, etc., before the agent can execute the action of eating the cake.

Before continuing, we remark that goals and subgoals in this paper are *declarative* goals, which means that they describe a state that is to be reached. This is in contrast with procedural goals, which are directly linked to courses of action. We refer to [21] for a discussion on declarative and procedural goals in general, and to [19, 16] for more background on declarative goals. Further, [17] explores semantics of subgoals, and relates declarative and procedural interpretations of subgoals.

Returning to our treatment of goal-oriented modularity, the idea is thus that agents try to achieve subgoals of a plan by dispatching the subgoal to an appropriate module, i.e., by *calling* a module. The module should then define the plans that can be used for achieving the (sub)goal. If a module is

[1]We simplify somewhat, since the details are not relevant for the purpose of this paper.

[2]Top-level goals are goals that the agent, e.g., has from start-up, or that it for example has adopted because of requests from other agents, etc.

[3]Assuming that both taking the bus into town and eating cake are actions that can be executed directly.

called for a goal, these plans are tried one by one until either the goal is achieved, or all plans have been tried. Control then returns to the plan from which the module was called. Depending on whether the subgoal is achieved or not upon returning from the module, the plan respectively continues execution, or fails. If the plan fails, another plan is selected (if it exists), for achieving the goal for which the failed plan was selected, etc.

2.3 Discussion

An advantage of our proposal is the *flexible* agent behavior with respect to handling plan failure, which comes with the usage of declarative goals. As argued in [21, 16], the usage of declarative goals facilitates a decoupling of plan execution and goal achievement. If a plan fails, the goal that was to be achieved by the plan remains a goal of the agent, and the agent can select a different plan to try to achieve the goal. While these ideas regarding declarative goals are not new, we contribute by proposing to use modules to *encapsulate* this mechanism for achieving goals by trying different plans. We thus exploit the advantages of declarative goals for obtaining modularization. Since in our view goals, proactiveness and flexible behavior are at the heart of (cognitive) agenthood, we argue that goal-oriented modularity, which builds on these notions, is a kind of modularity *fitting* for cognitive agents.

Comparing goal-oriented modularity with capabilities, we point out two major differences. Firstly, in the case of capabilities, there is no notion of *calling* a capability for a subgoal, thereby passing control to another capability. An event (or subgoal) posted from the plan of one capability, will be handled by the plans of this capability itself. That is, unless the capability imports other capabilities, in which case the plans of these other capabilities are *added* to the set of plans considered for handling the posted event (given appropriate import and export declarations of events). This is thus in contrast with goal-oriented modularity, where, in case of calling a module, *only* the plans of the called module are considered. These plans are not added to, e.g., some other set of plans, thereby preventing possibly unforeseen interactions between these plans. One could thus consider the idea of goal-oriented modularity to provide a higher degree of modularity or encapsulation of behavior at run-time, compared with the way in which capabilities are used. As we will explain in section 3, this is especially advantageous in the case of 3APL.

Secondly, modules in goal-oriented modularity contain only information about the plans which can be used to achieve certain goals. This is in contrast with capabilities, which can also contain beliefs. The idea for goal-oriented modularity is that the agent has one global belief base, rather than defining beliefs inside modules. When using capabilities, beliefs can by contrast be distributed over these capabilities, and only the beliefs of a certain capability can be accessed from this capability.

While from a software engineering perspective it might be convenient to be able to define beliefs inside capabilities (or modules), it can be considered less intuitive from a conceptual point of view. When testing, e.g., from within a plan whether the agent believes something, one could argue that the agent would have to consider *all* of its beliefs, rather than just the ones available in the capability. Also, if logical reasoning is involved as in the case of 3APL, it is more intuitive to let an agent have just one belief base. Consider for example that the formula p is in the belief base of one module, and that $p \rightarrow q$ is in the belief base of another. When testing whether q holds from the latter module, one would probably want the test to succeed.

Nevertheless, in JACK and Jadex, beliefs can be used in other capabilities if they are imported and exported in appropriate ways. Also, beliefs (or beliefsets) in those languages are effectively databases which store elements representing the beliefs of the agent. The definition of beliefs inside a capability could thus be viewed as the specification of a *part* of the larger database (or set of databases) comprising the total set of beliefs of the agent. From within a capability, an agent can then only refer to the part of its beliefs defined in this capability. It is then up to the programmer to make sure that the beliefs of a capability are the only ones relevant for the plans of this capability. The possibility of storing beliefs inside modules in a way which is somewhere inbetween the current proposal and the way it is done for capabilities, is discussed in section 4.

Comparing goal-oriented modularity with the notion of roles as used in [7], we remark the following. As in the case of capabilities, roles can, in contrast with modules, not call each other. Also, beliefs can be part of the definition of a role. However, a role does not have its own beliefs once it is enacted at run-time. If a role is enacted, its beliefs are added to the global belief base of the agent. This is more in line with goal-oriented modularity, but it is in contrast with the way capabilities are used. Further, contrary to roles, modules do not have goals. That is, a goal can be dispatched to a module, but goals are not part of the definition of a module.

3. GOAL-ORIENTED MODULARITY IN 3APL

In this section, we make the idea of goal-oriented modularity as presented in section 2 precise. In particular, we present a formalization in the context of 3APL. Nevertheless, we stress that we consider the general idea of goal-oriented modularity to be suited for cognitive agent programming languages in general, rather than for 3APL only.

3.1 Syntax

We build on a single agent and propositional version of 3APL, which comes closest to the one presented in [19]. We refer to [6, 8] for first order, multi-agent, and otherwise extended versions. This simplified version of 3APL is sufficient to present the ideas of the formalization of goal-oriented modularity. We conjecture that extending the formalization to a richer version of 3APL will not give rise to fundamental difficulties.

A 3APL agent in [19] has beliefs, a plan, and goals. Beliefs represent the current state of the world and information internal to the agent. Goals represent the desired state of the world, and plans are the means to achieve the goals. Further, a 3APL agent has rules for selecting a plan to achieve a certain goal given a certain belief, and it has rules for revising its plan during execution. We use these ingredients for our formalization of goal-oriented modularity.

Throughout this paper, we assume a language of propositional logic $\mathcal{L}$ with negation and conjunction. The symbol $\models$ will be used to denote the standard entailment relation for

$\mathcal{L}$. Further, we assume a belief query language $\mathcal{L}_\mathsf{B}$ with typical element β. A belief query can be used to test whether the agent has a certain belief. We assume an entailment relation on belief bases and belief query formulas, denoted by $\models_{\mathcal{L}_\mathsf{B}}$.[4]

Below, we define the language of plans Plan. A plan is a sequence of basic actions and module calls. Informally, basic actions can change the beliefs of an agent if executed. A module call is of the form $m(\phi)$, where m is the name of a module (to be defined in definition 4), and ϕ is a propositional formula representing the goal which is dispatched to module m.

Further, we define an auxiliary set of plans Plan′ with the additional construct $m(\phi)\downarrow$. This construct is used in the semantic definitions for recording whether module m has already been called for goal ϕ (see section 3.2 for further explanation). It should not be used by the agent programmer for programming plans, which is why we define two different plan languages.

Definition 1. *(plan) Let* BasicAction *with typical element* a *be the set of basic actions, let* ModName *with typical element* m *be a set of module names, and let* $\phi \in \mathcal{L}$. *The set of plans* Plan *with typical element* π *is then defined as follows.*

$$\pi ::= a \mid m(\phi) \mid \pi_1; \pi_2$$

We use ϵ *to denote the empty plan and identify* $\epsilon;\pi$ *and* $\pi;\epsilon$ *with* π. *The set* Plan′ *is defined as follows.*

$$\pi ::= a \mid m(\phi) \mid m(\phi)\downarrow \mid \pi_1; \pi_2$$

This simple language of plans could be extended with, e.g., if-then-else and while constructs as was done in [19], but the language as given here suffices for the purpose of this paper.

3APL uses so-called plan generation rules for selecting an appropriate plan for a certain goal. A plan generation rule is of the form $\phi \mid \beta \Rightarrow \pi$. This rule represents that it is appropriate to select plan π for goal ϕ, if the agent believes β.[5]

Definition 2. *(plan generation rule) The set of plan generation rules* $\mathcal{R}_\mathsf{PG}$ *is defined as follows:* $\mathcal{R}_\mathsf{PG} = \{\phi \mid \beta \Rightarrow \pi \ : \ \phi \in \mathcal{L},\ \beta \in \mathcal{L}_\mathsf{B},\ \pi \in \mathsf{Plan}\}$.[6]

The other type of rule which comes with 3APL, is the rule for plan revision. Plan revision rules are used to revise an agent's plan during execution. A plan revision rule $\pi_h \mid \beta \rightsquigarrow \pi_b$ represents that in case the agent believes β, it can replace the plan π_h by the plan π_b.

Definition 3. *(plan revision rule) The set of plan revision rules* $\mathcal{R}_\mathsf{PR}$ *is defined as follows:* $\mathcal{R}_\mathsf{PR} = \{\pi_h \mid \beta \rightsquigarrow \pi_b \ : \ \beta \in \mathcal{L}_\mathsf{B},\ \pi_h, \pi_b \in \mathsf{Plan}\}$.

[4]See, e.g., [19] for an example of a belief query language, in which one can express queries such as $\mathbf{B}(p)$ and $\neg\mathbf{B}(p \wedge q)$ for testing whether the agent believes p or does not believe $p \wedge q$, respectively.

[5]Note that it is up to the programmer to specify appropriate plans for a certain goal. 3APL agents can thus be viewed as a kind of reactive planning agents.

[6]We use the notation $\{\ldots \ : \ \ldots\}$ instead of $\{\ldots \mid \ldots\}$ to define sets, to prevent confusing usage of the symbol $\mid$ in this definition and definition 3.

Plan generation rules capture the information about which plan can be selected for which goal, and plan revision rules can be used during the execution of a plan. These rules thus specify the information on how to achieve goals, and therefore we propose to have these rules make up a module, as specified below. It is important to remark that non-modular versions of 3APL have one set of plan generation rules, and one set of plan revision rules, rather than an encapsulation of these into modules.

Definition 4. *(module) A module is a tuple* $\langle m, \mathsf{PG}, \mathsf{PR}\rangle$, *consisting of a module name* m, *a finite set of plan generation rules* $\mathsf{PG} \subseteq \mathcal{R}_\mathsf{PG}$, *and a finite set of plan revision rules* $\mathsf{PR} \subseteq \mathcal{R}_\mathsf{PR}$.

The mechanism of calling a module is formalized using the notion of a stack. This stack can be compared with the stack resulting from procedure calls in procedural programming, or method calls in object-oriented programming. During execution of the agent, a single stack is built (see definition 7). Each element of the stack represents, broadly speaking, a module call.

To be more specific, each element of the stack is of the form $(\phi, \pi, \mathsf{PG}, \mathsf{PR})$, where ϕ is the goal for which the module was called, π is the plan currently being executed in order to achieve ϕ, and PG and PR correspond with the plan generation and plan revision rules of the module which was called for achieving ϕ. Rather than using the name of the module to refer to the relevant plan generation and plan revision rules, we copy these rules into the stack element. The reason is that we want to remove plan generation rules if they are tried once. This will be explained further in section 3.2.

Definition 5. *(stack) The set of stacks* Stack *with typical element* S *to denote arbitrary stacks, and* s *to denote single elements of a stack, is defined as follows, where* $\phi \in \mathcal{L}$, $\pi \in \mathsf{Plan}'$, $\mathsf{PG} \subseteq \mathcal{R}_\mathsf{PG}$, *and* $\mathsf{PR} \subseteq \mathcal{R}_\mathsf{PR}$.

$$\begin{array}{lcl} \mathrm{s} & ::= & (\phi, \pi, \mathsf{PG}, \mathsf{PR}) \\ \mathrm{S} & ::= & \mathrm{s} \mid \mathrm{s}.\mathrm{S} \end{array}$$

E is used to denote the empty stack (or the empty stack element), and $E.\mathrm{S}$ *is identified with* S.

Note that the plan π of a stack element is from the extended set of plans Plan′, since a stack is a run-time construct which is not specified by the programmer when programming an agent (see definition 6). The plan π might thus, in contrast with the plans of plan generation and plan revision rules, contain a construct of the form $m(\phi)\downarrow$. The empty stack E is introduced for technical convenience when defining the semantics in section 3.2. Stacks as used here differ from intention stacks of AgentSpeak(L), as the elements comprising the stacks are essentially different: stack elements in this paper correspond with module calls, whereas in AgentSpeak(L) they represent (parts of) plans or intentions. Like non-modular 3APL, AgentSpeak(L) has one large set of plans (corresponding with the rules of 3APL).

An agent, as defined below, consists of a belief base, a goal base, a set of modules, and a function which specifies how beliefs are updated if actions are executed. As in non-modular 3APL, the belief base σ is a consistent set of propositional formulas. The goal base γ is essentially also a set of propositional formulas, and forms the top-level goals

of the agent. In contrast with non-modular 3APL however, each goal is associated with a module which should be called for achieving the goal, i.e., goals are of the form $m(\phi)$. The set of modules Mod form the modules which can be called to achieve (sub)goals. Defining how beliefs are updated through action execution by assuming a function $\mathcal{T}$, is standard for the way 3APL's semantics is usually defined. This function is used for technical convenience. We assume that $\mathcal{T}$ maintains consistency of the belief base.

DEFINITION 6. *(agent)* *Let* $\Sigma = \{\sigma \mid \sigma \subseteq \mathcal{L},\ \sigma \not\models \bot\}$ *be the set of belief bases. An agent is a tuple* $\langle \sigma, \gamma, \mathsf{Mod}, \mathcal{T}\rangle$ *where* $\sigma \in \Sigma$ *is the belief base,* $\gamma \subseteq \{m(\phi) \mid m \in \mathsf{ModName}, \phi \in \mathcal{L}\}$ *is the initial goal base, and* Mod *is a set of modules such that each module in this set has a distinct name.* $\mathcal{T}$ *is a partial function of type* $(\mathsf{BasicAction} \times \Sigma) \rightarrow \Sigma$ *and specifies the belief update resulting from the execution of basic actions.*

The notion of a configuration, as defined below, is used to represent the state of an agent at each point during computation. It consists of the elements which may change during execution of the agent, i.e., it consists of a belief base, a goal base, and a stack. Note that an agent initially has an empty stack.

DEFINITION 7. *(configuration)* *A configuration is a tuple* $\langle \sigma, \gamma, \mathrm{S}\rangle$ *where* $\sigma \in \Sigma$, $\gamma \subseteq \{m(\phi) \mid m \in \mathsf{ModName},\ \phi \in \mathcal{L}\}$, *and* $\mathrm{S} \in \mathsf{Stack}$. *If* $\langle \sigma, \gamma, \mathsf{Mod}, \mathcal{T}\rangle$ *is an agent, then* $\langle \sigma, \gamma, E\rangle$ *is the initial configuration of the agent.*

3.2 Semantics

The semantics of modular 3APL agents is defined by means of a transition system [12]. A transition system for a programming language consists of a set of axioms and transition rules for deriving transitions for this language. A transition is a transformation of one configuration into another and it corresponds to a single computation step. In the transition rules below, we assume an agent with a set of modules Mod, and a belief update function $\mathcal{T}$.

The first transition rule specifies how a transition for a composed stack can be derived, given a transition for a single stack element. It specifies that only the top element of a stack can be transformed or executed.[7]

DEFINITION 8. *(stack execution)* *Let* $s \neq E$.

$$\frac{\langle \sigma, \gamma, \mathrm{s}\rangle \rightarrow \langle \sigma', \gamma', \mathrm{S}'\rangle}{\langle \sigma, \gamma, \mathrm{s}.\mathrm{S}\rangle \rightarrow \langle \sigma', \gamma', \mathrm{S}'.\mathrm{S}\rangle}$$

In the transition rule for stack execution, we specify that s cannot be the empty stack. The reason is related to the transition rule for stack initialization of definition 9 below. In that rule, we want to specify that a stack initialization transition can only be derived if the current stack is empty. We however do not want to use that rule in combination with the rule for stack execution, since that would result in the possibility of deriving an "initialization" transition, even if the current stack is not empty.

An initialization transition can thus only be derived if the current stack is empty. The idea of initialization is that a (top-level) goal $m(\phi)$ from the goal base is (randomly) selected, and then the module m is called with the goal ϕ. The resulting stack is then of the form $(\phi, \epsilon, \mathsf{PG}, \mathsf{PR})$, where PG and PR are the plan generation and plan revision rules of m. Note that the plan of the resulting stack element is empty. The idea is that the plan generation rules of the module should now be used to generate an appropriate plan.

DEFINITION 9. *(initialization of stack)*

$$\frac{m(\phi) \in \gamma \qquad \langle m, \mathsf{PG}, \mathsf{PR}\rangle \in \mathsf{Mod}}{\langle \sigma, \gamma, E\rangle \rightarrow \langle \sigma, \gamma, (\phi, \epsilon, \mathsf{PG}, \mathsf{PR})\rangle}$$

Note that a consequence of this way of defining the goal base is that multiple goals from the goal base cannot be dispatched to the same module in one initialization. Thinking about alternative ways to define the goal base which might allow this, is left for future research.

The following transition rule is the rule for plan generation, defining when a plan generation rule can be applied. A prerequisite for applying a plan generation rule is that the plan of the relevant stack element is empty. A rule $\phi' \mid \beta \Rightarrow \pi$ from PG can then be applied if β is true, ϕ' follows from ϕ (i.e., the goal for which the module was called), and ϕ' is not believed to be achieved. Further, for a plan generation rule to be applicable, the goal ϕ' should not already be achieved, since there is no need to generate a plan for such a goal. Note that since ϕ' follows from ϕ, it is the case that ϕ is also not reached, if ϕ' is not reached. With respect to the second condition for plan generation rule application, we remark that a plan generation rule with goal antecedent p (i.e., $\phi' = p$) can thus be applied if the agent has, e.g., a goal $p \wedge q$ (i.e., $\phi = p \wedge q$). This interpretation of plan generation rules is standard for 3APL. It is convenient when programming, since plan generation rules can be specified for parts of a composed goal.

If a plan generation rule is applied, the plan π in its consequent becomes the plan of the resulting stack element. Further, the applied plan generation rule is removed from the set of plan generation rules of the stack element.

DEFINITION 10. *(plan generation)*

$$\frac{\phi' \mid \beta \Rightarrow \pi \in \mathsf{PG} \qquad \sigma \models_{\mathcal{L}_\mathrm{B}} \beta \qquad \phi \models \phi' \qquad \sigma \not\models \phi'}{\langle \sigma, \gamma, (\phi, \epsilon, \mathsf{PG}, \mathsf{PR})\rangle \rightarrow \langle \sigma, \gamma, (\phi, \pi, \mathsf{PG}', \mathsf{PR})\rangle}$$

where $\mathsf{PG}' = \mathsf{PG} \setminus \{\phi' \mid \beta \Rightarrow \pi\}$

The reason for removing plan generation rules is that we do not want the agent to try the same plan generation rule twice, to achieve a certain goal. Otherwise, the agent could get stuck "inside" a module trying to achieve a subgoal, if all its plans keep failing to reach the goal. The idea is that modules contain various plans for achieving a certain goal. If the agent cannot reach a certain subgoal of a plan with the designated module, the agent should thus at a certain point give up trying to reach the subgoal. It should just try another plan with possibly different subgoals. Wanting to remove plan generation rules of a stack element is the reason that we copy the rules into stack elements, rather than just referring to the rules of a module using the name of the module. If we would extend this semantics to a first order version, we would have to record which instances of a plan generation rule are tried, rather than just removing the rule. This mechanism is comparable to the mechanism

[7]For technical convenience, we overload the "." operator in definition 8. We use it to "push" a stack onto a stack, rather than to push a single stack element onto a stack, as it was, strictly speaking, defined in definition 5.

for selecting plans for subgoals in JACK. In JACK it is also the case that the same plan is not tried for the same subgoal twice.

The following two transition rules are standard for 3APL, and define how a plan is executed. The only difference is that the plan is now inside a stack element. A basic action a at the head of a plan can be executed if the function $\mathcal{T}$ is defined for a and the belief base σ in the configuration. The execution results in a change of belief base as specified through $\mathcal{T}$, and the action is removed from the plan. Further, goals from the goal base which are reached, are removed. Note that actions, which are executed from within a module, operate on the global belief base of the agent.

DEFINITION 11. *(action execution)*

$$\frac{\mathcal{T}(a,\sigma) = \sigma'}{\langle \sigma, \gamma, (\phi, a; \pi, \mathsf{PG}, \mathsf{PR})\rangle \rightarrow \langle \sigma', \gamma', (\phi, \pi, \mathsf{PG}, \mathsf{PR})\rangle}$$

where $\gamma' = \gamma \setminus \{m(\phi) \mid \sigma \models \phi\}$.

The transition below specifies the application of a plan revision rule of the form $\pi_h \mid \beta \Rightarrow \pi_b$ to a plan of the form $\pi_h; \pi$. The rule can be applied if β holds. If the rule is applied, the plan π_h is replaced by the body of the rule, yielding the plan $\pi_b; \pi$.

DEFINITION 12. *(plan revision)*

$$\frac{\pi_h \mid \beta \rightsquigarrow \pi_b \in \mathsf{PR} \qquad \sigma \models_{\mathcal{L}_\mathsf{B}} \beta}{\langle \sigma, \gamma, (\phi, \pi_h; \pi, \mathsf{PG}, \mathsf{PR})\rangle \rightarrow \langle \sigma, \gamma, (\phi, \pi_b; \pi, \mathsf{PG}, \mathsf{PR})\rangle}$$

We now revisit the point made in section 2.3, that the encapsulation provided by modules at run-time is especially advantageous in the case of 3APL. The reason is that modules encapsulate not only plan generation rules, but also plan revision rules. These plan revision rules provide very flexible ways for revising a plan during execution, but a large number of rules can interact in possibly unforeseen ways. This has to do with the semantics of plans not being compositional in case of plan revision, as explained in [18]. Being able to cluster plan revision rules into modules thus reduces the chances of unforeseen interactions with other rules: the number of plan revision rules in a module will be small compared with the global set of plan revision rules of a non-modular 3APL agent.

The next two transition rules specify the cases in which a stack element can be popped from the stack. Both transition rules specify that an element can be popped if its plan has finished execution, i.e., if the plan is empty. The idea is, that an agent should always finish the execution of an adopted plan, even though, e.g., the goal for which it was selected might already be reached. This is standard for 3APL, and the reason is that the programmer might have specified some necessary "clean-up" actions. Consider as an example the case where an agent still has to pay after refueling, even though the goal of having gas is already reached.

The first transition rule for popping a stack element specifies that the element can be popped if the goal ϕ of the stack element is reached.

DEFINITION 13. *(goal of stack element reached)*

$$\frac{\sigma \models \phi}{\langle \sigma, \gamma, (\phi, \epsilon, \mathsf{PG}, \mathsf{PR})\rangle \rightarrow \langle \sigma, \gamma, E\rangle}$$

The second transition rule for popping a stack element specifies that the element can be popped if there are no more applicable plan generation rules (regardless of whether the goal is reached).

DEFINITION 14. *(no applicable plan generation rules)*

$$\frac{\neg\exists(\phi' \mid \beta \Rightarrow \pi) \in \mathsf{PG} :\ (\sigma \models_{\mathcal{L}_\mathsf{B}} \beta \text{ and } \phi \models \phi' \text{ and } \sigma \not\models \phi')}{\langle \sigma, \gamma, (\phi, \epsilon, \mathsf{PG}, \mathsf{PR})\rangle \rightarrow \langle \sigma, \gamma, E\rangle}$$

The next three definitions specify the semantics of the construct $m(\phi')$ for calling a module. If this construct is encountered in a plan, and it is not annotated with the symbol $\downarrow$, the module m should be called for the goal ϕ'. That is, only if ϕ' is not yet reached. If a module with the name m exists,[8] a new element with goal ϕ', an empty plan, and the rules of m, is pushed onto the stack. Further, the construct $m(\phi')$ is annotated with $\downarrow$ to indicate that a module has been called for this subgoal. This is important, since we do not want to call module m again upon returning from m.

DEFINITION 15. *(calling a module)*

$$\frac{\langle m, \mathsf{PG}', \mathsf{PR}'\rangle \in \mathsf{Mod} \qquad \sigma \not\models \phi'}{\begin{array}{l}\langle \sigma, \gamma, (\phi, m(\phi'); \pi, \mathsf{PG}, \mathsf{PR})\rangle \rightarrow \\ \quad \langle \sigma, \gamma, (\phi', \epsilon, \mathsf{PG}', \mathsf{PR}').(\phi, m(\phi')\downarrow; \pi, \mathsf{PG}, \mathsf{PR})\rangle\end{array}}$$

The transition rules of the next definition specify that the agent can continue the execution of the rest of its plan, if the subgoal ϕ' occurring at the head of the plan is reached. The agent should continue if it has already called the module m for the subgoal, i.e., if the construct is of the form $m(\phi')\downarrow$, or if the module has not yet been called, i.e., if the construct is of the form $m(\phi')$. The latter case is however probably less likely to occur.

DEFINITION 16. *(subgoal reached)*

$$\frac{\sigma \models \phi'}{\langle \sigma, \gamma, (\phi, m(\phi')\downarrow; \pi, \mathsf{PG}, \mathsf{PR})\rangle \rightarrow \langle \sigma, \gamma, (\phi, \pi, \mathsf{PG}, \mathsf{PR})\rangle}$$

$$\frac{\sigma \models \phi'}{\langle \sigma, \gamma, (\phi, m(\phi'); \pi, \mathsf{PG}, \mathsf{PR})\rangle \rightarrow \langle \sigma, \gamma, (\phi, \pi, \mathsf{PG}, \mathsf{PR})\rangle}$$

The next transition rule specifies that if a module has been called for a subgoal, i.e., if a construct of the form $m(\phi')\downarrow$ is at the head of the plan of a stack element, and the subgoal ϕ' has not been reached, the plan fails. That is, the plan is replaced by an empty plan. If there are any applicable plan generation rules left, another plan can then be selected to try to achieve the goal ϕ of the stack element.

DEFINITION 17. *(subgoal dispatched and not reached)*

$$\frac{\sigma \not\models \phi'}{\langle \sigma, \gamma, (\phi, m(\phi')\downarrow; \pi, \mathsf{PG}, \mathsf{PR})\rangle \rightarrow \langle \sigma, \gamma, (\phi, \epsilon, \mathsf{PG}, \mathsf{PR})\rangle}$$

In this version of modular 3APL, a plan can fail to reach a goal only in case the programmer has specified the "wrong" actions in the plans. Since all actions (as is usually the case for 3APL's formal semantics) operate on the belief base and there is no notion of an external environment, there is no notion of the environment preventing an action from being

[8] And it should if the programmer has done a good job.

executed, thereby possibly causing a plan to fail. One could thus argue that it is not necessary to have a mechanism for selecting a different plan upon plan failure, since it is the job of the programmer to make sure that the plans do not fail and reach the goals. Any practical implementation of these ideas however would involve actions being executed in the environment, and it is thus important to incorporate a mechanism for handling plan failure..

The final transition rule below is very similar to the previous, in the sense that it specifies another reason for plan failure. In particular, it specifies that a plan fails if the action at its head cannot be executed, i.e., if the function $\mathcal{T}$ is undefined for the action and the belief base of the stack element.

DEFINITION 18. *(failure of action execution)*

$$\frac{\mathcal{T}(a,\sigma) \text{ is undefined}}{\langle \sigma, \gamma, (\phi, a; \pi, \mathsf{PG}, \mathsf{PR})\rangle \rightarrow \langle \sigma, \gamma, (\phi, \epsilon, \mathsf{PG}, \mathsf{PR})\rangle}$$

3.3 Example

For illustration, we present a simple example of a modular 3APL agent. The agent has to bring a rock from the location where the rocks are $loc(rock)$[9], to its base location $loc(base)$. It has three modules, i.e., *collectRock* for the general goal of collecting the rock, and *goTo* and *pickUp* for going to a location and picking up a rock, respectively. Initially, the agent believes that it is at the base location, and that it does not have a rock, i.e., $\neg have(rock)$. Further, it has the goal $collectRock(have(rock) \wedge loc(base))$, i.e., it wants to achieve the goal $have(rock) \wedge loc(base)$ using module *collectRock*.

Below, we define the initial belief base σ and goal base γ of the rock collecting agent.

$$\begin{array}{lcl} \sigma & = & \{loc(base), \neg have(rock)\} \\ \gamma & = & \{collectRock(have(rock) \wedge loc(base))\} \end{array}$$

The plan generation rules PG_{cr} of the module *collectRock* are defined as below. The set of plan revision rules is empty.

$$\begin{array}{lcl} \mathsf{PG}_{cr} & = & \{have(rock) \wedge loc(base) \mid \top \Rightarrow \\ & & goTo(loc(rock)); pickUp(have(rock)); \\ & & goTo(loc(base))\} \end{array}$$

There is one plan generation rule which can be used for the goal $have(rock) \wedge loc(base)$, i.e., the initial goal of the agent. The plan of the rule consists entirely of calls to other modules. Note that the *goTo* module is called with two different goals. The rule does not specify a condition on beliefs (other than $\top$). In particular, there is no need to specify that the agent should, e.g., believe that it is at the base location, rather than at the rock location. If the agent would already be at the rock location, the first module call of the plan, i.e., $goTo(loc(rock))$, would be skipped, since the subgoal $loc(rock)$ is already reached (see definition 16).

The plan generation rules PG_{gt} and plan revision rules PR_{gt} of the module *goTo* are defined as follows.

$$\begin{array}{lcll} \mathsf{PG}_{gt} & = & \{loc(rock) & \mid \mathbf{B}(loc(base)) \Rightarrow toRock \\ & & loc(base) & \mid \mathbf{B}(loc(rock)) \Rightarrow toBase\} \\ \mathsf{PR}_{gt} & = & \{toRock & \mid \mathbf{B}(loc(rock)) \rightsquigarrow skip \\ & & toRock & \mid \neg\mathbf{B}(loc(rock)) \rightsquigarrow east; toRock \\ & & toBase & \mid \mathbf{B}(loc(base)) \rightsquigarrow skip \\ & & toBase & \mid \neg\mathbf{B}(loc(base)) \rightsquigarrow west; toBase\} \end{array}$$

[9]Note that all formulas are propositional, although we use brackets for presentation purposes.

This module has two plan generation rules: one for selecting a plan to get from the base location to the rock location, and one for the other way around. The plans *toRock* and *toBase* in the bodies of the plan generation rules are non-executable basic actions, which are used as procedure variables.[10] Assuming that the rock is located east from the base location, we specify plan revision rules for moving east until the rock location is reached, and moving west until the base location is reached. The action *skip* is a special action which does nothing if executed.

In this simple example, we specify separate plan revision rules for moving east and west respectively. The plans for moving to the rock location and to the base location thus use different plan revision rules. We could therefore have created two separate modules, i.e., one for going to the rock, and one for going to the base. In a more realistic setting however, one could imagine to have *one* set of plan revision rules for moving to any given location. In that case, it would be advantageous to specify these rules only in one module.

The plan generation rules PG_{pu} of the module *pickUp* are defined as below, and the set of plan revision rules is empty.

$$\begin{array}{lcl} \mathsf{PG}_{pu} & = & \{have(rock) \mid \mathbf{B}(loc(rock)) \Rightarrow pickUp1 \\ & & have(rock) \mid \mathbf{B}(loc(rock)) \Rightarrow pickUp2\} \end{array}$$

The plan generation rules of this module can be applied if the agent believes it is at the rock location. Note that the call $pickUp(have(rock))$ to this module from the *collectRock* module, can only be executed if the subgoal of the previous module call, i.e., $goTo(loc(rock))$, has been achieved. If $pickUp(have(rock))$ is executed, we thus know for sure that the agent believes to be at the rock location. The module *pickUp* illustrates that the agent may have multiple plans, i.e., $pickUp1$ and $pickUp2$ in this case, for achieving the same goal. If, e.g., $pickUp1$ fails, the agent can try $pickUp2$.

4. FUTURE RESEARCH

As directions of future research, we mention the following. In the current proposal there is no notion of importing a module into another module. Some notion of importation could be added in a straightforward way, but it will have to be investigated what exactly the semantics of importation should be. Should the rules of the imported module just be "added" to the other module, or should there be some kind of prioritization among rules of the module itself and the imported module?

Further, goal-oriented modularity as presented here provides a high degree of information hiding. That is, when calling a module, only the name of the module has to be known. In principle, any part of a module can be adapted without having to adapt the call to the module. We thus have no notion of an interface of a module, i.e., those parts of the module which are known outside the module. Nevertheless, it might be worthwile to investigate whether the framework can be extended with some notion of interface, such as the goals for which a module can be called, etc. This might be a useful tool to help a programmer.

Also, we envisage that a mechanism similar to the mechanism of dispatching a goal to a module, could be used in a multi-agent team work setting (see, e.g., [22]) to delegate a goal to an agent. Using a uniform mechanism both for

[10]Non-executable basic actions are often termed *abstract plans* in the 3APL literature.

calling modules and delegating goals to agents could potentially yield more transparent systems. In case of delegation, a plan would have to contain a request message to an agent, rather than a call to a module. A way of "returning" from the request, just as one can return from a module, would have to be defined. An agent would for example have to report back to the requesting agent, either with a message expressing that the goal has been achieved, or that he has failed and stopped trying. Moreover, it can be interesting to investigate a construct for calling modules in parallel. This can also be interesting in the case of multi-agent teamwork as also discussed in [22], since an agent could then dispatch several goals to different agents in parallel.

Finally, we remark that it could be useful to be able to store information during execution within a module, which would not need to be kept after returning from the module. This could be realized by adding actions to the specification of a module, which would then update the module's internal information store, rather than the global belief base. This could be considered as a compromise between the way beliefs are handled in capabilities, and the way we handle beliefs in the presented framework.

Concluding, we have presented the idea of goal-oriented modularity, which takes the goals of an agent as the basis for modularization. Since we view goals as being an essential ingredient of cognitive agents, we argue that this approach to modularity is suited for cognitive agent programming languages. Further, we have shown how goal-oriented modularity might be incorporated in a cognitive agent programming language, by presenting a formalization of goal-oriented modularity in 3APL.

5. REFERENCES

[1] F. Bellifemine, A. Poggi, G. Rimassa, and P. Turci. An object oriented framework to realize agent systems. In *Proc. of WOA 2000 Workshop*, pages 52–57. 2000.

[2] J. A. Bergstra, J. Heering, and P. Klint. Module algebra. *Journal of the Association for Computing Machinery*, 37(2):335–372, 1990.

[3] R. H. Bordini and A. F. Moreira. Proving the asymmetry thesis principles for a BDI agent-oriented programming language. *Electronic Notes in Theoretical Computer Science*, 70(5), 2002.

[4] L. Braubach, A. Pokahr, and W. Lamersdorf. Extending the capability concept for flexible BDI agent modularization. In *Proc. of ProMAS'05*, 2005.

[5] P. Busetta, N. Howden, R. Rönnquist, and A. Hodgson. Structuring BDI agents in functional clusters. In *ATAL '99: 6th Int. Workshop on Intelligent Agents VI, Agent Theories, Architectures, and Languages*, pages 277–289, 2000. Springer-Verlag.

[6] M. Dastani, M. B. van Riemsdijk, F. Dignum, and J.-J. Ch. Meyer. A programming language for cognitive agents: goal directed 3APL. In *Programming multiagent systems, first int. workshop (ProMAS'03)*, *LNAI*, pages 111–130. Springer, Berlin, 2004.

[7] M. Dastani, M. B. van Riemsdijk, J. Hulstijn, F. Dignum, and J.-J. Ch. Meyer. Enacting and deacting roles in agent programming. *Agent-Oriented Software Engineering V*, *LNCS*, pages 189–204. Springer-Verlag, 2005.

[8] M. Dastani, M. B. van Riemsdijk, and J.-J. Ch. Meyer. Programming multi-agent systems in 3APL. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.

[9] C. Ghezzi, M. Jazayeri, and D. Mandrioli. *Fundamentals of software engineering*. Prentice-Hall International, London, 1991.

[10] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming in 3APL. *Int. J. of Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

[11] B. Meyer. *Object-oriented software construction*. Series in Computer Science. Prentice-Hall International, London, 1988.

[12] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.

[13] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: a BDI reasoning engine. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.

[14] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In W. van der Velde and J. Perram, editors, *Agents Breaking Away (LNAI 1038)*, pages 42–55. Springer-Verlag, 1996.

[15] M. B. van Riemsdijk, M. Dastani, F. Dignum, and J.-J. Ch. Meyer. Dynamics of declarative goals in agent programming. *Proc. of the second int. workshop on Declarative agent languages and technologies (DALT'04)*, *LNCS*, pages 1–18. Springer-Verlag, 2005.

[16] M. B. van Riemsdijk, M. Dastani, and J.-J. Ch. Meyer. Semantics of declarative goals in agent programming. In *Proc. of AAMAS'05*, 2005.

[17] M. B. van Riemsdijk, M. Dastani, and J.-J. Ch. Meyer. Subgoal semantics in agent programming. *Progress in Artifical Intelligence: 12th Portuguese Conference on Artificial Intelligence (EPIA'05)*, *LNCS*, pages 548–559. Springer-Verlag, 2005.

[18] M. B. van Riemsdijk, J.-J. Ch. Meyer, and F. S. de Boer. Semantics of plan revision in intelligent agents. In C. Rattray, S. Maharaj, and C. Shankland, editors, *Theoretical Computer Science*, pages 240–257. 2006. Special issue of Algebraic Methodology and Software Technology (AMAST'04).

[19] M. B. van Riemsdijk, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming in Dribble: from beliefs to goals using plans. In *Proc. of AAMAS'03*, 2003.

[20] M. Winikoff. JACK™ intelligent agents: an industrial strength platform. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.

[21] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative and procedural goals in intelligent agent systems. In *Proc. of the eighth int. conf. on principles of knowledge respresentation and reasoning (KR2002)*, 2002.

[22] K. Yoshimura, R. Rönnquist, and L. Sonenberg. An approach to specifying coordinated agent behaviour. In *PRIMA'00*, *LNAI*, pages 115–127. Springer, 2000.

Prototyping 3APL in the Maude Term Rewriting Language

Extended Abstract

M. Birna van Riemsdijk[1] Frank S. de Boer[1,2] Mehdi Dastani[1] John-Jules Ch. Meyer[1]
Utrecht University[1]
CWI, Amsterdam[2]
The Netherlands
[birna | frankb | mehdi | jj]@cs.uu.nl

ABSTRACT

This paper presents an implementation of (a simplified version of) the cognitive agent programming language 3APL in the Maude term rewriting language. We argue that Maude is very well suited for prototyping agent programming languages such as 3APL.

Categories and Subject Descriptors

D.3.1 [**Programming Languages**]: Formal Definitions and Theory; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, languages and structures*; I.2.5 [**Artificial Intelligence**]: Programming Languages and Software

General Terms

Theory, Languages

Keywords

Agent programming languages, prototyping, term rewriting

1. INTRODUCTION

An important line of research in the agent systems field is research on agent programming languages [2]. This type of research is concerned with an investigation of what kind of programming constructs an agent programming language should contain, and what exactly the meaning of these constructs should be. In order to test whether these constructs indeed facilitate the programming of agents in an effective way, the programming language has to be implemented.

This can be done using Java, which was for example used for implementing the agent programming language 3APL [13]. Java has several advantages, such as its platform independence, its support for building graphical user interfaces, and the extensive standard Java libraries. A disadvantage is however that the translation of the formal semantics of an agent programming language such as 3APL into Java is not very direct. It can therefore be difficult to ascertain that such an implementation is a faithful implementation of the semantics of the agent programming language, and experimenting with different language constructs and semantics can be quite cumbersome.

As an alternative to the use of Java, we discuss in this abstract[1] the usage of the Maude term rewriting language [6] for prototyping 3APL. Maude is based on the mathematical theory of rewriting logic. The language has been shown to be suitable both as a *logical* framework in which many other logics can be represented, and as a *semantic* framework, through which programming languages with an operational semantics can be implemented in a rigorous way [17]. We argue that, since agent programming languages such as 3APL have both a logical and a semantic component, Maude is very well suited for prototyping such languages. Further, it turns out that, since Maude is reflective, 3APL's meta-level reasoning cycle or deliberation cycle can be implemented very naturally in Maude.

An important advantage of Maude is that it can be used for verification as it comes with an LTL model checker [12]. This abstract does not further address model checking 3APL using Maude, since the usage of Maude's model checker is relatively easy, given the implementation of 3APL in Maude. The fact that Maude provides these verification facilities however, is an important, and in fact, our original, motivation for our effort of implementing 3APL in Maude.

2. IMPLEMENTATION OF 3APL IN MAUDE

3APL, which was first introduced by Hindriks [13], is a cognitive agent programming language. This means that it has explicit constructs for representing the high-level mental attitudes of an agent. The version of 3APL that we have implemented in Maude comes closest to the one presented in [25]. It is single-agent and builds on propositional logic. In this version, a 3APL agent has beliefs, a plan, and goals. Beliefs represent the current state of the world and information internal to the agent. Goals represent the desired state of the world, and plans are the means to achieve the goals. Further, a 3APL agent has rules for selecting a plan to achieve a certain goal given a certain belief, and it has rules for revising its plan during execution. Together, these elements may form the state or configuration of a 3APL

[1]The full version of this paper has been submitted elsewhere.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

agent. In order to be able to refer to an agent's beliefs and goals, 3APL incorporates logical languages (belief and goal query languages) with corresponding semantics. The formal operational semantics of 3APL is defined by means of a transition system [21].

"Maude is a formal declarative programming language based on the mathematical theory of rewriting logic [18]. Maude and rewriting logic were both developed by José Meseguer. Maude is a state-of-the-art formal method in the fields of algebraic specification [27] and modeling of concurrent systems. The Maude language specifies rewriting logic theories. Data types are defined algebraically by equations and the dynamic behavior of a system is defined by rewrite rules which describe how a part of the state can change in one step." [20]

The general idea of the implementation of 3APL in Maude, is that 3APL configurations are represented as terms in Maude, and the transition rules of 3APL are mapped onto rewrite rules of Maude. This idea is taken from [26], in which, among others, implementations in Maude of the operational semantics of a simple functional language and an imperative language are discussed. Further, the logical languages of 3APL and their corresponding semantics can be represented in Maude as equational theories. Finally, the meta-level reasoning cycle or deliberation cycle of 3APL [13, 8, 24] can also be modelled very naturally in Maude, since rewriting logic is *reflective* [7]. "Informally, a reflective logic is a logic in which important aspects of its metatheory can be represented at the object level in a consistent way, so that the object level representation correctly simulates the relevant metatheoretic aspects. " [6, Chapter 10]

3. ADVANTAGES OF MAUDE

Based on our experience with the implementation of 3APL in Maude, we argue that Maude is well suited as a prototyping and analysis tool for logic-based cognitive agent programming languages. We observe that cognitive agent programming languages such as 3APL have a logical *as well as* a semantic component: the logical part consists of the belief and goal query languages, and the semantic part consists of the transition system. Since Maude supports both the logical and the semantic component, the implementation of languages like 3APL in Maude is very natural, and the integration of the two components is seamless.

We observe that the direct mapping of transition rules of 3APL into rewrite rules of Maude ensures a *faithful* implementation of the operational semantics of 3APL in Maude. This direct mapping is a big advantage compared with the implementation of a 3APL interpreter in a general purpose language such as Java, in which the implementation is less direct. In particular, in Java one needs to program a mechanism for applying the specified transition rules in appropriate ways, whereas in the case of Maude the term rewriting engine takes care of this. As another approach of implementing a cognitive agent programming language in Java, one might consider to implement the plans of the agent as methods in Java, which is for example done in the Jadex framework [22]. Since Java does not have support for revision of programs, implementing 3APL plans as methods in Java is not possible. We refer to [24] for a theoretical treatment of the issues with respect to semantics of plan revision.

A faithful implementation of 3APL's semantics in Maude is very important with regard to our main original motivation for this work, i.e., to use the Maude LTL model checker to do formal verification for 3APL. The natural and transparent way in which the operational semantics of 3APL can be mapped to Maude, is a big advantage compared with the use of, e.g., the PROMELA language [14] in combination with the SPIN model checker [15].

SPIN is a generic verification system which supports the design and verification of asynchronous process systems. SPIN verification models are focused on proving the correctness of process interactions, and they attempt to abstract as much as possible from internal sequential computations. The language PROMELA is a high level language for specifying abstractions of distributed systems which can be used by SPIN, and it's main data structure is the message channel. In [3], an implementation of the cognitive agent programming language AgentSpeak(F) - the finite state version of AgentSpeak(L) [4] - in PROMELA is described, for usage with SPIN. Most of the effort is devoted to translating AgentSpeak(F) into the PROMELA data structures. It is shown how to translate the data structures of AgentSpeak(F) into PROMELA channels. It is however not shown that this translation is correct, i.e., that the obtained PROMELA program correctly simulates the AgentSpeak(F) semantics. In contrast with the correctness of the implementation of 3APL in Maude, the correctness of the AgentSpeak(F) implementation in PROMELA is not obvious, because of the big gap between AgentSpeak(F) data structures and semantics, and PROMELA data structures and semantics. In [12], it is shown that the performance of the Maude model checker is comparable with that of SPIN.

A further important advantage of Maude is that deliberation cycles can be programmed very naturally as strategies, using reflection. A related advantage is that a clear separation of the object-level and meta-level semantics can be maintained in Maude. A 3APL program can be executed without making use of a deliberation cycle, while it can equally easily be executed *with* a deliberation cycle.

While we have implemented a simplified version of 3APL in Maude, we argue that the features of Maude are very well suited to support the implementation of various extensions of this version of 3APL. Implementing these extensions is left for future research.

In particular, an extension of a single-agent to a *multi-agent* version will be naturally implementable, since, from a computational point of view, rewriting logic is intrinsically concurrent [17]. It was in fact the search for a general concurrency model that would help unify the heterogenity of existing models, that provided the original impetus for the first investigations on rewriting logic [18].

Further, a more practically useful implementation will have to be *first-order*, rather than propositional. Although the implementation of a first-order version will be more involved, it can essentially be implemented in the same way as the current version, i.e., by mapping transition rules to rewrite rules. In [9], the transition rules for a first-order version of 3APL are presented. Configurations in this setting have an extra substitution component, which records the assignment of values to variables. An implementation of this version in Maude will involve extending the notion of a configuration with such a substitution component, as specified in the cited paper.

Finally, we aim to extend the *logical part* in various ways.

Regarding this propositional version, one could think of extending the belief base and goal base to arbitrary sets of propositional formulas, rather than just sets of atoms as we have used in the current implementation. Also, the belief and goal query languages could be extended to query arbitrary propositional formulas. The satisfaction relations for queries could then be implemented using, e.g., tableau methods as suggested in [17]. Further, when considering a first-order version of 3APL, the belief base can be implemented as a set of Horn clauses, or even as a Prolog program. Implementing the belief base as a Prolog program is possible in the current Java implementation of 3APL. How to define standard Prolog in rewriting logic has been described in [16]. Finally, we aim to experiment with the implementation of more sophisticated specifications of the goals and their accompanying satisfaction relations, as proposed in [23].

4. REFERENCES

[1] J. V. Baalen, J. L. Caldwell, and S. Mishra. Specifying and checking fault-tolerant agent-based protocols using Maude. In *FAABS '00: Proc. of the First Int. Workshop on Formal Approaches to Agent-Based Systems-Revised Papers*, volume 1871 of *LNCS*, London, UK, 2001. Springer-Verlag.

[2] R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni. *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.

[3] R. H. Bordini, M. Fisher, C. Pardavila, and M. Wooldridge. Model checking AgentSpeak. In *Proc. of AAMAS'03*, 2003.

[4] R. H. Bordini and A. F. Moreira. Proving the asymmetry thesis principles for a BDI agent-oriented programming language. *Electronic Notes in Theoretical Computer Science*, 70(5), 2002.

[5] N. Boudiaf, F. Mokhati, M. Badri, and L. Badri. Specifying DIMA multi-agent models using Maude. In *Proc. of PRIMA'04*, volume 3371 of *LNCS*. Springer, 2005.

[6] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. Maude manual (version 2.1.1). 2005.

[7] M. Clavel and J. Meseguer. Reflection and strategies in rewriting logic. *Electronic Notes in Theoretical Computer Science*, 4:125–147, 1996.

[8] M. Dastani, F. S. de Boer, F. Dignum, and J.-J. Ch. Meyer. Programming agent deliberation – an approach illustrated using the 3APL language. In *Proc. of AAMAS'03*, 2003.

[9] M. Dastani, M. B. van Riemsdijk, F. Dignum, and J.-J. Ch. Meyer. A programming language for cognitive agents: goal directed 3APL. In *Programming multiagent systems, first int. workshop (ProMAS'03)*, volume 3067 of *LNAI*. Springer, Berlin, 2004.

[10] M. Dastani, M. B. van Riemsdijk, and J.-J. Ch. Meyer. Programming multi-agent systems in 3APL. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.

[11] F. Durán, S. Eker, P. Lincoln, and J. Meseguer. Principles of mobile Maude. In *Proc. of the Second Int. Symposium on Agent Systems and Applications and Fourth Int. Symposium on Mobile Agents*, volume 1882 of *LNCS*, 2000. Springer-Verlag.

[12] S. Eker, J. Meseguer, and A. Sridharanarayanan. The Maude LTL model checker. In F. Gaducci and U. Montanari, editors, *Proc. of the 4th Int. Workshop on Rewriting Logic and Its Applications (WRLA 2002)*, volume 71 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2002.

[13] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming in 3APL. *Int. J. of AAMAS*, 2(4):357–401, 1999.

[14] G. Holzmann. *Design and Validation of Computer Protocols*. Prentice Hall, New Jersey, 1991.

[15] G. Holzmann. The model checker SPIN. *IEEE Trans. Software Engineering*, 23(5):279–295, 1997.

[16] M. Kulas and C. Beierle. Defining standard Prolog in rewriting logic. In *Electronic Notes in Theoretical Computer Science*, volume 36. Elsevier Science Publishers, 2000.

[17] N. Martí-Oliet and J. Meseguer. Rewriting logic as a logical and semantic framework. In J. Meseguer, editor, *Electronic Notes in Theoretical Computer Science*, volume 4. Elsevier Science Publishers, 2000.

[18] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96:73–155, 1992.

[19] J. Meseguer. Membership algebra as a logical framework for equational specification. In *Proc. of the 12th Int. Workshop on Recent Trends in Algebraic Development Techniques*, 1997. Springer-Verlag.

[20] P. C. Ölveczky. Formal modeling and analysis of distributed systems in Maude. Lecture Notes, 2005.

[21] G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.

[22] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: a BDI reasoning engine. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.

[23] M. B. van Riemsdijk, M. Dastani, and J.-J. Ch. Meyer. Semantics of declarative goals in agent programming. In *Proc. of AAMAS'05*, 2005.

[24] M. B. van Riemsdijk, J.-J. Ch. Meyer, and F. S. de Boer. Semantics of plan revision in intelligent agents. In C. Rattray, S. Maharaj, and C. Shankland, editors, *Theoretical Computer Science*, pages 240–257. 2006. Special issue of Algebraic Methodology and Software Technology (AMAST'04).

[25] M. B. van Riemsdijk, W. van der Hoek, and J.-J. Ch. Meyer. Agent programming in Dribble: from beliefs to goals using plans. In *Proc. of AAMAS'03*, 2003.

[26] A. Verdejo and N. Martí-Oliet. Executable structural operational semantics in Maude. Technical report, Universidad Complutense de Madrid, Madrid, 2003.

[27] M. Wirsing. Algebraic specification. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics. Elsevier, Amsterdam, 1990.

A Jabber-based Multi-Agent System Platform *

Miguel Escrivá Gregori
Dept. de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n
Valencia, Spain
mescriva@dsic.upv.es

Javier Palanca Cámara
Dept. de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n
Valencia, Spain
jpalanca@dsic.upv.es

Gustavo Aranda Bada
Dept. de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n
Valencia, Spain
garanda@dsic.upv.es

ABSTRACT

This paper presents a new multi-agent system platform that takes advantage of XMPP. Among the new message transport protocols that have been recently developed, Jabber (XMPP) is a protocol that is based on Instant Messaging networks that use real-time messaging between applications. This platform is fully FIPA compliant and provides a simple interface to create agents using this new communication concept.

1. INTRODUCTION

In this paper we introduce a new multiagent-system platform (called SPADE) that takes advantages of a new communication model and that provides a simple interface to create agents using this new communication concept. This communication model is based on Instant Messaging protocols that use real-time messaging between applications. Agent communication is improved by using this protocol due to its new features, such as the Contact List management for agents, the ability to receive a notification whenever any other known agent (included in its Contact List) comes online or the possibility of engaging a simultaneous multi-conference with many agents.

This new platform is fully FIPA compliant to allow interoperability with other platforms. Its main objective is to provide good performance while retaining strong scalability options and to provide an open design that allows the implementation of platform components in several high-level programming languages.

Our proposal provides a human-like framework to communicate agents. It allows them to have a conversation (with all the meta-information required in a real conversation included) with any other 'entity'. This platform is described in depth in the following section.

*This work is partially supported by the TIC2003-07369-C02-01 and TIN2005-03395 projects of the spanish government.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. SPADE

SPADE (a Smart Python multi-Agent Development Environment) is a new platform aimed at solving the drawbacks of the communication models from other platforms. The multi-agent system platform SPADE is built around a new communication framework (Jabber[1]) which provides new and powerful capabilities to the communication layer. The platform core is designed over this layer, which communicates the platform elements with one another.

2.1 The Communication Model

Communications in SPADE are handled by means of the Jabber protocol. Jabber is the Internet standard protocol for instant messaging and presence communication[1, 2]. Its base technology is the Extensible Messaging and Presence Protocol (XMPP) standard, a set of streaming XML protocols and technologies that enable any two entities on the Internet to exchange messages, presence, and other structured information in close to real-time.

Presence notification is one of the most useful Jabber features that SPADE provides. It is a system by which an entity involved in a Jabber communication can know at any time what the current status of any of the other entities involved is. Jabber entities can declare what their current status is and all other entities who share a bond with them will automatically be notified. An entity status usually indicates its availability and readiness to engage in a communication (e.g. *'Available', 'Busy', 'Don't disturb', ...*), but it is not closed to a short predefined list. Entities can define their own status and their meanings. Two or more Jabber entities can be bonded (or *subscribed*) to each other. Entities that share a bond may make use of presence notification as described above.

Another interesting SPADE feature is **Multi-user Conference** (MUC). MUC features can be used to create forums for agents to connect to when performing a multi-agent activity. For instance, in a virtual auction scenario performed by agents, an agent playing the role of the auctioneer can create a password-protected channel and give the password only to previously certified agents. Then, in the channel, the auctioner can give the right to speak only to agents who are going to play the bidder role, leaving the rest of the agents in spectator mode. Once the auction starts, every time a bidder wants to place a bid, the agent only needs to *'speak'* in the channel, and all the other bidders and the auctioneer

will know the bid at the same time. MUC channels can also be used for broadcasting announcements, public platform information, selected distribution lists, etc.

2.2 The Platform Model

The SPADE platform is modeled according to the FIPA standard proposal for a multi-agent platform [3]. It features standard basic services (such as an Agent Management System and a Directory Facilitator) that are designed as Jabber server components (or add-ins). Although the core communication system relies on the Jabber technology, other message transport protocols, such as the *HTTP* protocol[4], are also supported.

A brief schema of the SPADE Platform can be viewed in Figure 1.

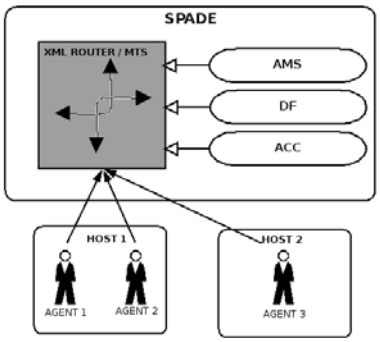


Figure 1: Platform Model

The main platform element is the **XML Router**, which the rest of the platform components and agents are connected to. It is a standard XMPP server[1] that routes all the messages from its sender to the specified receiver with no user intervention. This XML Router acts as the Message Transport System (MTS) and is the only 'third-party' component of the platform. One of this components connected to the router is the **SPADE Agent Communication Channel** (ACC), which manages all the communication within the platform. It receives the FIPA-ACL[1][5] messages that arrive to the platform and it redirects them to the correct destination element. This destination can be either an agent or another component.

2.2.1 Security Concerns

The SPADE framework provides some built-in security mechanisms that help to maintain the system's integrity. These mechanisms are developed in different security levels:

- Logging to the XML Router requires a **username and a password**. This mechanism prevents an unauthorized connection to the platform if it is not approved by the platform administrator.
- A connection to the XML Router can be encoded with a symmetric cryptographic algorithm using SSL (Secure Socket Layer). SSL provides data **cyphering**, server **authentication**, message **integrity** and, optionally, client authentication for TCP-IP connections.

[1]FIPA Agent Communication Language

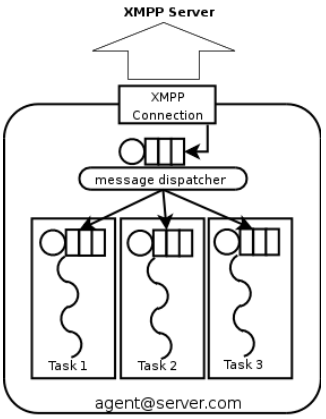


Figure 2: SPADE Agent Model

- To ensure the authentication of a message, the XML Router provides another mechanism to avoid identity theft. The platform prevents identity spoofing by overriding a '*from*' field that does not match the sender's.

2.2.2 Platform Inter-operability

The Message Transport System can manage as many MTPs as there are plugged to the platform. If the protocol is XMPP (Jabber protocol), the MTS delivers the message directly to the receiver agent. This feature allows SPADE to communicate with any kind of FIPA compliant platform that supports the XMPP Message Transport Protocol.

We have also developed a new JADE plug-in that provides a MTP implementing the XMPP protocol. This plug-in has been accepted by the JADE Team and is published in [6]. Thanks to this new MTP, a SPADE platform and a JADE platform can communicate with each other using the Jabber protocol. Actually, two JADE platforms can use this new XMPP plug-in to communicate their agents and take advantage of this new proposed Transport Protocol.

2.3 The Agent Model

Every component inside the platform is built as an agent. SPADE agents are elements that are connected to the MTS that can send messages to each other and to other platforms.

The Agent Model is basically composed of a connection mechanism to the platform, a message dispatcher, and a set of different tasks that the dispatcher gives the messages to (–see Figure 2). Every agent needs an identifier called Jabber ID (JID) and a valid password to establish a connection with the platform. If the platform registration process is disabled, then the platform administrator should define the registration policies. An agent can run serveral tasks simultaneously. A **task** is a process that an agent can execute using repeating patterns. SPADE provides some predefined task types: Cyclic, One-Shot, Periodic, Time-Out, and Finite State Machine Task. Those task types help to implement different the tasks that an agent can perform. Every agent can have as many tasks as desired. When a message arrives to the agent, the message dispatcher redirects it to the correct task queue (–see Figure 2).

2.4 The Main Features

The main features of SPADE are the following:

1. The SPADE platform has already covered the FIPA standard. AMS and DF ontology is supported.
2. SPADE supports two Message Transport Protocols: HTTP and XMPP. This feature allows different platforms to communicate using this new communication model or even older platforms that make use of the wide-spread HTTP protocol.
3. SPADE supports two different content languages: **FIPA-SL** and **RDF**, which are the two main options for content language used in today's agent communication.
4. Another feature defined by FIPA that SPADE supports is agent **mobility**. A SPADE agent can move from one platform to another authorized platform and continue it's execution in the new target.
5. As we stated in Section 2.3, SPADE agents do reach their goals by running one or more tasks.
6. SPADE has a special agent to provide a user-friendly view of the platform. This GUI provides a mechanism to load and unload agents in the platform. It allows the user to search for agents and services and send messages to any agent.
7. SPADE was developed in Python. This allows the execution of the platform in several architectures and operating systems such as Windows, Linux, MacOS, Windows Mobile, PalmOS, SymbianOS for mobile phones, etc. Python also allows using some performance characteristics lile *stackless* threads or the *psyco* module to massively speed up the execution.
8. One of the new and powerful feature of SPADE is the possibility to use **Presence Notification** between components. This allows the system to determine the current state of the components that are connected to the platform in real-time.
9. Managing a conversation with more than one agent is possible thanks to a Jabber mechanism. This mechanism (called **multi-user conference**) is based on chat rooms where an agent can join and share messages with a group of agents.
10. The Protocol used to send messages is a distributed client-server protocol. The servers that route the Jabber messages are designed to support a very large number of users and messages. This feature helps to improve the SPADE performance and scalability.
11. The platform provides a user and password authentication to improve security as well as a cyphered connection to the platform to perform confidentiality.

3. CONCLUSIONS AND FUTURE WORK

One of the main drawback on multi-agent platforms is the use of an inappropiate message transport protocol to communicate agents. To solve this, we have presented a new communication model using the Jabber Instant Messaging Protocol and we have developed a new agent platform using this technology. The new platform, called SPADE, is a fully FIPA compliant platform that supports all the required features. It has been developed in a modern language that increases the productivity of the programmer. The main characteristic of SPADE is the support of the new communication protocol between agents. This protocol, which is based on Instant Messaging systems, uses a distributed network to route messages from one agent to another. It also provides new mechanisms that improve the communicative acts, such as Presence Notification or Multi-User Conference. These characteristics, which are not present in any other agent platform, provide new communication capabilities between agents, which make agents more versatile. SPADE also has some security properties which provide confidentiality and authentication systems to the platform.

SPADE is released under the General Public License (GPL) and can be downloaded and tested at [7]. The plug-in for the JADE multi-agent platform that enables communication through XMPP is also available for download at [7] and [6].

4. ADDITIONAL AUTHORS

Additional authors: Ana García-Fornes, Vicente Julian and Vicent Botti (Dept. de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia).
{agarcia, vinglada, vbotti}@dsic.upv.es

5. REFERENCES

[1] Jabber Software Foundation. Extensible Messaging and Presence Protocol (XMPP): Core. Technical report, http://www.ietf.org/rfc/rfc3920.txt, October 2004.
[2] Jabber Software Foundation. Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. Technical report, http://www.ietf.org/rfc/rfc3921.txt, October 2004.
[3] FIPA. Abstract architecture specification. Technical Report SC00001L, 2002.
[4] FIPA. HTTP Message Trasport Protocol. Technical Report SC00084, 2002.
[5] FIPA. Agent ACL message structure specification. Technical Report XC00061E, 2001.
[6] JADE XMPP-Plugin. http://jade.tilab.com/community-3rdpartysw.htm.
[7] SPADE: Smart Python multi-Agent Development Environment. http://gti-ia.dsic.upv.es/projects/magentix/.

Goal Types in Agent Programming

[Extended Abstract] *

Mehdi Dastani
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands
mehdi@cs.uu.nl

M. Birna van Riemsdijk
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands
birna@cs.uu.nl

John-Jules Ch. Meyer
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands
jj@cs.uu.nl

ABSTRACT

This paper discusses three types of declarative goals and motivates their integration in logic-based agent-oriented programming languages. These goal types are perform goals, achieve goals, and maintain goals. A goal type is considered as a specific agent attitude towards goals. The semantics for each goal type is explained from an operational perspective. It is argued that the suggested semantics of the goal types ensure some desirable and expected properties.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial IntelligenceIntelligent agents, Languages and structures; I.2.5 [**Artificial Intelligence**]: Programming Languages and Software

General Terms

Theory, Languages

Keywords

Declarative goals, Agent programming languages, Agent types

1. INTRODUCTION

An essential characteristic of autonomous agents is their pro-active behaviour [9]. Such agents are assumed to have goals (or objectives) based on which they decide and perform actions. According to this view, an agent's goal denotes a state that the agent desires to realize by means of actions available to it. Different logics have been proposed to characterize goals, to represent and reason about them, and to specify their relations to other agent concepts such as beliefs and actions [2, 3, 6, 5]. These logics, which are mainly used to specify the behavior of autonomous agents, allow the specification of agents that have a certain attitude towards their goals. These attitudes, which are specified as logical axioms, are also known as agent types [3, 6, 4]. For example, an agent can be specified either to hold its goal until it has achieved it (blindly-committed agent type), or to drop the goal if it believes that it can *never* achieve it (single-minded agent type). It is important to note that the specification of many interesting agent types involves temporal aspects, such as in the case of the single-minded agent type, which involves the belief of an agent that it can *never* achieve some goal.

Motivated by these logics, many agent-oriented programming languages have been proposed to implement agents that can represent and reason about goals and are capable of generating plans for their goals [1]. One problem related to the implementation of agent types in agent programming languages is the implementation of the agent types that involve temporal aspects. Some agent types, such as the single-minded agent type, require that the agent is able to reason about its future execution steps to conclude that it believes the goals can *never* be achieved. Defining a programming construct that enables agents to perform such a reasoning task is, however, not an easy task, if not impossible. Moreover, the idea of implementing an agent with *one* specific attitude towards its goals is not attractive, especially when agents can have more than one goal. A more general agent-oriented programming language should therefore allow the implementation of agents that can have different attitudes towards different goals.

In order to implement goal-related agent types and to allow agents to have different attitudes towards different goals, existing programming languages have introduced the notion of goal types. For example, JACK [1] provides programming constructs to implement, among others, test, achieve, insist, and maintain goals, Jason [1] allows the implementation of achieve and test goals, and Jadex [1] provides programming constructs for achieve, query, perform and maintain goals. It is important to note that goal types with the same name captured by these agent-oriented programming languages may have a different meaning across these languages. Moreover, goals in Jadex are represented in XML using a label as the name of the goal, and a number of other parameters. In Jason and JACK, goals are created by generating a particular type of event. Such a notion of goal does not support

*A full version of this paper is available at `http://www.cs.uu.nl/~mehdi/publications.html`

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

(logical) reasoning and therefore does not allow the derivation of subgoals. As we will explain, the derivation of goals is important in the context of agent-oriented programming. Finally, neither JACK nor Jadex provides the formal semantics of their goal types. This is in contrast with our proposal, as we aim at providing a formal semantics for the goal types. Below, we will explain our idea of these goal types and we will briefly address differences and similarities with comparable goal types from other agent-oriented programming languages. A detailed comparison between the goal types of the various languages is beyond the scope of this paper.

In this paper, we discuss three types of goals: perform goals, achieve goals, and maintain goals. In contrast to some other approaches, goals are here assumed to be represented as logical formulas with declarative semantics. This supports reasoning about goals which facilitates the generation of plans for subgoals. Moreover, we explain the *type* of a goal as an agent's attitude towards the goal. We do this with the aim of integrating them in logic-based agent-oriented programming languages. Therefore, the semantics of a goal type will be explained from an operational perspective. Note that the operational semantics perspective can directly be used to design a programming language that allows the implementation of the goal types. Unfortunately, the space limitation does not allow us to present the formal operational semantics of the goal types. Finally, we will discuss some properties of the goal types and show that they have certain desirable and expected properties.

The basic idea in this paper is that agents can have a set of goals based on which planning rules can be selected and applied to generate plans for the goals. In our framework, a planning rule indicates that a specific plan can achieve a certain goal. The application of a planning rule will add the plan of the planning rule to the plan base. Although it is possible to use a specific kind of planning rule for each goal type, we take a different approach and assume *one* kind of planning rule for every goal type. The advantage of such an approach is that it can reduce the number of rules which a programmer has to specify; a planning rule for a certain goal might be used in case this goal is an achieve goal or a maintain goal. As noted, the goals we consider in this paper are declarative goals, i.e., a goal can be represented as a logical formula denoting a desired state. An important advantage of declarative goals is that they allow reasoning, facilitating the generation of subgoals. This is especially advantageous in the context of the planning rules.

Consider, for example, "`FC and CC`" as the goal of an agent, where `FC` stands for having-fuel-in-car and `CC` for having-clean-car. We consider logical consequences of this goal as its subgoals, i.e., "`FC`" and "`CC`" are in this case the subgoals. The idea is, that for this complex goal "`FC and CC`", an agent programmer can implement either one single planning rule or two planning rules, i.e., one for each subgoal. The latter is however more generic, as the two rules can also be applied if the agent has, e.g., the goal `CC`, rather than the complex goal. The mechanism of deriving subgoals of a complex goal can be realized through logical inference mechanisms [7].

2. PERFORM GOALS

The idea of the perform goal is to allow the generation of a set of plans without demanding that the plans must reach the states denoted by the goals. The attitude of an agent toward a perform goal is thus to generate plans after which the goal will be dropped. For example, consider "`FC and CC`" as a perform goal of an agent that can generate the plan to refuel at the gas station `gs1` and the plan to clean the car in the car wash `cw1`. In this case, the agent will generate both plans after which it drops the goal entirely, regardless of the effect of those plans. However, if the agent does not have a planning rule to clean the car and can only generate the plan to refuel, then it will only generate the refuel plan after which the entire goal is dropped. This idea to drop a goal when plans are generated is practical, and similar ideas have been implemented in most agent-oriented programming languages, although sometimes under different names. For example, in JACK and Jason this type of goal is known as achieve goal.

As noted, in our approach the perform goals are represented as logical formulas. If the logical and declarative nature of the perform goals is not essential, then the question arises why we should not use *labels* instead of *logical formulas* to generate a set of plans. The reason to use logical formulas is to allow reasoning about the goals to decide which planning rule to apply. In our approach, the planning rules, which can be applied for all goal types, can be applied only if they are relevant, i.e., if the goal in their heads is logically entailed by an agent's goal. Using labels to represent perform goals requires different types of planning rules and a different procedure for rule application. Moreover, using logical formulas to represent a goal, in general, allows us to design planning rules for atomic or simple goals such that plans for more complex goals can be generated by means of plans for subgoals.

The perform goal can be enriched with a *redo* flag which may be true or false, as is also proposed in Jadex [1]. If the flag is false, then the perform goal will be removed after examining and applying all relevant planning rules. The removal of the perform goals is therefore independent of whether they are achieved or not. However, if the flag is true, then the perform goal remains in the goal base such that the applicable planning rules can be applied once again. This ensures that the plans associated with the perform goal will be generated and performed again. For example, consider a vacuum cleaner that has to clean a number of rooms repeatedly without the ability to check if a room is clean. The perform goal with a true flag can be used to model this type of behaviour. A consequence of this view is that an agent with a perform goal to which a false redo flag is assigned, can drop the goal if the relevant planning rules have tautologies as belief conditions. This is because in our framework a planning rule has a belief condition. A planning rule can only be applied if its belief condition holds.

3. ACHIEVE GOALS

The idea of an achieve goal is to reach the state denoted by it. The emphasis here is that the goal will not be dropped until the state denoted by it is achieved. This is in contrast with perform goals, which can be dropped once the plans generated for it have been executed, regardless of whether the perform goal was achieved. As for perform goals, an agent with an achieve goal will apply relevant planning rules to generate and execute plans. If the achieve goal is not reached after the execution of these plans however, it applies the planning rules again, hopefully generating different

plans, since the circumstances might have changed. Once the achieve goal is reached, it will stop generating and executing plans for this goal. For example, consider an agent with the achieve goal `FC` and two planning rules to generate either the plan to refuel at the gas station `gs1` or the plan to refuel at the gas station `gs2`. This agent will apply one rule and execute the plan to refuel at `gs1`. If the agent succeeds to refuel, the achieve goal will be dropped. Otherwise, it will apply the second rule to generate the plan to refuel at `gs2`. If both plans do not achieve the goal, then it will apply the rules again.

As suggested in [8] and implemented in Jadex, we assign a failure condition to each achieve goal to indicate when the agent should stop trying to achieve the goal and thus drop the goal. For example, `no-fuel-at-gs1-and-gs2` can be the failure condition of the achieve goal `FC`. A similar kind of goal type is introduced in Jadex and in JACK, though in JACK under the name *insist* goal. An achieve goal can thus be dropped under two circumstances: either if its failure condition becomes true, or if the state denoted by it is reached. In both cases, besides the removal of the achieve goal, the plans associated with it will be removed. A planning rule can be applied if the goal in the head of the rule is not achieved yet, if there is no plan for the same subgoal in the plan base, and if the failure condition does not hold.

Following this view, a goal with an achieve type and an impossible failure condition will not be removed from the goal base unless the state denoted by the goal is reached. Moreover, a goal with an achieve type such that the state denoted by the goal cannot be reached and its failure condition never becomes true can be replaced by the same goal with the perform type and a true redo flag. In both cases, relevant planning rules will be applied repeatedly.

4. MAINTAIN GOALS

The idea of a maintain goal is to ensure that a state holds and continues to hold. Plans should be generated and executed if the state denoted by the maintain goal is *threatened* not to hold, rather than waiting and taking action once the state does not hold. The condition under which the maintain goal is threatened not to hold will be called the maintain condition. The agent starts to generate and execute plans when the maintain condition becomes true. This triggering condition can be considered as an alarm to act in order to ensure the maintenance of the state denoted by the maintain goal. It should be noted that there might be no logical relation between the maintain goals and their triggering conditions. This relation depends usually on the application domain. However, we assume that in all agent configurations, if the triggering condition does not hold, then the maintain goal holds. Note that a maintain goal cannot be removed from the goal base.

For example, consider an agent with `FC` as a maintain goal. This means that the agent wants to maintain having a fueled car. In such a case, the maintain condition is the illuminated lamp warning of a shortage of fuel. The agent should in this case generate and execute a refuel plan. If the maintain condition continues to hold after the execution of the plan, because for example the tank station had no fuel, the agent may try to generate and execute another plan, e.g., to go to another tank station to refuel. If all plans are generated and the maintain condition still holds, then there are two options. The agent can either stop generating and executing plans since it has tried all plans, or it can continue to apply the planning rules once again, hopefully generating new plans this time. In order to allow both options, we add a *retry* flag, like in Jadex [1], to the maintain goals. If the flag is true, the agent will try to re-apply planning rules, otherwise it does not. Note that if the maintain condition holds, the state denoted by the maintain goal can either hold or not. A similar idea of maintain goals is implemented in Jadex and JACK, although in JACK the idea is that the maintain condition should hold during the execution of its related task. As a consequence, a goal with the maintain type for which the maintain condition is a tautology and the retry flag is set to false can be replaced by the same goal with the perform type having a false flag. In both cases, the relevant planning rules will be applied once.

5. CONCLUSION

In this paper, we have discussed three types of declarative goals for which we argued that they should be integrated in logic-based agent-oriented programming languages. We have already designed a simple agent-oriented programming language that allows the implementation of these goal types. This language uses a propositional language to represent the agent's goals and beliefs. The formal semantics of the programming language, and thus of the goal types, are defined and it is shown that this semantics has the properties as we have discussed in this paper. Unfortunately, because of the space limitation we could not present the details in this paper.

We plan to extend the logic-based agent-oriented programming language using a (computational) subset of the first-order predicate language instead of using a propositional language. In this way, declarative goals become more expressive. We also have a plan to start implementing an interpreter for an extended version of the programming language which we hope to be available soon.

6. REFERENCES

[1] R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni. *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.

[2] C. Boutilier. Toward a logic for qualitative decision theory. In *Proceedings of the KR'94*, pages 75–86, 1994.

[3] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42, 1990.

[4] M. Dastani and L. van der Torre. Programming BOID-Plan agents: deliberating about conflicts among defeasible mental attitudes and plans. In *Proc. of AAMAS'04*, pages 706–713, New York, USA, 2004.

[5] J.-J. Ch. Meyer, W. van der Hoek, and B. van Linder. A logical approach to the dynamics of commitments. *Arificial Intelligence*, 113:1–40, 1999.

[6] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proc. of KR'91*, 1991.

[7] M. B. van Riemsdijk, M. Dastani, F. Dignum, and J.-J. Ch. Meyer. Dynamics of declarative goals in agent programming. In *Proc. of DALT'04*, 2004.

[8] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative and procedural goals in intelligent agent systems. In *Proc. of KR'02*, 2002.

[9] M. Woolridge. *Introduction to Multiagent Systems*. John Wiley & Sons, Inc., 2002.

Belief Revision for AgentSpeak Agents

Natasha Alechina
University of Nottingham
School of Computer Science
Nottingham, UK
nza@cs.nott.ac.uk

Rafael H. Bordini
University of Durham
Dept. of Computer Science
Durham, UK
r.bordini@durham.ac.uk

Jomi Fred Hübner
Univ. Regional de Blumenau
Dept. Sistemas e Computação
Blumenau, SC, Brazil
jomi@inf.furb.br

Mark Jago
University of Nottingham
School of Computer Science
Nottingham, UK
mtw@cs.nott.ac.uk

Brian Logan
University of Nottingham
School of Computer Science
Nottingham, UK
bsl@cs.nott.ac.uk

ABSTRACT

The AgentSpeak agent-oriented programming language has recently been extended with a number of new features, such as speech-act based communication, internal belief additions, and support for ontological reasoning, which imply a need for belief revision within an AgentSpeak agent. In this paper, we show how a polynomial-time belief-revision algorithm can be incorporated into the ***Jason*** AgentSpeak interpreter. To the best of our knowledge, this is the first attempt to include belief revision within an interpreter for a practical agent programming language.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents, Multiagent Systems*; I.2.5 [**Artificial Intelligence**]: Programming Languages and Software; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Nonmonotonic reasoning and belief revision*

General Terms

Languages

Keywords

Belief Revision, BDI, AgentSpeak, Jason

1. INTRODUCTION

Agents can be viewed as rational entities characterised in terms of their beliefs, desires and intentions and the relationships between them, and the *Belief*, *Desire* and *Intention* (BDI) approach has emerged as perhaps the dominant paradigm in agent research and development [9, 11, 6]. A key problem for BDI agents is the dynamics of beliefs (and consequently of desires and intentions). Agents are continuously presented with a stream of new information (e.g., from their sensors or other agents) requiring changes in their beliefs. In simple cases, the problem of how the agent's beliefs should be updated is straightforward: for example, if an agent's battery sensor reports that the agent's battery is 50% charged, then the agent's existing beliefs about its battery level (and any derived beliefs regarding how far it can travel before recharging) can be simply overwritten with the appropriate new belief(s). However for agents with multiple sources of information and/or more complex belief states a simple 'overwriting' strategy is inadequate. *Belief revision* is necessary when the agent receives conflicting information from different sources, or information which is contradictory in the context of the agent's own beliefs. For example, an agent may be told by Alice that the next group meeting will be held on Wednesday and by Bob that the meeting is on Thursday. If the agent's ontology contains a rule stating that each meeting has a unique date, then, from the agent's point of view, this information is contradictory. In such cases the problem of how the agent's beliefs should be revised to restore consistency is non-trivial.

Current agent-oriented programming languages typically leave belief update up to the programmer [3] and provide no support for belief revision. However as agent's belief states and the inferences performed on them become richer and more complex, such ad-hoc approaches push an ever-greater burden onto the agent programmer. In this paper, we show how to integrate correct and efficient (polynomial-time) belief revision into the architecture of a BDI agent. We focus on the well-known AgentSpeak agent-oriented programming language. However our approach and algorithms are equally applicable to other BDI agent implementations where agent's beliefs can be represented in the form of ground literals and Horn clauses (e.g., rules, or plans in case of AgentSpeak). While there has been some initial work on belief revision in abstract programming languages [10], to the best of our knowledge, the work presented here is the first attempt to include belief-revision within an interpreter for a *practical* agent-oriented programming language.

2. AgentSpeak AND *Jason*

The AgentSpeak(L) programming language was introduced in [8]. It is based on logic programming and provides an elegant abstract framework for programming BDI agents.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

An AgentSpeak agent is defined by a set of *beliefs* (ground literals) which constitute the agent's *belief base*, and a set of plans which form its *plan library*. An AgentSpeak plan has a *head* which consists of a triggering event (the event(s) for which that plan is *relevant*), and a conjunction of belief literals representing a *context*. The conjunction of literals in the context must be a logical consequence of the agent's current beliefs for the plan to be considered *applicable* when triggered. A plan also has a *body*, which is a sequence of basic actions (i.e., atomic operations the agent can perform so as to change the environment), (sub)goals that the agent has to achieve (or test) when the plan is executed, and belief changes that can serve as "mental notes". Plans are triggered by the addition or deletion of beliefs due to communication or perception of the environment, or due to the addition or deletion of goals as a result the execution of plans triggered by previous events.

The ***Jason*** interpreter implements the operational semantics of AgentSpeak as given in, e.g., [5]. ***Jason*** in written in Java, and its IDE supports the development and execution of distributed multi-agent systems [4].Recent work has made important additions to AgentSpeak, which have also been (or are in the process of being) implemented in ***Jason***. Some of these new features, such as speech-act based communication, internal belief additions, and support for reasoning with ontological knowledge, presuppose a *belief revision* capability as part of agents' reasoning cycle. For example, in [7], an extension of AgentSpeak was proposed which incorporates ontological reasoning within the AgentSpeak interpreter. In the extended language, called AgentSpeak-DL, queries to the belief base are more expressive as their results do not depend only on the agent's explicit knowledge but also on what can be inferred from the ontology; the search for a relevant plan for a particular event is more flexible as this is not based solely on unification, but also on the subsumption relation between concepts; and agents may share knowledge by using web ontology languages such as OWL. These new capabilities can result in inconsistencies in an agent's belief base which cannot be eliminated by a simple 'overwriting' strategy.

The current implementation of ***Jason*** provides a basic belief update capability. This deletes any perceptual beliefs which are no longer perceived from the belief base and adds any new percepts as new perceptual beliefs. Belief additions executed within a plan as well as any information communicated by trusted sources is simply added to the belief base (without checking for consistency). While this approach ensures that the agent's perceptual beliefs remain consistent, it is incapable of handling derived inconsistencies, e.g., if a property is asserted of an individual which is inconsistent with the corresponding concept description. The elimination of such inconsistencies remains the responsibility of the agent developer and must be coded anew for each agent application. Our aim is to extend the ***Jason*** implementation with a belief revision capability which automatically restores belief base consistency in the face of derived inconsistencies.

3. THE BELIEF REVISION ALGORITHM

We have two main objectives when introducing belief revision in AgentSpeak. First the algorithm should be theoretically well motivated, in the sense of producing revisions which conform to a generally accepted set of postulates characterising rational belief revision. Second, we want the resulting language to be practical, which means that the belief revision algorithm must be efficient.

In the belief revision literature, these two goals have traditionally been seen as incompatible. Our approach draws on recent work [2] on efficient (polynomial-time) belief revision algorithms for rule-based agents which satisfy the well-known Alchourrón, Makinson and Gärdenfors (AGM) postulates [1] characterising rational belief revision and contraction. In this section we briefly describe the linear-time belief contraction algorithm introduced in [2]. We then explain how belief revision can be defined in terms of contraction.

We distinguish two kinds of contraction operation. AGM (or coherence) contraction of the agent's belief base K by a belief literal A is defined as the removal of A and sufficient literals from K such that A is no longer derivable using the set of plan instances executed to date. Reason-maintenance contraction additionally removes any beliefs for which A is (recursively) the sole support. To facilitate both AGM and reason-maintenance contraction, the inferential relationships between the beliefs comprising the agent's belief base are represented as a directed graph, where the nodes are beliefs and *justifications*. A justification consists of a belief and a *support list* containing the context (and possibly the triggering event) of the plan used to derive this belief: for example, the assertion of a belief A by a plan with context B triggered by a belief addition event C (or derived by an ontology rule $B, C \rightarrow A$) would have a justification $(A, [B, C])$. Foundational beliefs which were not derived, have a justification of the form $(D, [])$. In the graph, each justification has one outgoing edge to the belief it is a justification for, and an incoming edge from each belief in its support list. We assume that each support list s has a designated *least preferred* member $w(s)$. Intuitively, this is a belief which is not preferred to any other belief in the support list, and which we would be prepared to discard first, if we have to give up one of the beliefs in the list. We assume that we have constant time access to $w(s)$.

The algorithm for AGM contraction by A is then:

```
for each of A's outgoing edges
   to a justification (C, s),
   remove (C,s) from the graph
for each of A's incoming edges
   from a justification (A, s),
   if s is empty, remove (A, s);
   else contract by w(s);
remove A.
```

To implement reason-maintenance type contraction, we also remove beliefs which have no incoming edges.

In [2], it was shown that the contraction operator defined by the algorithm satisfies the AGM postulates for contraction (K-1)–(K-4) and (K-6) [1] (the recovery postulate, (K-5), is not satisfied). Closure under consequence in classical logic is replaced with closure in a weaker logic W which has a single inference rule of generalised modus ponens.

The algorithm runs in time $O(kr + n)$, where r is the number of plans, k the maximal number of beliefs in any support list, and n the number of literals in the belief base. Indeed, the upper bound on the number of steps required to remove justifications corresponding to plan instances is $r(k + 1)$ (one constant time operation for each belief in a context of the plan and one for the belief asserted by the plan). Removing all justifications corresponding to foundational beliefs costs n steps. The last step in the contraction algorithm (removing a belief) is executed at most n times.

The AGM (resp. reason-maintenance) revision of the set of literals in the agent's belief base K can then be defined as: add A, close under consequence and AGM (resp. reason-maintenance) contract by all contradictions. Note that for agents which reason using ground literals and Horn clauses, closure under consequence can be computed in polynomial time.

4. BELIEF REVISION IN *Jason*

Future releases of ***Jason*** will incorporate an implementation of the belief revision algorithm presented above as a user-selectable

option. The new implementation clearly distinguishes between *belief update*, i.e., the addition and deletion of perceptual beliefs following perception, and *belief revision* proper, i.e., the elimination of inconsistencies following belief additions by plans or ontological reasoning. A belief addition may be discarded (as at present) or may result in the deletion of some other belief(s) in order to allow the new belief to be consistently added to the belief base. Which beliefs are deleted is determined by a user-specified preference order (see below). Here, we assume that the agent's plans contain no explicit belief deletions, and that deletion of beliefs only occurs as a result of belief update and revision.

Each belief literal added to the belief base has an associated "dependencies list" (the literals that allowed the derivation of the literal in question), and a "justifies list" (the literals which the literal in question justifies, that is, it appears in their dependencies list). For example, if the plan that generated the belief change, say $+bl$, has the form "`@p` te: l_1 `&` ... `&` l_n `<-` bd", where te is a triggering event and bd a plan body, the support list of the corresponding justification is simply the ground literals from the plan context, "[l_1,...,l_n]". Note that if the triggering event, te, is itself a belief addition, the literal in te is included together with the context literals in the support list. Further, for each literal $l_1, \ldots, l_n$ we include the newly added belief in the literal's "justifies" list. We also record the time at which the belief was added as part of the justification. When deleting perceptual beliefs during belief update, we remove any justification which has the deleted perceptual belief in its support list, and mark the justified beliefs as self supporting (foundational) beliefs. This additional bookkeeping reflects the special status of perceptual beliefs: perceptual beliefs can trigger or justify the addition of other beliefs, but their deletion is not in itself a reason for removing a derived belief.

The belief revision algorithm also requires the definition of a partial order expressing the preference, or entrenchment, of each belief. To allow for user customisation, this is defined as a separate method that can also be overridden. The default method gives preference to perceived information over communicated information (as also happens in [10]), and in case of information from the same source, it gives preference to newer information over older information.

The implementation described above is conservative in revising only the agent's belief state. The agent's plans are considered part of the agent's program and are not revised (though revising, e.g., plans received from other agents would be an interesting extension). Similarly, when revising beliefs derived using ontological rules, we assume the ontology used by the agent to be immutable and consistent, and that it is consistent with every other ontology it references.

5. CONCLUSIONS AND FUTURE WORK

Experience has shown that correct and efficient implementations of agent-oriented programming languages are key in the widespread adoption of agent technology. As agent programming languages become richer, it becomes harder for the agent programmer to ensure that the belief states of agents developed using these languages are consistent. In this paper we briefly summarised the rationale for including automatic belief revision in a BDI agent programming language, and described the integration of an efficient (polynomial-time) belief-revision algorithm into the ***Jason*** AgentSpeak interpreter. Our approach is theoretically well motivated in the sense of producing revisions which conform to a generally accepted set of postulates characterising rational belief revision. We believe that other BDI agent-oriented programming languages and their platforms [3] which leave the problem of maintaining a consistent belief state entirely to agent programmer, can also benefit from our approach.

In future work, we plan to further investigate the implications of integrating belief revision into the AgentSpeak execution cycle (e.g., allowing belief deletions in plans, ontological reasoning). On the more practical side, we plan to develop large-scale agent applications to assess the performance of ***Jason*** with belief revision.

Acknowledgements

Rafael Bordini gratefully acknowledges the support of The Nuffield Foundation (grant number NAL/01065/G).

6. REFERENCES

[1] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet functions for contraction and revision. *Journal of Symbolic Logic*, 50:510–530, 1985.

[2] N. Alechina, M. Jago, and B. Logan. Resource-bounded belief revision and contraction. In *Proceedings of the 3rd International Workshop on Declarative Agent Languages and Technologies (DALT 2005)*, 2005.

[3] R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications*. Number 15 in Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer, 2005.

[4] R. H. Bordini, J. F. Hübner, et al. ***Jason****: A Java-based agentSpeak interpreter used with saci for multi-agent distribution over the net*, manual, release version 0.7 edition, August 2005. `http://jason.sourceforge.net/`.

[5] R. H. Bordini and Á. F. Moreira. Proving BDI properties of agent-oriented programming languages: The asymmetry thesis principles in AgentSpeak(L). *Annals of Mathematics and Artificial Intelligence*, 42(1–3):197–226, 2004.

[6] D. Kinny. Agents — the challenge of relevance to the it mainstream. In *Programming Multi-Agent Systems, Second International Workshop ProMAS 2004*, LNCS Vol. 3346, pages 38–43. Springer-Verlag, 2005.

[7] A. F. Moreira, R. Vieira, R. H. Bordini, and J. Hübner. Agent-oriented programming with underlying ontological reasoning. In *Proceedings of the 3rd International Workshop on Declarative Agent Languages and Technologies (DALT 2005)*, 2005.

[8] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proceedings of the Seventh Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96)*, LNAI Vol. 1038, pages 42–55, London, 1996. Springer-Verlag.

[9] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484, 1991.

[10] R. M. van Eijk, F. S. de Boer, W. van der Hoek, and J.-J. C. Meyer. Information-passing and belief revision in multi-agent systems. In *Intelligent Agents V — Agent Theories, Architectures, and Languages*, LNCS Vol. 1555, pages 29–45, Berlin, 1999. Springer-Verlag.

[11] M. Wooldridge. *Reasoning About Rational Agents*. MIT Press, 2000.

Plan Patterns for Declarative Goals in AgentSpeak

Jomi Fred Hübner
Univ. Regional de Blumenau
Depto. de Sist. e Computação
Blumenau, SC, Brazil
jomi@inf.furb.br

Rafael H. Bordini
University of Durham
Dept. of Computer Science
Durham, UK
r.bordini@durham.ac.uk

Michael Wooldridge
University of Liverpool
Dept. of Computer Science
Liverpool, UK
mjw@csc.liv.ac.uk

1. INTRODUCTION

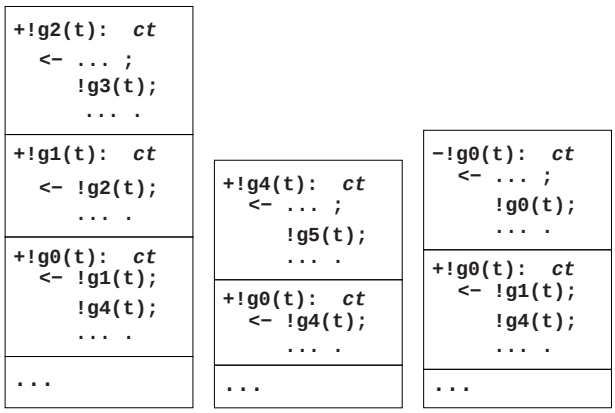
In this paper, we consider the use of *declarative goals* in the AgentSpeak programming language, originally introduced in [6]. By a declarative goal, we mean a goal that *explicitly* represents a state of affairs to be achieved, in the sense that, if an agent has a goal $p(t_1, \ldots, t_n)$ then it expects to eventually believe $p(t_1, \ldots, t_n)$, and only then can the goal be considered achieved. Currently, although goals form a central component of AgentSpeak programming, they are only *implicit* in the plans defined by the agent programmer. For example, there is no explicit way of expressing that a goal should be maintained until a certain condition holds; such temporal goal structures are hidden within the plan structures themselves. While one possibility would be to extend the language to introduce an explicit construct for declarative goals and a goal base, we show that this is unnecessary. We introduce a number of *plan patterns*, corresponding to common types of explicit temporal/conditional (declarative) goal structures, and show how these can be mapped into AgentSpeak code. Thus, a programmer or designer can conceive of a goal at the declarative level, and this goal will be "translated" via these patterns into standard AgentSpeak code.

2. PLAN FAILURE

In order to present plan patterns for declarative goals, the *plan failure* handling mechanism implemented in ***Jason*** [2], and some specific *internal actions* used for dropping goals need to be presented.

We identify three cases of plan failure. The first cause of failure is a *lack of relevant or applicable plans*, which can be understood as the agent "not knowing how to do something". The second is where a test goal fails; that is, where the agent "expected" to believe something about the world, but that did not hold as expected. The third is when an action within a plan fails. Regardless of the reason for a plan failing, the interpreter generates a goal deletion event (represented by `-!`g) if the achievement of the goal g has failed. This event may be handled by a plan written to deal with such failure; e.g., the plan "`-!`g `: true <- !`g`.`" will attempt the achieve goal g again when a plan to achieve g has failed.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.



(a) Initial Intention (b) After .dropGoal(g1(t),true) (c) After .dropGoal(g1(t),false)

Figure 1: Internal Actions for Dropping Goals

In the next section, we also make use of the following predefined internal actions. The first, `.dropGoal(`g`,true)`, terminates g successfully, as if g were somehow achieved, while `.dropGoal(`g`,false)` terminates g with failure, meaning that g was impossible to achieve (in this case, a failure event for the plan that generated goal g is generated). It is perhaps easier to understand how these actions work with reference to Figure 1. The figure shows the consequence of each of these internal actions being executed (the plan where the internal action appeared is not shown; it is likely to be within another intention). Note that the state of the intention affected by the execution of one of these internal actions, as shown in the figure, is not the immediate resulting state (at the end of the reasoning cycle where the internal action was executed) but the most significant next state of the changed intention.

3. DECLARATIVE GOAL PATTERNS

Although goals form a central component of the AgentSpeak conceptual framework, the language itself does not provide any explicit constructs for handling goals with complex temporal structures. For example, a system designer and programmer will often think in terms of goals such as "maintain P until Q becomes true", or "prevent P from becoming true". Creating AgentSpeak code to realise such complex goals has, to date, been largely an ad hoc process, dependent upon the experience of the programmer. Below, we define a number of *plan patterns* for declarative goals — that is, complex combinations of plan structures which frequently occur in

actual scenarios — and to show how these can be realised in terms of AgentSpeak code. As we shall see, such patterns can be used to implement, in a systematic way, certain types of declarative goals and the types of commitments they represent, as discussed e.g. by Cohen and Levesque [4].

As an initial motivational example for declarative goals, consider a robot agent with the goal of being at some location (represented by the predicate `l(`X`,`Y`)`) and the following plan to achieve this goal, provided the battery charge is greater than 20% (the predicate `bc/1` stands for "battery charge", and `go` identifies an action that the robot is able to perform in the environment):

```
+!l(X,Y) : bc(B) & B > 0.2 <- go(X,Y).
```

Sometimes, using an AgentSpeak plan as a procedure can be a quite useful programming tool. Thus, in a way, it is important that the AgentSpeak interpreter does not enforce any declarative semantics to its only (syntactically defined) goal construct. However, in the plan above, `l(`X`,`Y`)` is clearly meant as a *declarative goal*; that is, the programmer expects the robot to believe `l(`X`,`Y`)` (by perceiving the environment), if the plan executes to completion. If this does not happen because, e.g., the environment is dynamic or non-deterministic, the goal cannot be considered achieved.

This is where our plan patterns for declarative goals help to express this type of situation without adding new syntactic/semantic constructs to AgentSpeak. We can easily transform the above procedural goal into a declarative goal by adding a corresponding *test goal* at the end of the plan's body, as follows:

```
+!l(X,Y) : bc(B) & B > 0.2
   <- go(X,Y);
      ?l(X,Y).
```

This plan only succeeds if the goal is actually achieved; if the given (procedural) plan executes to completion (without failing) but the goal happens not to be achieved, the test goal at the end will fail.

This solution forms a *plan pattern*, and other patterns can be applied to solve other similar problems. Our approach to the inclusion of declarative goals in the AgentSpeak programming language is inspired by the successful adoption of design patterns in object-oriented design. To represent such patterns for AgentSpeak, we will define skeleton programs with meta variables. For example, the pattern for a simple declarative goal, as the one used in the robot's location goal, is defined by the following AgentSpeak plan:

```
+!g : c <- p; ?g.
```

Here, g is a meta variable that represents the declarative goal, c is a meta variable that represents the context expression stating in which circumstances the plan is applicable, and p represents the procedural part of the plan body (i.e., a course of actions to achieve g). Note that, with the introduction of the final test goal, this plan to achieve g finishes successfully only if the agent believes g after the execution of plan body p.

To simplify the use of the patterns, we introduce *pattern rules*, which rewrite a set of AgentSpeak plans into a new AgentSpeak program according to a given pattern. The following pattern rule, called **DG** (Declarative Goal), shows how procedural goals are transformed into declarative goals. The pattern rule name is followed by the parameters which need to be provided by the programmer, besides the actual code (i.e., sequence of plans) on which the pattern will be applied.

```
+!g : c1 <- p1.
+!g : c2 <- p2.
...
+!g : cn <- pn.
```
DG$_g$ $(n \geq 1)$

```
+!g : g <- true.
+!g : c1 <- p1; ?g.
+!g : c2 <- p2; ?g.
...
+!g : cn <- pn; ?g.
+g : true <- .dropGoal(g, true).
```

Essentially, this rule adds `?`g to the end of every plan in the plan library that has `+!`g as triggering event, and creates two extra plans. The first additional plan checks whether goal g has already been achieved — in such case, there is nothing else to do. The second additional plan is triggered when the agent perceives that g has been achieved while it is executing any of the courses of action p_i $(1 \leq i \leq n)$ which aim at achieving g; if that occurs, the plan being executed in order to achieve g can be immediately terminated (with success). The internal action `.dropGoal(`g`, true)` can be used for this.

In this pattern, when one of the plans to achieve g fails, the agent gives up achieving the goal altogether. However, it could be the case that for such a goal, the agent should try another plan to achieve it, similarly to the "backtracking" plan selection mechanism available in platforms such as JACK [9] and 3APL [5]. The *Backtracking Declarative Goal* rule (**BDG**), defines this pattern based on a set of AgentSpeak plans $\mathcal{P}$ transformed by the **DG** pattern (each plan in $\mathcal{P}$ is of the form `+!`g `:` c `<-` p):

$\mathcal{P}$

BDG$_g$

DG$_g(\mathcal{P})$

```
-!g : true <- !g.
```

The last plan of the pattern catches a failure event, caused when a plan from $\mathcal{P}$ fails, and then tries to achieve that same goal g again. Notice that it is possible that the same (failed) plan is selected again, causing a loop if the plan contexts have not been carefully programmed (and the environment does not change).

In the pattern above, despite the various alternative plans, the agent can still end up dropping the intention with the goal g unachieved if none of those plans happen to be applicable. Conversely, with a *blind commitment goal*, the agent can drop the goal only when it is achieved:

$\mathcal{P}$

BCG$_g$

BDG$(\mathcal{P})$

```
+!g : true <- !g.
```

The **BCG** pattern is based on the **BDG** pattern and extends it by adding a plan that ensures that the agent will still pursue goal g even when all known plans to achieve g are not applicable.

For most applications, **BCG**-style fanatical commitment is too strong. For example, if a robot has the goal to be at some location, it is reasonable that it can drop this goal in case its battery charge is getting very low; in other words, the agent has realised that it has become impossible to achieve the goal, so it is useless to keep trying. This is very similar to the idea of a persistent goal in the work of Cohen and Levesque: a persistent goal is a goal that is maintained as long as it is believed not achieved, but still believed possible [4]. The drop condition f (e.g., "low battery charge") is

used in the Single-Minded Commitment (**SMC**) pattern to allow the agent to drop a goal if it becomes impossible:

$$\frac{\mathcal{P}}{\mathbf{BCG}_{g,BDG}(\mathcal{P})}\ \mathbf{SMC}_{g,f}$$

```
+f : true <- .dropGoal(g, false).
```

This pattern extends the **BCG** pattern by adding the drop condition represented by the literal f in the last plan. If the agent comes to believe f, it can drop goal g, signalling failure. This effectively means that the plan which generated goal g must itself fail (as g is now impossible to achieve). However, there might be an alternative for that plan which does not depend on g, so failure handling for that plan may take care of such situation.

As we have a failure drop condition for a goal, we can also have a success drop condition, e.g., because the motivation to achieve the goal has ceased to exist. Suppose a robot has the goal of going to the fridge because its owner has asked it to fetch a beer; if the robot realises that its owner does not want a beer anymore, it should drop the goal [4]. The belief "my owner wants a beer" is the *motivation* m for the goal.

Of course we can combine the drop condition and motivation to create a goal which can be dropped if it has been achieved, has become impossible to achieve, or the motivation to achieve the goal no longer exists. This *Open-Minded Commitment* (**OMC**) pattern is defined as follows:

$$\frac{\mathcal{P}}{\mathbf{BCG}_{g,BDG}(\mathcal{P})}\ \mathbf{OMC}_{g,f,m}$$

```
+f : true <- .dropGoal(g, false).
-m : true <- .dropGoal(g, true).
```

For example, a drop condition could be "no beer at location (X,Y)" (denoted below by `~b(X,Y)`), and the motivational condition could be "my owner wants a beer" (denoted below by `wb`). Consider the initial plan below, which is intended to achieve goal `l(X,Y)`:

```
+!l(X,Y) : bc(B) & B > 0.2 <- go(X,Y).
```

When the pattern $\mathbf{OMC}_{l(X,Y),\sim b(X,Y),wb}$ is applied to the plan above, we get the following program:

```
+!l(X,Y) : l(X,Y) <- true.
+!l(X,Y): bc(B) & B > 0.2
   ← go(X,Y); ?l(X,Y).
+!l(X,Y) : true <- !l(X,Y).
-!l(X,Y) : true <- !l(X,Y).
+~b(X,Y) : true <- .dropGoal(l(X,Y), false).
-wb : true <- .dropGoal(l(X,Y), true).
```

The approach of defining goals with complex temporal structures by means of patterns can also be used to specify more complex declarative goals, such as, for example, maintenance goals (the agent needs to ensure that the state of the world will always be such that g holds) and sequenced goals (i.e., when various instances of a goal should not be adopted concurrently).

4. CONCLUSIONS

In this paper we have showed that sophisticated types of goal structures can be implemented in the AgentSpeak language, with only the extensions (and extensibility mechanisms) available in ***Jason***. This is done by combining certain types of AgentSpeak plans, forming certain patterns for each type of goal and commitment towards goals that agents might have. Our approach is to take advantage of the simplicity of the AgentSpeak language, using only its well-known support for procedural goals plus the idea of patterns to support the use of declarative goals in AgentSpeak programming.

Despite the use of pre-defined internal actions such as `.dropGoal` (which had already been conceived for use with procedural goals in ***Jason***), our approach requires neither syntactical nor semantical changes in the language (as done, for example, in [10, 3]), nor the usual addition of a goal base (see e.g. [7]). Van Riemsdijk *et al.* [8] also pointed out that it is possible to build declarative goals using the procedural goals available in 3APL. Their idea correspond to our **BDG** pattern. In this paper, we have defined various other types of declarative goals and represented them as *patterns* of AgentSpeak plans. Another advantage of our approach is that, as complex types of goals are mapped down to plain AgentSpeak code using patterns, programmers can change the patterns to suit their needs, or indeed create new patterns easily.

Acknowledgements

Rafael Bordini gratefully acknowledges the support of The Nuffield Foundation (grant number NAL/01065/G).

5. REFERENCES

[1] R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors. *Multi-Agent Programming: Languages, Platforms, and Applications*. Number 15 in Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer, 2005.

[2] R. H. Bordini, J. F. Hübner, and R. Vieira. ***Jason*** and the Golden Fleece of agent-oriented programming. In Bordini et al. [1], chapter 1.

[3] L. Braubach, A. Pokahr, W. Lamersdorf, and D. Moldt. Goal representation for BDI agent systems. In R. H. Bordini, M. Dastani, J. Dix, and A. E. Fallah-Seghrouchni, editors, *Second Int. Workshop on Programming Multiagent Systems: Languages and Tools (ProMAS 2004)*, pages 9–20, 2004.

[4] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3):213–261, 1990.

[5] M. Dastani, M. B. van Riemsdijk, and J.-J. C. Meyer. Programming multi-agent systems in 3APL. In Bordini et al. [1], chapter 2.

[6] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In W. Van de Velde and J. Perram, editors, *Proc. of MAAMAW'96, 22–25 January, Eindhoven, The Netherlands*, number 1038 in LNAI, pages 42–55, London, 1996. Springer-Verlag.

[7] B. van Riemsdijk, M. Dastani, and J.-J. C. Meyer. Semantics of declarative goals in agent programming. In F. Dignum *et al.*, editors, *Proc. of AAMAS 2005*, pages 133–140. ACM, 2005.

[8] M. B. van Riemsdijk, M. Dastani, and J.-J. C. Meyer. Subgoal semantics in agent programming. In C. Bento, A. Cardoso, and G. Dias, editors, *Proc. of EPIA 2005, Covilhã, Portugal, December 5-8, 2005*, volume 3808 of *LNCS*, pages 548–559, 2005.

[9] M. Winikoff. Jack intelligent agents: An industrial strength platform. In Bordini et al. [1], chapter 7.

[10] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative and procedural goals in intelligent agent systems. In *Proc. of 8th Int. Conf. on Principles of Knowledge Representation and Reasoning*, 2002.

A Service-Oriented Language for Programming Mobile Agents

Hervé Paulino
Faculdade de Ciências e Tecnologia,
Universidade Nova de Lisboa, Portugal
herve@di.fct.unl.pt

Luís Lopes
Faculdade de Ciências,
Universidade do Porto, Portugal
lblopes@dcc.fc.up.pt

ABSTRACT

In this paper we present Mob, a service-oriented scripting language for programming mobile agents in distributed systems. The main feature of the language is the integration of the service-oriented and the mobile agent paradigms. Mob is encoded onto a process calculus with a well studied semantics which provides us with a tool to prove the soundness of the language relative to the underlying calculus.

Keywords

Mobile Agents, Service-Oriented, Programming Language

1. INTRODUCTION AND MOTIVATION

Service-Oriented programming departs from the object-oriented paradigm by de-coupling data and processing. Services are provided in a transparent way for clients, requiring only knowledge of the contract (the service's interface). One of the main advantages of service-oriented programming is that it provides a framework on which to develop component-based systems. These have recently received a lot of attention for distributed systems, namely with the .NET [14], Jini [19] and Openwings [9] platforms.

Another major technology for Web applications is that of *mobile agents*. Mobile agents are computations that have the ability to travel through a network, by halting their execution, saving their state and then restoring it in a new host. In this paradigm, mobile agents move towards the resources (e.g., data, servers) and interact locally, unlike the usual communication paradigms (e.g., client-server), that require costly remote sessions to be maintained.

Programming languages for mobile agents come in two flavors: those *designed by hand* and those based on formal systems. In the first set we have systems such as Aglets [7], Mole [16] and Voyager [4] that mostly extend Java classes to define an agent's behavior, and scripting languages such as D'Agents [5] or Ara [13]. Providing a demonstrably sound semantics for these systems is rather difficult given the gap between the implementation and an adequate formal model.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Languages in the second set are based on formal systems, mostly some form or extension of the π-calculus [6, 8]. This allows researchers to build solid specifications for these programming languages and prove the languages correct *by design* by providing adequate encodings onto a base calculus. Examples of such languages have been implemented in recent years, namely, JoCaml [3], TyCO [18], X-Klaim [2], Nomadic Pict [20], Acute [10] and Alice [15].

In [12] we first introduced Mob, a programming language for developing applications based on mobile agents. Here we extend Mob with another main abstraction, *services*, thus uniting two major abstractions for Web applications. The main novelties introduced in this paper, in particular relative to [12], are as follows: a) services, described as interfaces implemented by agents, and (mobile) agents are the main abstractions of the language. Agents provide and require services and are bound to these dynamically as they move through the network; b) the Mob language has been encoded in TyCO a form of the π-calculus extended with distribution and mobility of resources and with a well-studied semantics [1,18]. This provides a tool to prove the soundness of the operational semantics of the language, thus providing a form of language security not readily available in the other languages discussed; c) the encoding onto the process calculus provided a full specification of the front-end of the compiler for Mob and allowed us to use the TyCO run-time system to run compiled Mob programs.

2. THE LANGUAGE

One of the main abstractions of Mob is the **agent**. Agents may be viewed as special objects, with a run-time associated, that provide/require services to/from the network (other agents). Agents are handled in Mob programs much like objects are handled in object-oriented languages.

The **service** is another main abstraction. An agent **implements** services and, simultaneously, **requires** services provided by other agents. There is absolutely no distinction between clients and servers. To access a service, a programmer must get a binding for an agent that provides it. The binding is obtained dynamically through the primitive **bind** that asks a network name service for an agent that provides the service. When the binding is received, interaction through method invocation can happen.

Agents may move through the network and this is controlled explicitly, at high-level, by the programmer using a primitive **go**. The movement of an agent involves moving an entire virtual machine and its state to the target host in the network. The execution resumes on arrival at the target

$$
\begin{array}{rcl}
P & ::= & \textbf{agent}\ X(\tilde{x})\ \textbf{implements}\ \tilde{S}\ \textbf{requires}\ \tilde{S}\ M \\
& | & \textbf{service}\ S\ \{\tilde{m}\} \quad | \quad \textbf{requires}\ \tilde{S} \quad | \quad \textbf{class}\ X(\tilde{x})\ M \\
& | & \textbf{join}\ (x) \quad | \quad \textbf{go}\ (e) \quad | \quad \textbf{lock}\ (x) \quad | \quad \textbf{unlock}\ (x) \\
& | & \textbf{if}\ (e)\ \{\ P\ \}\ \textbf{else}\ \{\ P\ \} \quad | \quad \textbf{while}\ (e)\ \{\ P\ \} \\
& | & \textbf{for}\ (x\ \textbf{in}\ e)\ \{P\} \quad | \quad \textbf{break} \quad | \quad \textbf{return}\ (e) \\
& | & \textbf{exit}() \quad | \quad x = e \quad | \quad P\ ;\ P \\
M & ::= & \{m_1(\tilde{x}_1)\ \{P_1\}\ \ldots\ m_n(\tilde{x}_n)\ \{P_n\}\} \\
e & ::= & e\ bop\ e \quad | \quad uop\ e \quad | \quad \textbf{self} \quad | \quad x \quad | \quad e.x \\
& | & c \quad | \quad \textbf{null} \quad | \quad C \\
C & ::= & \textbf{new}\ X(\tilde{e}) \quad | \quad \textbf{fork}\ \{\ P\ \} \quad | \quad \textbf{bind}\ (S) \quad | \quad \textbf{host}() \\
& | & \textbf{exec}\ (\tilde{e}) \quad | \quad C.m\ (\tilde{e}) \quad | \quad \{\tilde{e} : \tilde{e}\} \quad | \quad [\tilde{e}]
\end{array}
$$

Figure 1: Syntax of the Mob programming language.

host, without intervention from other agents.

Agents are multi-threaded. Thread creation and synchronization is supported through the **fork** and **join** primitives. Another form of synchronization, on data-structures, is provided by the instructions **lock** and **unlock** that support a simple form of mutual exclusion in data access.

Interaction with external services is supported with **exec** which is used to implement extensions to the core Mob language to support external functionality. These external services may be implemented in other languages, such as Java, or access network services, such as WWW queries.

The remainder of the language constructs provide fairly standard support for control flow, expressions and external calls. Due to space constraints we will briefly describe the syntax and semantics of the language. The full definition of the language and its operational semantics may be found in [11]. The syntax for a Mob program is presented in figure 1. The language defines a set reserved words identified in bold-face. The main syntactic categories are: constants (booleans, integers and strings) ranged over by c; variables, ranged over by x; agent and class identifiers, ranged over by X; service identifiers, ranged over by S; method names and method collections, ranged over by m and M, respectively; expressions, ranged over by e; commands, ranged over by C; arrays and maps, ranged over by $\{\tilde{e} : \tilde{e}\}$ and $[\tilde{e}]$, respectively, and; instructions, ranged over by P. A sequence of elements of a given grammatical category α is denoted by $\tilde{\alpha}$. The concrete syntax of Mob imposes some syntactic restrictions to the above.

We exemplify the syntax with a small example of a server and a mobile client for a simplified Time service. The server provides the service Time with a single method getTime(). Note that the **main** method may be empty since Mob agents run as daemons and some external action is required to terminate their execution.

```
service Time { getTime }
agent TimeServer() implements Time {
   main () { }
   getTime() { return exec ("getTimeApplication");}
}
new TimeServer();
```

The client requires this service and when run, it takes a sequence of hosts, and performs a cycle in which it moves to each of them, setting their time accordingly with the central time from the TimeServer.

```
agent TimeClient(hostList) requires Time {
   main () {
      timeServer = bind (Time);
      for (h in hostList) {
         go (h);
         exec ("setTimeApplication", "" ^ timeServer.getTime());
      }
}
new TimeClient([host1,...,hostn]);
```

The semantics for the Mob core language is provided in the form of an abstract machine that describes state transitions between network configurations. Each network configuration is composed of a set of agents running concurrently plus a name service for binding services to agents. Each agent has its own heap, code, a set of threads running concurrently and a set of threads suspended on resources. Besides their own local environment, threads share the agent's heap and use it to synchronize their access to resources.

For a complete formal specification of the semantics of the language we refer the reader to [11].

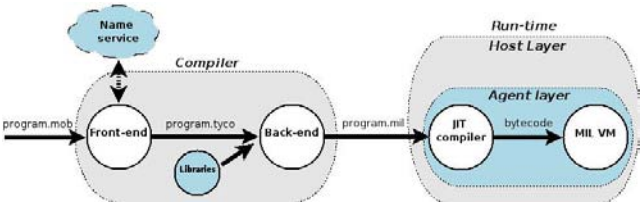


Figure 2: Compiling and running a Mob program.

3. THE COMPILER AND RUN-TIME

The compiler is divided in two main stages (figure 2). First, the *front-end* of the compiler takes the Mob source code and outputs the corresponding code in the TyCO language, as specified by the encoding of Mob in TyCO. The front-end of the compiler performs type inference on the Mob source code and in particular finds the types for both the implemented and the required services for agents. At this point the name service is contacted and a type-check is performed. The types inferred by the compiler are matched with those assumed for the service in the network. If the agent implements a service yet not registered, the interface provided becomes the *de facto* interface for that service. If type-checking succeeds, the source Mob program is transformed into a program written in the TyCO language.

The *back-end* of the Mob compiler is just the compiler for the TyCO language. It takes the TyCO code generated by the front-end and produces code written in an intermediate language called MIL (Multi-threaded Intermediate Language [17]), which is compiled *just-in-time* by the run-time system before being executed. This scheme allows the run-time to check type information in the MIL code before running it.

The run-time system is divided into two layers (figure 2). The top layer is the *host layer* and is implemented as a service that must run on every Mob-enabled host in a network. This layer is responsible for managing agents within a host and supporting their mobility. More specifically, it provides the means to create, execute, marshall, send, unmarshall and receive agents. The second layer is the *agent layer* and implements agents and the name service (also an agent). All are executed with the TyCO run-time system. Each agent is implemented with two concurrent threads: the first is an instance of the MIL virtual machine that runs the MIL code for the agent; the second handles network communication for the MIL virtual machine.

4. RELATED WORK

Of the languages more closely related to Mob none adheres to the service-oriented paradigm.

JoCaml [3] is a programming language based on the Join-calculus that provides support for distributed and mobile agent based applications. The language uses a custom vir-

tual machine and supports the migration of trees of computations (an agent and its tree of sub-agents) between hosts in a network. Just as in Mob, type-checking is mostly done at compile time except for interaction with other modules which is done dynamically. The required type information is annotated in the source program.

The X-Klaim [2] language is an implementation of the Klaim model with ad-hoc extensions to incorporate higher-order constructs, asynchronous reading of tuple-spaces and hierarchical structured networks. Programs in X-Klaim are compiled into Java classes that resort to a package, Klava, to run. Mobile agents in X-Klaim are processes with a single execution flow, rather than the multi-threaded agents found in Mob. This makes the migration of multi-threaded agents a complex and user aware operation.

Nomadic Pict [20] is perhaps the closest to ours in that it grows from another process calculus based language, Pict, and adding primitives for programming mobile computations such as agent creation, agent migration and asynchronous communication between agents. Nomadic Pict also focuses on verification and a proof of correctness for an instance of the infrastructure has been achieved. Despite the similarities our aim is to provide a scripting language that abstracts away from network location dependent information. In this respect we feel that Mob, even in its core language is higher level than Nomadic-Pict.

Acute [10] is a programming language for mobile agents built on top of Objective Caml. The language provides type-safety through a partially static/dynamic type checking scheme. Moving computations is achieved through an atomic operation that captures a collection of threads in a structure (a *thunk*) that can afterwards be marshalled and moved across the network. This contrasts to Mob where marshalling of objects or agents is transparent to the programmer. The Mob service on that node will be responsible for unmarshalling the agent and restart it. In this respect Acute provides a finer, lower level, control over migration and marshalling than Mob.

Alice is a programming language based on Oz and its implementation, Mozart [15] (itself based on Standard ML). The functionality provided by Alice is similar to that of Acute. The front-end of the Alice compiler produces Oz intermediate code so that the Oz run-time, Mozart, is used to run Alice applications. The approach is similar to the one we use since the front-end of the Mob compiler produces TyCO code that is then executed with TyCO's run-time system.

5. CONCLUSIONS

We have produced an implementation of the Mob language compiler and its run-time system. Currently we are working to prove the soundness of the Mob language. In other words, given an encoding map $[\![\cdot]\!]$, from Mob abstract-machine states into TyCO programs, we wish to prove the following conjecture:

Conjecture (Soundness) Let N and N' be network configurations in the Mob abstract-machine. If $N \rightarrow N'$ (*reduces to* in Mob) then, $[\![N]\!] \equiv [\![N']\!]$ (*is congruent to* in TyCO) or $[\![N]\!] \rightarrow [\![N']\!]$ (*reduces to* in TyCO).

6. REFERENCES

[1] A. Ravara et all. Lexically Scoping Distribution: What You See Is What You Get. In *F. of Global Computing*, volume 85(1) of *ENTCS*. Elsevier Science, 2003.

[2] L. Bettini, R. D. Nicola, and R. Pugliese. X-Klaim and Klava: Programming Mobile Code. In *TOSCA 2001*, volume 62. Elsevier Science, 2001.

[3] S. Conchon and F. L. Fessant. Jocaml: Mobile Agents for Objective-Caml. In *ASA/MA'99*, pages 22–29. IEEE Computer Society, 1999.

[4] G. Glass. Overview of Voyager: ObjectSpace's Product Family for State-of-the-art Distributed Computing, Tech. Report CTO ObjectSpace, 1999.

[5] R. S. Gray. Agent Tcl: A Transportable Agent System. In *CIKM'95 Workshop on Intelligent Information Agents*, 1995.

[6] K. Honda and M. Tokoro. An Object Calculus for Asynchronous Communication. In *ECOOP'91*, volume 512 of *LNCS*, pages 141–162. Springer-Verlag, 1991.

[7] D. B. Lange and M. Oshima. *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley, 1998.

[8] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes (parts I and II). *Information and Computation*, 100(1):1–77, 1992.

[9] Openwings. Openwings: A Service-Oriented Component Architecture for Self-Forming, Self-Healing, Network-Centric Systems. `http://www.openwings.org`, 2001.

[10] P. Sewell et all. Acute: High-Level Programming Language Design for Distributed Computation. `http://www.cl.cam.ac.uk/users/pes20/acute/`.

[11] H. Paulino and L. Lopes. Mob Core Language and Virtual Machine, Tech Report DCC-2005-05, DCC - FC and LIACC, Universidade do Porto, 2005. `http://www.dcc.fc.up.pt/Pubs/TR05/dcc-2005-05.pdf`.

[12] H. Paulino, L. Lopes, and F. Silva. Mob: A Scripting Language for Mobile Agents Based on a Process Calculus. In *ICWE 2003*, volume 2272 of *LNCS*, pages 40–43. Springer-Verlag, 2003.

[13] H. Peine and T. Stolpmann. The Architecture of the Ara Platform for Mobile Agents. In *MA'97*, volume 1219 of *LNCS*, pages 316–323. Springer-Verlag, 1997.

[14] D. Platt. *Introducing Microsoft .NET, Third Edition*. Microsoft Press, 2003.

[15] G. Smolka. Concurrent Constraint Programming Based on Functional Programming. In *Programming Languages and Systems*, volume 1381 of *LNCS*, pages 1–11. Springer-Verlag, 1998.

[16] F. Straber and J. Baumann. Mole - A Java Based Mobile Agent System. In M. M., editor, *Special Issues in Object Oriented Programming*, pages 301–308, 1997.

[17] V. Vasconcelos and L. Lopes. A Multithreaded Assembly Language and Virtual Machine. Unpublished.

[18] V. Vasconcelos, L. Lopes, and F. Silva. Distribution and Mobility with Lexical Scoping in Process Calculi. In *HLCL'98*, volume 16(3) of *ENTCS*, pages 19–34. Elsevier Science, 1998.

[19] J. Waldo. The Jini Architecture for Network-Centric Computing. *Commun. ACM*, 42(7):76–82, 1999.

[20] P. T. Wojciechowski and P. Sewell. Nomadic Pict: Language and Infrastructure Design for Mobile Agents. *IEEE Concurrency*, 8(2):42–52, 2000.

ActorNet: An Actor Platform for Wireless Sensor Networks

YoungMin Kwon, Sameer Sundresh, Kirill Mechitov and Gul Agha *
Department of Computer Science
University of Illinois at Urbana Champaign
Urbana, IL 61801, USA
{ykwon4 | sundresh | mechitov | agha}@cs.uiuc.edu

ABSTRACT

We present *actorNet*, a mobile agent platform for wireless sensor networks (WSNs). WSNs are well-suited to multi-agent systems: agent autonomy reduces the need for communication, saving precious energy. Mobile agents are also an intuitive technique for remotely reprogramming sensors deployed in the field. However, implementing agent programs directly on a WSN is complicated by the many limitations of sensor nodes, including limited memory, slow processors, low bandwidth and finite energy. *ActorNet* eases development by providing an abstract environment for lightweight concurrent object-oriented mobile code on WSNs. As such, it enables a wide range of new dynamic applications, including fully customizable queries and aggregation functions, in-network interactive debugging and high-level concurrent programming on the inherently parallel sensor network platform. Moreover, *actorNet* cleanly integrates all of these features into a fine-tuned, multi-threaded embedded Scheme interpreter that supports compact, maintainable programs—a significant advantage over primitive stack-based virtual machines.

1. INTRODUCTION

A wireless sensor network is a system of autonomous sensor nodes which interact via wireless communication channels. Canonical WSN applications include environment monitoring, target tracking and structural health monitoring [9, 3, 2]. These applications are enabled by the unique features of WSNs: independent energy sources, wireless communication and large-scale deployment of inexpensive devices. Unfortunately, these features also impose severe constraints on WSNs, including limited energy and network bandwidth, small memory sizes and low processing power.

Agent systems are one promising solution to such problems. For example, an autonomous agent can make intelligent decisions on the spot without consulting a central base station, saving energy and bandwidth. Indeed, existing WSN applications such as real-time target tracking already use an agent-based approach [8]. Several challenges become apparent when building agent programs for WSN platforms. Some wireless sensor nodes have only 4KB of SRAM—a severe constraint for even moderately complex applications. Furthermore, sensor node operating systems typically support only non-blocking I/O for performance reasons, complicating application development. The tight coupling between applications and the operating system makes cross-platform interoperability and code migration difficult. In addition, WSN in-network storage services [10], while similar to agent coordination services such as tuple spaces [4], cannot meet their full potential without a supporting agent platform.

ActorNet is an agent platform that addresses these issues, making the development of agent-based solutions to sensing problems significantly easier. Specifically, *actorNet* provides services such as virtual memory, context switching and multi-tasking to support a highly-expressive yet efficient agent language for WSNs. Several agent primitives can be implemented on top of *actorNet*. For example, *actorNet* supports coordination among agents in the asynchronous communication model, typically employed in WSNs due to the gap between communication and computation speeds. Synchronization between concurrent agents in an asynchronous setting can be performed via synchronizers or join continuations, which have previously been defined in an Actor language similar to *actorNet* [1].

2. ACTOR LANGUAGE

The top layer of the *actorNet* platform is an actor language interpreter. The actor language uses the well-known S-expression syntax. Notable primitive operators include `par`, `send`, `msgq` and `callcc`, which are used extensively in this paper.

The *actorNet* language is based on the syntax and semantics of Scheme, with extensions specific to actor computation. The `par`, `send` and `msgq` operators are not found in Scheme. `Par` creates separate actors to evaluate each argument expression in parallel and returns a list of the created actors' identifiers. While these actors remain on the same *actorNet* platform, they share parts of their environments, allowing them to communicate efficiently. If an actor migrate to another platform, it can communicate back via message passing. The `send` operator provides a simple, abstract mechanism to send messages to an actor. A deep copy is made of the message data to prevent any dependence on

*This research has been supported by the DARPA IXO NEST Award F33615-01-C-1907 and the ONR MURI Award N0014-02-1-0715.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

the source host. For example, `(send (list 100 x))` sends all data reachable from variable `x` to an actor with id 100. An actor can access its message queue by calling the `msgq` operator, which returns the list of messages the actor has received.

The `callcc` operator accesses the *current continuation* (CC)—an abstraction of the rest of the program remaining to execute. For example, the CC of the expression `(add 1 (mul 2 ↓ 3))` at the ↓ mark can be regarded as a single-parameter function `c1`: `(lambda (x) (c2 (mul 2 x)))`, where `c2` is an another single-parameter function `(lambda (x) (add 1 x))`. In general, the CC can be regarded as a stack of single-parameter functions. The operand of `callcc` is a single-parameter function: when `callcc` is called the CC is passed to the operand function.

In *actorNet*, the CC and the value that will be passed to it form the state of an actor. Because an actor can read its current continuation, it can duplicate itself or migrate to another platform voluntarily by sending its continuation–value pair to another actor. The other actor simply evaluates the continuation on the value to duplicate the sender's behavior. Using these primitives we can easily and intuitively define the agent migration function of Section 3.

```
 1 (rec (move path temp) ;;return path, max temp
 2    (seq
 3       ;;broadcast measure actors to neighbors
 4       (send (list 0 measure (id)))
 5       (delay 100) ;;wait for 10 sec
 6       (  (lambda (maxt)
 7             (par
 8                ;;if it arrives at an maximal point
 9                (cond (le (car maxt) temp)
10                ;;then return the temp along the path
11                   (return migrate path temp)
12                ;;else move to the highest temp. node
13                   (move
14                      (cond (equal path nil)
15                         (cons launch path)
16                         (cons (io 0) path))
17                      (migrate
18                         (cadr maxt)   ;;node id
19                         (car maxt)))) ;;temp
20                (setcdr (msgq) nil)))) ;;reset msgq
21          ;;find the max temp. and the node
22          (max (cdr (msgq)) (list 0 0)))))
```

Figure 1: An example actor program that migrates to a point of maximal temperature in a WSN and returns the temperature back to the base station

3. EXAMPLE

ActorNet is an implementation of the actor model of distributed computing [1] for wireless sensor networks. We present an example application that demonstrates the utility of mobile agents in a sensor network setting. *actorNet*'s support for coordination amongst mobile agents. Our example consists of an actor migrating through the WSN, in search of a local maximal temperature when the temperature is assumed to be continuously changing in space—a typical environment monitoring task. The actor in this example autonomously selects it migration path based on environmental information and reports the final result to a base station.An outline of the application follows.

1. An actor A broadcasts to its neighbors a small actor that measures the temperature at a node and sends back the result.

2. Actor A determines the local maximum temperature and migrates itself to the corresponding node. When it migrates to another node, A records its point of origin so that it can forward the maximum temperature reading back along the path it followed.

3. When it arrives at a point of maximal temperature, A migrates back to the base station. Upon arrival at the base station it prints out the temperature value.

Note that in this example we do not need any support for message routing. An actor locally broadcasts and moves itself to its neighbor with the greatest temperature, and the return path is constructed as it migrates from node to node.

Let us first construct a `migrate` function that makes an actor migrate to another node and continue its execution. The state of an actor can be considered as a pair of a continuation and a value to be passed to the continuation. Using this explicit representation, an actor can easily migrate itself to a neighboring node by sending its current continuation as obtained via `callcc`. Each *actorNet* platform hosts a `launcher` actor which regards messages sent to it as programs and evaluates them. More precisely, `migrate` can be defined as follows:

```
1  (lambda (adrs val) ;; migrate
2     (callcc
3        (lambda (cc)
4           (send (list adrs cc (list quote val))))))
```

Figure 1 shows the code for our temperature-search example. The program first broadcasts a `measure` actor that reads a temperature at a remote node and sends back the reading. The sender then waits for 10 seconds and subsequently checks its message queue, `msgq`, for the measurement. No other work is needed for synchronization. The `measure` actor can be encoded simply as

```
1     (lambda (ret) ;;measure
2        (send (list ret (io 1) (io 0)))).
```

The `(io 1)` call returns a temperature reading and the call `(io 0)` returns the node identifier. The `launcher` actor of a remote platform will evaluate this function with the return address, which is the `id` function call of the 4th line of Figure 1. Although the `measure` actor in this application is simple, it could potentially be any arbitrarily complex function. That is, one can easily distribute complex subprocesses to other nodes and later collect the results in the form of messages. This simple example demonstrates how *actorNet* facilitates multi-agent computation in sensor networks.

Returning to Figure 1, the `move` function takes as arguments a return path and the current maximum temperature reading. Migration occurs after evaluating the second parameter. Line 13 shows how the actor migrates to another node: it first appends its node id—`(io 0)`—to the return path and then migrates to the node where the greatest temperature was read. When it arrives at a point of maximal temperature, it returns the temperature value using the `return` function, listed below.

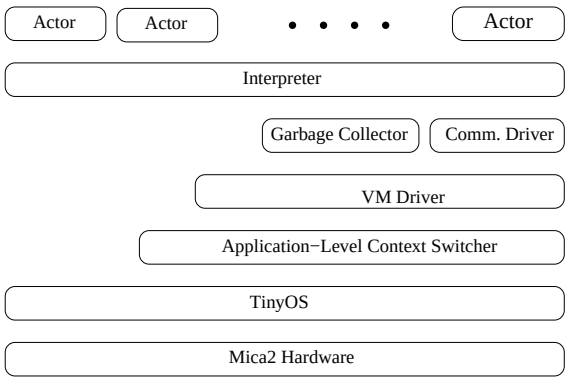


Figure 2: Architecture of the actorNet platform (Mica2 node)

```
1  (rec (return migrate path temp)
2     (cond (equal path nil)
3        (print temp)
4        (return migrate (cdr path)
5                    (migrate (car path) temp))))
```

The `return` function is similar to `move`: it migrates across the nodes along the return path. Incidentally, the `move` and `return` operations can be used to implement a variety of ant-based routing algorithms [6].

4. ACTORNET PLATFORM DESIGN

An *actorNet* is a collection of *actorNet platforms* running on sensor nodes or PCs. Platforms are networked via local wireless channels and the Internet. Each platform hosts several computing elements, called actors, which are able to interact with each other both locally and over the network. Like other virtual machines, the *actorNet* platform provides a uniform computing environment for all actors, regardless of hardware or operating system differences.

Figure 2 depicts the layered architecture of a sensor node *actorNet* platform. Actors only use the interpreter module directly; thus implementation details are hidden from actor programs. Lower-level services are necessary to reconcile the desired properties of simplicity and platform-independence in the high-level language with the specifics of the wireless sensor network environment. The layered architecture alleviates some of the complexity of application development for sensor networks, which currently involves a significant amount of low-level programming due to the tight coupling between the application and the operating system. *ActorNet* aims to provide a stricter decoupling of applications from the operating system, enabling a sensor node to safely load and execute multiple agents, while at the same time simplifying application development. In the same vein, *actorNet*'s support for application-level multitasking makes available such intuitive techniques as blocking I/O, which are usually sacrificed for the sake of efficiency by low-level WSN operating systems such as TinyOS. *ActorNet* maintains this efficiency by introducing threads which are preempted when blocking on I/O.

Several other limitations of present-generation WSN platforms are significant when designing a high-level agent language in this setting, particularly limited memory sizes. We examine representative hardware and software WSN platforms and develop services needed to overcome these limitations.

4.1 WSN Environment

Currently, the sensor node *actorNet* platform is specifically designed for Mica2 hardware. The Mica2 has an 8 MHz 8-bit ATmega 128L CPU with 4 KB of SRAM, 128 KB of program flash memory and 512 KB of serial flash [5]. The 4 KB SRAM space is shared by the stack, heap, and all TinyOS components' static variables. Limited memory heavily constrains applications requiring several coordination services. Application code, large constant tables and logged data are loaded in the flash memory units. ActorNet uses the serial flash as a virtual memory space. Flash read operations are fast, but writes are slow—it takes ∼15ms to write a 128-byte page.

TinyOS is a lightweight operating system for sensor nodes written primarily in NesC [7]. The system is structured as a collection of modules which are statically linked together based on a component specification. Modules consist of statically-allocated variables and three kinds of program blocks: `command`s, `event`s, and `task`s. Service requests are typically split-phase: a caller invokes a command, which returns quickly; once the request is satisfied, the service notifies a corresponding event procedure in the caller. This technique enables higher application throughput as compared to single-threaded blocking I/O. Long-running procedures are explicitly executed as *tasks*, which are scheduled in series and run to completion. Since only interrupts can preempt tasks or lower-priority interrupt handlers, processes must be segmented into sequences of tasks if they are to run concurrently.

4.2 Platform Services

In response to the challenges presented by the WSN environment, we have designed platform services allowing efficient memory management and blocking I/O operations.

Virtual Memory. Since the 4 KB SRAM space on the Mica2 is insufficient for many applications, *actorNet* provides a *virtual memory* (VM) subsystem. We build a page structure on the serial flash area of the Mica2 and use an inverted page table to access pages stored in an LRU cache in SRAM. The current implementation of the VM driver does not perform memory compaction, because the *actorNet* platform does not frequently allocate large chunks of memory; the primary data types are small: bytes, integers, single precision floating point numbers, and pairs of 16-bit addresses (cons cells).

Application-Level Context Switching. The non-blocking command-event I/O model employed by TinyOS is infeasible to support directly in *actorNet*, since every virtual memory access potentially involves split-phase flash memory I/O. For example, let us consider the code in Figure 3. The code on the left assumes blocking I/O: `bar` calls `foo` and `foo` calls `read`, which performs I/O. Without blocking I/O, we must resort to something like the right hand side of Figure 3: the `read` and its callers must be divided into two parts. The problem is that every time `read` is called, the functions along the call chain must be divided into two parts. Furthermore, one cannot use *automatic local variables*: all data must now be declared as static variables outside of the function def-

```
foo() {
   int a;
   ...
   read();
   ...
}
bar() {
   int a;
   ...
   foo();
   ...
}
```

```
int foo_a;
int bar_a;
prefoo() {
   ...
   preread();
}
postfoo() {
   postread();
   ...
}
```

```
prebar() {
   ...
   prefoo();
}
postbar() {
   postfoo();
   ...
}
```

Figure 3: Example program with blocking I/O (left) and without (right)

inition. This makes programs unnecessarily complicated, and also results in inefficient memory usage: local variables must be exclusively allocated as static data rather than as temporary values on a shared stack.

We have devised an application-level context switching mechanism that allows efficient blocking I/O. To explore the utility of the context switching mechanism, consider the following NesC program for `read()`. Note that there is a spin-loop in the `read()` function waiting for `isFlashReadDone` to become true.

```
read() {
   ...
   while(!isFlashReadDone)
      yield();
   return flashData;
}
```

```
task loop() {
   resume();
   post loop();
}
```

With our context switching mechanism, the `yield()` call in `read()` causes control to exit from the `resume()` call of the `loop` task. Thus, TinyOS can schedule other tasks and process pending events. Later, when the `loop` task is scheduled again and `resume()` is called, control continues following the `yield()` call in `read()` as if it had just returned from `yield()`. Note that we do not need to divide the application program into two phases as in Figure 3. Hence the yield–resume mechanism eases development of maintainable applications. To preserve portability and modularity, the context switching mechanism is implemented purely as an application-level service; we do not modify the TinyOS scheduler.

Multi-Phase Garbage Collector. The *actorNet* platform provides a *mark and sweep garbage collection* (GC) mechanism. System-level support for garbage collection has many benefits: it eases application development, eliminates the chance of memory leaks, protects other applications from misbehaving actors, and reduces the actor code size. However, for embedded applications it also has one serious drawback: GC may not always occur at the most opportune moment. This problem becomes especially severe for the *actorNet* platform, because it takes a significant amount of time to write pages to the flash memory. When virtual memory is lightly loaded, the marking step can be done quickly. However, the sweep step must check and restore all pages. In order to solve this problem, we divide the sweep step into many short phases. Combined with the context switching functionality, this greatly reduces the impact of garbage collection on application performance.

5. CONCLUSIONS

We have developed *actorNet*, a mobile agent platform for WSNs. *ActorNet* provides high-level abstractions for coordination in distributed systems, and a set of essential features to develop agent systems on WSNs. Though the current implementation is fully functional, it still has some limitations. One of the biggest remaining challenges is fault tolerence: as message transmission in WSNs is via local broadcast, using a simple message acknowledgement mechanism is infeasible since it requires that each node knows all its neighbor ids'. We are currently investigating techniques to implement a negative-acknowledgement-based rebroadcast mechanism while maintaining as much performance as possible. The virtual memory and multi-tasking environments provided by *actorNet* open the possibility for more advanced coordination mechanisms such as tuple spaces, which can be built on top of existing distributed storage services for sensor networks. Despite the current limitations, *actorNet* provides powerful, efficient high-level services for developing agent systems on WSNs.

6. REFERENCES

[1] G. A. Agha, I. A. Mason, S. F. Smith, and C. L. Talcott. A foundation for actor computation. In *Journal of Functional Programming*, volume 7, pages 1–72. Cambridge University Press, 1997.

[2] A. Basharat, N. Catbas, and M. Shah. A framework for intelligent sensor network with video camera for structural health monitoring of bridges. In *Proceedings of Third IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2005.

[3] R. R. Brooks, P. Ramanathan, and A. M. Sayed. Distributed target classification and tracking in sensor networks. In *Proceedings of the IEEE*, 2003.

[4] N. Carriero and D. Gelernter. Linda in context. In *Communications of the ACM*, volume 32, pages 444–458, 1989.

[5] Crossbow Technology, Inc. http://www.xbow.com/.

[6] M. Dorigo, G. D. Caro, and L. Gambardella. Ant algorithms for discrete optimization. In *Artificial Life*, pages 137–172, 1999.

[7] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *In Proceedings of Programming Language Design and Implementation (PLDI) 2003*, June 2003.

[8] B. Horling, R. Mailler, J. Shen, R. Vincent, and V. Lesser. Using Autonomy, Organizational Design and Negotiation in a Distributed Sensor Network. In V. Lesser, C. Ortiz, and M. Tambe, editors, *Distributed Sensor Networks: A multiagent perspective*, pages 139–183. Kluwer Academic Publishers, 2003.

[9] A. Mainwaring, J. Polastre, R. S. D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

[10] S. Mazumdar. Fast range queries using pre-aggregated in-network storage. Masters' thesis, University of Illinois at Urbana Champaign, 2004.

Agent-oriented engineering and methodologies

Integrating Agent-Oriented Methodologies with UML-AT

Rubén Fuentes, Jorge J. Gómez-Sanz, Juan Pavón
Dept. Sistemas Informáticos y Programación
Universidad Complutense Madrid
c/ Profesor José García Santesmases s/n, 28040 Madrid, Spain
{ruben, jjgomez, jpavon}@sip.ucm.es

ABSTRACT

There are many methodological approaches for Agent-Oriented Software Engineering, each one focusing on some features of multi-agent systems, but leaving others underdefined. For this reason, it would be interesting to have the possibility of applying different methods, according to their suitability to each particular problem domain and system view. Here, a key issue is how to integrate the information resulting from different methods in a common specification. We propose the use of an intermediate language called UML-AT, which would enable bidirectional transformations between models in different languages. These transformations allow representing views of those models in the language of choice at every moment. UML-AT is a UML profile based on the Activity Theory (AT) framework, which includes the concepts to describe societies of actors that are both autonomous and intentional. The translation with UML-AT mappings makes the integration process independent of any given methodology. At the same time, these mappings can be a basis to study missing features in design languages, according to what is needed in the study of the intentional and social aspects in multi-agent systems and human organizations following AT. This integration architecture by means of UML-AT mappings is illustrated with a case study about a legacy tourism application. It uses a support tool that automates the translation.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures – *domain-specific architectures, languages, patterns.*

General Terms

Documentation, Design, Standardization, Languages.

Keywords

Multi-Agent Systems, Method Engineering, Conceptual Mappings, Activity Theory, UML-AT, Tropos, INGENIAS.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

1. INTRODUCTION

Multi-Agent Systems (MAS) are used in heterogeneous application domains. Generally, a MAS is considered as a group of intentional autonomous entities (agents) that interact to satisfy some goals. Despite of this, distinct needs generate differences in methodologies about what the relevant features of MAS are and how to analyze them. Agent-Oriented Software Engineering (AOSE) includes methodologies that, for instance, focus on goals (e.g. KAOS [4]), the mental state of agents (e.g. AAII/BDI [9]), expert systems (e.g. MAS-CommonKADS [8]), recursive interconnected tasks (e.g. DESIRE [2]) or interactions (e.g. INGENIAS [15]). This specificity makes sense to explore concrete aspects of MAS. However, the development of real systems needs to gather information about several of these aspects and, frequently, from different points of view (e.g. customers, managers, analysts, or deployers). Agent-oriented methodologies facilitate the study of some features on which they are focused, whereas they give little support in the analysis of other features that they consider as less relevant.

In the absence of a general modelling language for MAS, developers can cope with the previous problem using models from different methodologies. For every view of their MAS, they could choose the languages, methodologies, and tools that are best-suited, according to their own requirements and experience. This approach has the problem of how to integrate the information from different sources: the information of a view is gathered with some models, but it influences/appears on other views, which may be specified using different models. For instance, developers can use KAOS [4] to elicit the goals and models from DESIRE [2] to specify the task decomposition; in this setting, goals from KAOS should be available in DESIRE models, without duplicating information or extra specifying effort. This idea is analogous to the method engineering approach (as in [1]), which defines a generic metamodel that is a representational infrastructure to unify existing methodologies into a single specification. Although in [1] the authors illustrate its application with two agent models, this is done in an informal way and no tool exists yet to work with it automatically. We similarly consider the definition of such a generic language, which is UML-AT. Nevertheless, instead of deriving it conceptually, our proposal is the result of experience in its use for validation of intentional and social properties of MAS [5]. Moreover, we offer tool support with the *INGENIAS Development Kit* (IDK) [15], which serves to facilitate its application and to validate the specification of UML-AT.

UML-AT [5] is defined as an UML profile that includes the concepts of the Activity Theory (AT) [16]. AT provides a

conceptual and analytical framework to understand motivations, individual-society relationships, and networks of interacting activity systems embedded in organizations. Considering AT as the foundation for this intermediate language comes from the fact that it shares with the agent-oriented approach the view of systems as organizations of intentional entities. Besides, there are precedents of its application to AOSE, for instance, in requirements engineering [7], the analysis of contradictions [5], and the design of agent communication infrastructure [14]. These features of AT are shared with other theories from Social Sciences that has been used in agent-oriented research. Some relevant examples are the Reciprocal Role Taking by G. H. Mead, the Actor-Network Theory by Callon & Latour, the foundation of the BDI by M. E. Bratman, or the Biological Theory of Knowledge by H. Maturana (see [11] for an overview of theories from Social Sciences in Distributed Artificial Intelligence). The main difference is that AT is a whole trend of social research, which has a holistic approach covering from the social movements to the individual cognition in the frame of historical development. This allows the study of MAS and their human context from different points of view using an integrating theoretical foundation.

The use of UML-AT as a *lingua franca* for the translation has several advantages in AOSE. First, the translation process becomes independent of any specific methodology. The process remains unchanged and UML-AT evolves to satisfy the map requirements of the agent-oriented methodologies. Second, it reduces the number of mappings required for a methodology to those from and to UML-AT. The set of underlying AT concepts is rich enough to support most of the intentional and social concepts considered in AOSE. In fact, AT analysis of these features in human groups is usually more sophisticated than the current studies in agent-oriented methodologies. So, the third advantage is that UML-AT is a test-bed to detect missing concepts for these features in design languages, given that human organizations are a natural metaphor for MAS. This kind of studies improves the methodologies and also helps to choose the proper modelling language for specific views of a MAS, as these studies evaluate its expressive power.

The rest of this paper is organized as follows. Section 2 briefly introduces AT and the UML-AT language. Section 3 describes the mappings and the process to specify them. Section 4 shows the translation process that uses those mappings while Section 5 describes the support tool that implements them. This process and tools are applied to a case study about the use of Tropos and INGENIAS methodologies for the development of a MAS for cultural information, in Section 6. Finally, Section 7 discusses some final issues about the integration framework.

2. THE ACTIVITY THEORY

Activity Theory (AT) [16] is an analytical framework from Social Sciences intended to study human groups through their contextualised practices. AT considers people embedded in a context that is temporal, cultural, and social. People's practices and their context reciprocally influence each other and this interaction is called *activity*.

The *activity* is the fundamental unit of analysis in AT. An *activity* is a transformation process driven by people's needs. The *activity* satisfies these needs with the *outcome* obtained from the transmutation of an *object*. The active agent that carries out the *activity* is known as the *subject*. Those *subjects* who share an *object* constitute a *community*.

The principle of mediation plays a central role in the AT. An *activity* always contains various artefacts (e.g. instruments, signs, procedures, machines, materials, laws, forms of work, or organisations). These artefacts have a mediating role: relations between elements of an *activity* are not directed, but mediated. *Tools* shape the way human beings interact with their context. A *tool* can be anything used in the transformation process of the *object*, including both material tools and tools for thinking. The relationship between *subject* and *community* is mediated by *rules* (e.g. laws, social conventions, or norms), and the relationship between *object* and *community* is mediated by the *division of labour*, which establishes how the *activity* is distributed among the members of the *community*. The *division of labour* considers the role each individual in the *community* plays in the *activity*, the power each wields, and the tasks each is held responsible for.

These concepts constitute the common core of the vocabulary for AT as agreed in its research community. To complete the vocabulary of the AT for MAS, some other entities (e.g. *activity systems*, *artifacts*, or *roles*) and relationships (e.g. mediations, contributions, or decompositions) were made explicit (since they already appear in some AT studies) or added to the previous core. An *activity system* is the context of an *activity*, and includes all the elements that appear in the AT core. An *artifact* is a generalization that can represent any concept of an *activity system*.

The set of AT concepts for MAS were translated into UML using profiles [13] to facilitate its use in AOSE processes. The resulting language is called UML-AT. The representation of an *activity system* in UML-AT, plus the *artifact* concept and mediation relationships, appears in Figure 1. The full specification of UML-AT can be found at http://grasia.fdi.ucm.es/at/uml-at.

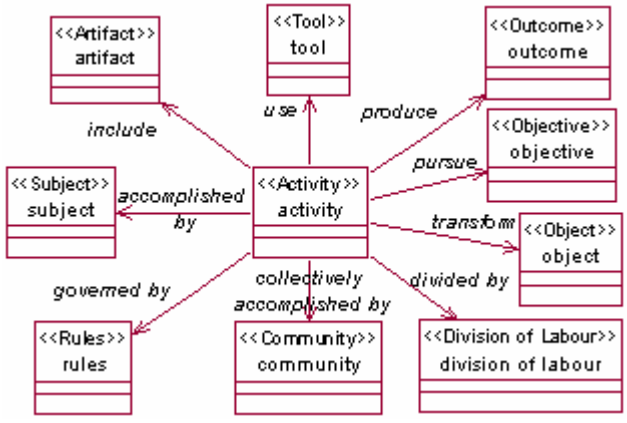


Figure 1. *Activity system* with UML-AT.

In its current version, UML-AT supports the most widespread concepts used in AOSE to describe intentional and social features of MAS. This allows representing the concepts of a given agent-oriented methodology as UML-AT diagrams and,

thus, translating MAS specifications obtained with that methodology to UML-AT. In the case of concepts from agent-oriented methods that cannot be mapped as UML-AT patterns, they will lead to the addition of new AT concepts to UML-AT.

3. MAPPINGS WITH UML-AT

Mappings contain the information required to transform the specifications of a MAS between UML-AT and the language of an agent-oriented method. A mapping is a correspondence from structures in a source language to structures in a target language, where both patterns can contain constant and variable values. These structures exchange information through shared variables. The following discussion about mappings is illustrated with the example in Figure 2. It shows a mapping from a relationship defined with the INGENIAS methodology [15] to UML-AT. This is an excerpt of the XML files for mappings that use the tool that is later introduced in Section 5.

Figure 2 shows the main elements required to define mappings. The tags *language_source* and *language_target* indicate the languages of the source specifications (e.g. INGENIAS) and the translated ones (e.g. UML-AT). A *match* describes a correspondence, in a given translation level, between two structures: the *source pattern*, which is enclosed within the *source* tags, and the *target pattern*, which is surrounded by the *target* tags. The translation level gives a sorting for the application of the mappings, as the Section 4 explains. The first translation cycle use the mappings of level 0; every subsequent cycle increases the level by 1. Both INGENIAS and UML-AT work with information represented as graphs. Their languages are specified with a meta-modelling language called GOPRR. GOPRR [10] describes *graphs* in terms of *relationships*, whose extremes are *roles* that connect with *entities*. All of these elements can have related *properties*. The elements within *source* and *target* tags reflect these primitives. A *relationship GTFailsource* in an INGENIAS view is translated to a *relationship AT_Contribute* in an UML-AT diagram, which has the same name as the *GTFailsource* (shared variable "*v2*") and a value *Avoided* of the attribute *contribution*.

```
<correspondences xmlns:xsi=
  "http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="Mapping.xsd">
  <language_source name="INGENIAS"/>
  <language_target name="UML_AT"/>
  <match> <level value="1"/>
    <source>
      <entity>
        <name> <v> v0 </v> </name>
        <type> Fact </type>
      </entity>
...
      <relationship>
        <name> <v> v2 </v> </name>
        <type> GTFails </type>
        <role>
          <name> <v> v3 </v> </name>
          <type> GTFailssource </type>
          <attribute>
            <name> origin </name>
            <value> true </value>
          </attribute>
          <attribute>
            <name> player </name>
            <value> <v> v0 </v> </value>
          </attribute>
        </role>
...
      </relationship>
    </source>
    <target>
      <entity>
        <name> <v> v0 </v> </name>
        <type> AT_Object </type>
      </entity>
...
      <relationship>
        <name> <v> v2 </v> </name>
        <type> AT_Contribute </type>
        <attribute>
          <name> Contribution </name>
          <value> Avoided </value>
        </attribute>
        <role>
          <name> <v> v3 </v> </name>
          <type> AT_Contribute_Source </type>
          <attribute>
            <name> origin </name>
            <value> true </value>
          </attribute>
          <attribute>
            <name> player </name>
            <value> <v> v0 </v> </value>
          </attribute>
        </role>
...
      </relationship>
    </target>
  </match>
</correspondences>
```

Figure 2 Excerpt of mapping from INGENIAS to UML-AT.

The example in Figure 2 corresponds to a simple correspondence. Real mappings usually relate graphs. The reason for that is that the meaning of the elements in the specifications strongly depends on their relationships with others. Very simple mappings can generate translations that are not completely accurate and introduce mistakes in the resulting specifications.

The main difficulty when developing these mappings is to determine what the correct correspondence is for a given *source pattern*. This task demands a deep knowledge of the concepts involved in UML-AT and the agent-oriented methodology, but it has only to be done once for a given methodology. There are some available heuristics to carry out this process. These are based on the description of AT concepts from the point of view of the agent paradigm that Table 1 summarizes.

Table 1. AT concepts for the agent paradigm.

AT concepts	Definition with the agent paradigm
Activity System	Set of modelling elements representing a total or partial view of the MAS.
Activity	Task that pursues some relevant objective for an agent or group of agents in the MAS.
Subject	Agent, role, or organization. In general, whatever active element that behaves according to the Rationality Principle[1] can be modelled as a *subject*.
Object	An element transformed by a task and that can be considered as the reason/origin for that task. It is a prerequisite for the task.
Outcome	The product of a task that satisfies the goal of that task or is an evidence to evaluate that goal.
Objective	Need satisfied by a task.
Tool	A resource used in a task that is neither the *object* nor the *outcome* of that task. It is a prerequisite for the task.
Community	Set of *subjects* that interact in the workflows of a MAS.
Rules	Set of norms that affect the behaviour of the MAS and do not belong to the *division of labour*.
Division of Labour	Organization of the *subjects* to carry out the workflows of the system. It includes the capabilities of the involved agents, the resources that they control, and their power relationships, among others.
Artifact	Whatever element use to model a MAS.

The definitions of Table 1 give some hints to develop mappings in agent-oriented methodologies. The concepts of *activity*, *subject*, and *objective* have restrictive definitions that can be fulfilled by few concepts in a MAS. So, these concepts are a good starting point to identify the elements of an *activity system*. A *community* can only be constituted by those *subjects* that participate in a common workflow. On the contrary, the concepts of *object*, *outcome*, and *tool* are difficult to identify in a MAS. Their definitions are precise but almost any element used to model a MAS can be them depending on the context. Besides, in the case of the *object* and the *tool*, it could be necessary to know the semantic of the involved *activity* to distinguish what element is transformed into the *outcome*, i.e. the *object*, and what supports the transformation, i.e. the *tool*. The *rules* and the *division of labour* define behavioural patterns for the MAS. Many methodologies define these behaviours assigning roles. The differences between both groups of patterns emerge from the *object*. The *division of labour* determines the organization of the *community* to transform the *object*, while *rules* are more generic. Finally, an *activity system* can be constituted by multiple interconnected *activities* described with different levels of detail. This abstraction must be mainly considered as a useful way of grouping concepts that are not relevant for the analysis in the current level of detail.

4. TRANSLATION PROCESS

The translation process can be seen in Figure 3. This process consists of two main activities: the management of the translation levels and the application of individual mappings.

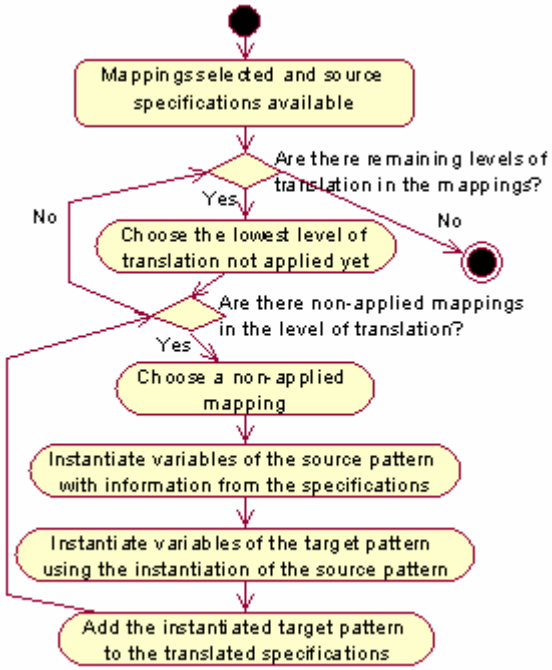


Figure 3. Translation process with mappings.

A translation level groups a set of mappings and has a sorting number. Translation levels are applied according to their sorting number from lowest to highest. Mappings in a translation level are applied sequentially, without any particular order, and can use all the concepts previously translated by other levels. These prerequisites of other concepts for the translation determine the sorting of the levels.

[1] The Rationality Principle states that "Every action taken by an agent aims at achieving one of its goals" [12]

The translation process with the mappings in a specific level consists basically of three steps:

1. Identify the pattern of the source language, i.e. the *match pattern*, over the specifications and instantiate its variables with that information. The process begins looking for groups of elements with the same structure of the *match pattern* in the specifications. When identification happens, the process has to check that the values of the properties in both the pattern and the specification are compatible. If they are compatible, the identification creates a substitution function from the variables in the *match pattern* to the corresponding elements in the specifications.
2. Instantiate the pattern of the target language, i.e. the *target pattern*, with the substitution function obtained from the identification of the *match pattern*.
3. Add the instantiated *target pattern* to the translated specifications. This is the translation of the part of the original specifications that corresponds with the *source pattern* of the mapping.

A remark has to be done about the translation process. Translations between groups of concepts are not always one to one. A given group in the source language may have several corresponding groups in the target language. These groups can correspond to several views (i.e. instances of models) of the original information in the target language; but sometimes they are different possible interpretations of the same information. This ambiguity is inherent to the fact that the modelling primitives and their meaning are different in the source and target languages. However, the negative effects of this potential multiplicity of translations are minimized for the singularity of the original information. That is to say that the multiple interpretations are only information to work in the tool, but they do not exist as part of the specifications, where only the origin of the translation exists.

5. TRANSLATION TOOL

The application of the translation process in a development is automated by a tool called the *Activity Theory Assistant* (ATA). The general purpose of the ATA is to help MAS developers with AT-based techniques. Part of this support is the translation process presented in this paper. Other techniques implemented by the ATA include the support to elicit requirements [7] or to detect AT contradictions in the specifications [5]. The modules required for the translation can be seen in Figure 4.

The knowledge of the assistant about the languages and the correspondences involved in translations comes from XML files, as those seen in Section 3. The information concerning the languages allows the ATA to validate the specifications, checking that their contents are according to the definition of the language. Correspondences specify the translations themselves.

The *graph* package gives an implementation of the primitives of GOPRR [10]. Its components are instantiated for the internal representation of information in the ATA about languages, correspondences, and MAS specifications.

The *wrappers* package provides uniform APIs for the sources of MAS specifications. The *targets* package includes wrappers for the resources related to the input/output about the translation.

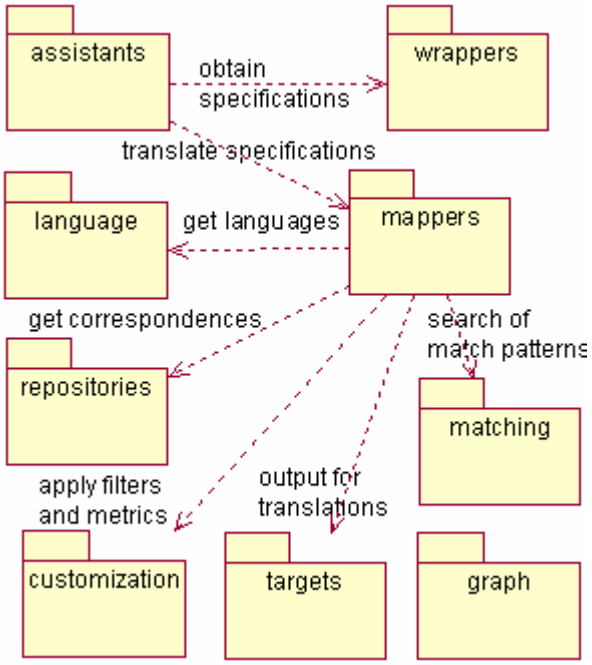


Figure 4. ATA architecture for translation.

Language gives the functionality to deal with the definition of the vocabularies, while the *repositories* package does it for the correspondences. They transform the XML descriptions to the internal structures of the ATA defined by *graph*.

Matching deals with the identification of *source patterns* in the specifications. Its results are used by the *mappers* package to translate specifications according to correspondences.

The *customization* package includes functionality to evaluate the tool. It generates statistics about the use of mappings, generated translations, or elements that the ATA could not translate.

The *assistants* module implements the functionality to coordinate all the previous packages. The core of the ATA belongs to this package.

The integration of the ATA in a specific agent-oriented development environment requires writing the corresponding components for the modules *wrappers* and *targets*. The component for *targets* is optional, since the ATA has default resources for output. As a proof of concept, the ATA is currently embedded as a plug-in of the *INGENIAS Development Kit* (IDK) (http://ingenias.sourceforge.net), which is the modelling and development tool of the INGENIAS methodology [15].

6. CASE STUDY: *eCULTURE*

To illustrate the integration architecture, this section shows a case study based on the *eCulture* system [3]. *eCulture* is a web-based broker of cultural information developed with the Tropos methodology. Tropos [3] describes a whole development process, from early requirements to implementation. This process does not fully address some issues, like the transition from the detailed design to the implementation. This transition is largely a manual process depending on developers' skills to map

concepts from Tropos specifications to the target platform. Even worse, as the process is manual, modifications in the original specifications have to be traced and updated to the final code, with the consequent negative impact on the maintainability.

Given the previous situation for Tropos, developers would prefer using the INGENIAS approach for this transition. INGENIAS [15] mainly bases the coding on templates and the IDK. The templates contain code for the target platform with variables that have to be filled with elements from the specifications. The IDK processes the templates according to the specifications and generates the code for the target platform. Of course, developers have to refine the resulting code, as there is information that neither the specifications nor the templates contain. However, most of the work caused by changes in the specifications is performed by the IDK.

The problem for the previous approach is that the languages of Tropos and INGENIAS are different. Despite the availability of support tools (OME for Tropos and IDK for INGENIAS), none of them can interoperate with the other. A possible solution is the interconnection through the ATA. In order to perform the integration, the ATA needs two mappings: from Tropos to UML-AT and from UML-AT to INGENIAS. The mappings for INGENIAS and Tropos can be found at [6]. A summary of some mappings with INGENIAS appears in Table II. There is a discussion about how to build some of these mappings in the next paragraphs. To keep the explanation simple, this argumentation only considers correspondences from concept to concept, instead of correspondences between graphs, which are more precise semantically (see Section 3 for an explanation about this topic).

The development of the mappings begins with those concepts of the AT that are more restricted. These concepts are *subject*, *activity*, and *objective*.

A *subject* is any element of a MAS that behaves according to the Rationality Principle [12]. INGENIAS has two abstractions that satisfy this definition, *agent* and *role*. Both of them have related goals and tasks to accomplish them.

An *activity* is a practice that satisfies some goal for a *subject*. The concept of *task* in INGENIAS satisfies this definition.

Another possible *activity* in INGENIAS is the *workflow*, which comprehends a set of *tasks* and their related elements like *facts* and *applications*. A *workflow* does not have a satisfaction relationship with goals, but it has a consumption relationship with them. This consumption has to be interpreted as a relation of the *workflows* with the goals that they globally pursue. The reason is that INGENIAS allows including *tasks* without goals. If these *tasks* did not share the goals of their *workflows*, they would not satisfy any purpose, which is impossible according to the Rationality Principle. Given this interpretation, *workflows* are another kind of *activities*. An *interaction* is also an *activity*. It is defined as a sorted set of *tasks* that is executed pursuing a goal, either individual for the participants or global for a group.

An AT *objective* represents a state of the world that agents want to be true. INGENIAS has its own concept of *objective* for these states. Besides, they can also be graphically represented as *mental states* of agents. So, *mental states* are AT *objectives* too.

The concept of *community* is explicitly supported in INGENIAS with *groups* and *organizations*. These *communities* include *agents* and *roles*, which share resources, and the *goals* that they pursue. These *agents* and *roles* usually participate in the same *workflows*. INGENIAS also has implicit *communities*, which are composed by *subjects* participating in the same *workflow* but that do not belong to a common *group* or *organization*.

Table II. Mappings between AT and INGENIAS.

AT	INGENIAS
Activity System	Task, Workflow, Interaction
Activity	
Subject	Agent, Role
Object	Resource, Application, Interaction, Interaction Unit, Fact, Belief, Mental State, Autonomous Entity Query
Outcome	
Objective	Objective, Mental State
Tool	Resource, Application, Task, Workflow, Interaction, Interaction Unit, Fact, Belief, Event, Mental State, Autonomous Entity Query
Community	Group, Organization
Rules	Mental State, Autonomous Entity Query, Interaction, Group, Fact, Belief
Division of Labour	
subject 1 → [decompose] → subject 2	- Organization Model group(subject 1) → [OHasMember] → agent(subject 2)
activity → [accomplished by] → subject	- Agent Model agent(subject) → [WFResponsible] → task(activity)
activity→[transform] → object	- Tasks and Goals Model task(activity)→[GTAffects]→fact(object)
object → [contribute positively] → subject	- Agent Model agent(subject)→ [GTPursues] → *goal 1*[2] agent(subject) → [WFResponsible] → *task 1*
	- Tasks and Goals Model *task 1* → [GTSatisfies] → *goal 1* *task 1* → [GTAffects] → fact(object)

The concepts of *object*, *outcome*, and *tool* are conceptually very close. Any element used by an *activity* is a *tool*. According to the previous translations for *activity*, a *tool* from INGENIAS can be a *resource*, *application*, *task*, *workflow*, *interaction*, *interaction unit*, *fact*, *belief*, *event*, *mental state*, and *autonomous entity query*. *Objects* and *products* can be translated to almost the same concepts as a *tool*. The exceptions are *task*, *workflow*, *interaction*, *interaction unit*, and *event*, as they cannot be transformed or generated by an *activity*.

The *rules* and the *division of labour* can be specified in INGENIAS with *mental states*, *autonomous entity queries*, *interactions*, *groups*, *facts*, and *beliefs*. All of them describe

[2] Elements in italics are new ones introduced by the mappings.

behavioural patterns for the agents and are therefore suitable to establish the norms of the system.

Finally, an *activity system* is a description of one or several *activities* and its context. It can be modelled in INGENIAS with *tasks*, *workflows*, and *interactions*. As it happens with the *communities*, the translation can consider implicit *activity systems* as arbitrary groups of elements in a MAS specification.

This discussion is just an example of the kind of reasoning needed to develop the mappings. More complex patterns (as those about relationships in Table II) would improve the semantic accuracy of the resulting translations.

Once having the mappings, the ATA can perform the translation. For this case study, the actor of Tropos in Figure 5 is translated to INGENIAS. The translation from UML-AT to INGENIAS is going to be discussed with more details; for Tropos, Figure 6 shows the translation to UML-AT using the mappings in [6].

The translation of Figure 6 to UML-AT is, according to mappings for relationships in Table II, as follows. The *match pattern* (i.e. the one on the left column of Table II) is searched over the specifications. For the *decompose* relationship, three matches are found, those corresponding to *Info Broker* with *Results Synthesizer*, *Info Searcher*, and *Area Classifier*. The *solution pattern* (i.e. the one on the right column of Table II) proposes to insert as translations for the *decompose* relations new *OHasMember* relations in the Organization Model. These relations connect the *Info Broker* as a *group* with the other *subjects* as INGENIAS *agents*.

The translation of the relations *accomplished by* and *transform* in Figure 6 is very similar to the one for *decompose*. It uses the second mapping for relationships in Table II.

The mapping for *contribute positively* deserves some remarks. Source and target languages of mappings usually have different semantics. For this reason, it is not always possible to find a direct translation between concepts of both languages. In the case of the *contribute* relations in AT, they can connect *artifacts* (i.e. whatever modelling element of AT), what cannot be represented in INGENIAS. A similar semantic can be obtained with the introduction of two new entities (i.e. *task 1* and *goal 1*) in the solution pattern. These entities are not saved in the specifications; they are just temporarily introduced for the modelling activity.

The result of the application of these three mappings for relationships over the specifications from Figure 6 can be seen in Figure 7. The result is split in different models of an INGENIAS specification.

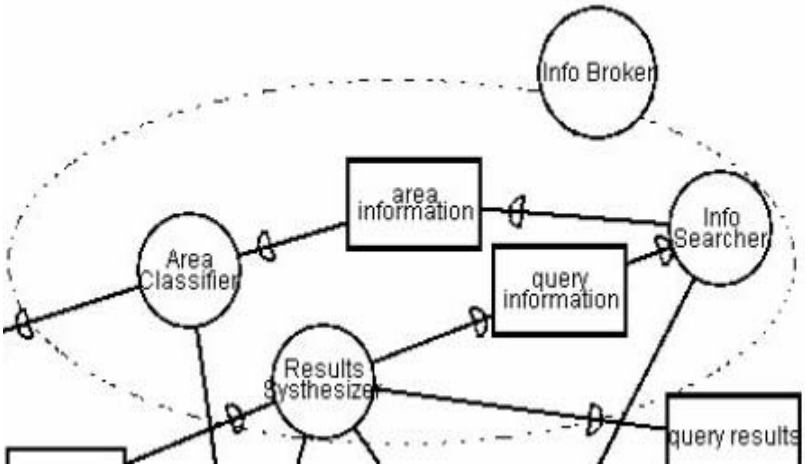


Figure 5 An actor and its components with Tropos.

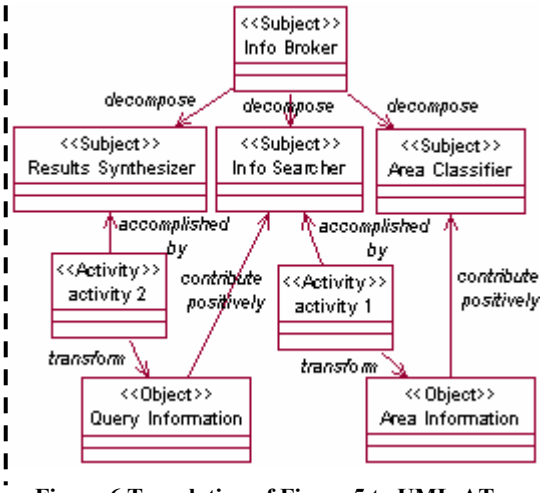


Figure 6 Translation of Figure 5 to UML-AT.

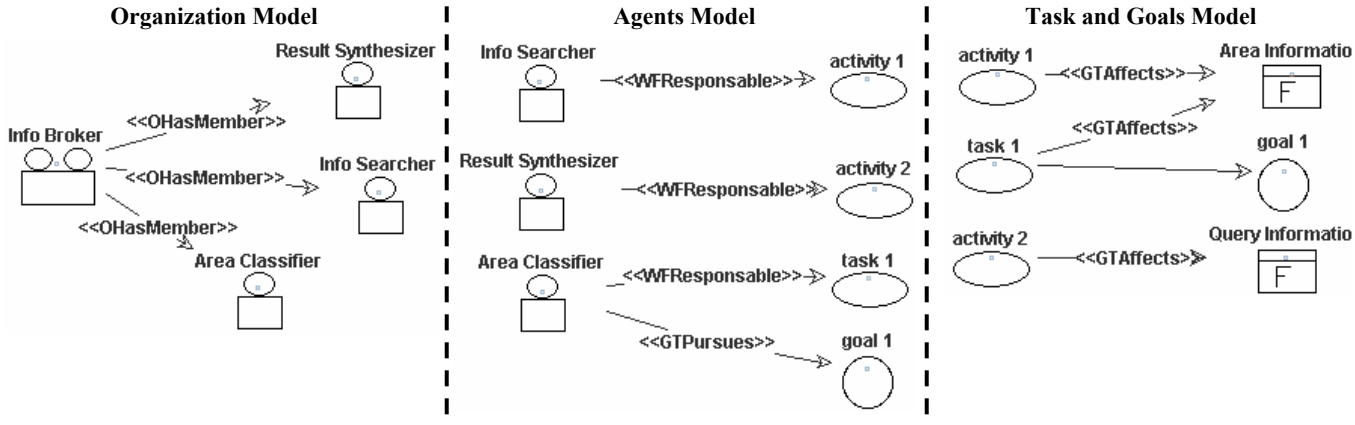


Figure 7. Translation of Figure 6 to INGENIAS.

7. CONCLUSIONS AND FUTURE WORK

This paper presents a framework for the integration of different agent-oriented methods in MAS development. It allows the use of the best-suited method to specify every aspect of a MAS.

The integration is based on a translation process with mappings. A mapping indicates how a structure in a source language can be represented by a structure in another language with the same meaning, or at least a close one. Mappings use as an intermediate language UML-AT. This language grounded in AT is rich enough to support most intentional and social concepts present in agent-oriented modelling languages.

The main advantages of the proposed integration architecture over others are: 1) it is independent of any agent-oriented methodology, as it is based on the neutral language UML-AT; 2) the pattern matching process gives simple explanations about the correspondences applied in the translation; 3) the addition of new methodologies just requires the specification of their mappings; 4) mappings with UML-AT are a suitable resource to discover and analyze missing features in agent-oriented languages; 5) the architecture has automated support tools.

The specification of the mappings is the hardest task of those related with the application of the architecture. It demands a deep analysis of the semantics of the involved concepts. Besides, the possibility of introducing new elements in the *solution pattern* deserves a careful consideration as it should avoid the appearance of unnecessary additional information.

Future work includes two main lines of research. The first one is the creation of mappings for new methodologies, which will enrich UML-AT with new concepts. The second one is the study of agent-oriented methodologies from the AT perspective. This study aims to improve these methods and their integrated use for development, and to get a better understanding of the foundations of the agent paradigm in Social Sciences.

8. ACKNOWLEDGMENTS

This work has been funded by the Spanish Council for Science and Technology under grants TIC2002-04516-C03-03 and TIN2005-08501-C03-01.

9. REFERENCES

[1] Beydoun, G., Gonzalez-Perez, C., Low, G., and Henderson-Sellers, B. Synthesis of a Generic MAS Metamodel. In *Proc. 4th Int. Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'05)* (St. Louis, MO, USA, May 15–16, 2005).

[2] Brazier, F. M. T., Dunin-Keplicz, B. M., Jennings, N. R., and Treur, J. DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework. *International Journal of Cooperative Information Systems 6, 1* (1997*),* 67–94.

[3] Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., and Perini, A. A Knowledge Level Software Engineering Methodology for Agent Oriented Programming. In *Proc. 5th Int. Conf. on Autonomous Agents (AGENTS'01)*, ACM Press, 2001, 648–655.

[4] Dardenne, A, van Lamsweerde, A., and Fickas, S. Goal-directed Requirements Acquisition. *Science of Computer Programming, 20* (1993), 3–50.

[5] Fuentes, R., Gómez-Sanz, and J. J., Pavón, J. Activity Theory for the Analysis and Design of Multi-Agent Systems. *Agent-Oriented Software Engineering IV, 4th International Workshop on Agent-Oriented Software Engineering (AOSE–2003), Lecture Notes in Computer Science, 2935,* Springer-Verlag, 2003, 110–122.

[6] Fuentes-Fernández, R. *Activity Theory for the Development of Multi-agent Systems* (in Spanish: *Teoría de Actividad para el Desarrollo de Sistemas Multi-Agente*). Ph. D. Thesis, Universidad Complutense, Madrid, Spain, 2004.

[7] Fuentes-Fernández, R., Gómez-Sanz, J. J., Pavón, J. Requirements Elicitation for Agent-Based Applications. In *Proc. 6th Int. Workshop on Agent-Oriented Software Engineering (AOSE–2005)* (Utrecht, The Netherlands, July 25–26, 2005).

[8] Iglesias, C. A., Garijo, M., Gonzalez, J. C., and Velasco, J. R. Analysis and design of multiagent systems using MAS-CommonKADS. *Intelligent Agents IV, Lecture Notes in Artificial Intelligence, 1365,* Springer Verlag, 1998, 313–326.

[9] Kinny, D., Georgeff, M., Rao, A. A methodology and modelling technique for systems of BDI agents. *Agents Breaking Away: Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW–96), Lecture Notes in Artificial Intelligence, 1038,* Springer-Verlag, 1996, 56–71.

[10] Lyytinen, K. S., and Rossi, M. METAEDIT+ − A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. *Advances Information System Engineering, 8th International Conference (CAiSE'96), Lecture Notes in Computer Science, 1080,* Springer-Verlag, 1999, 1–21.

[11] Malsch, T. Naming the Unnamable: Socionics or the Sociological Turn of/to Distributed Artificial Intelligence. *Autonomous Agents and Multi-Agent Systems 4 (3)*, 155-186. 2001.

[12] Newell, A. The knowledge level. *Artificial Intelligence, 18*, 87-127, 1982.

[13] OMG. Unified Modelling Language Specification. Version 1.5 [Online]. Available: http://www.omg.org. 2003.

[14] Omicini, A., Ricci, A., Viroli, M., Castelfranchi, C., and Tummolini, L. Coordination artifacts: Environment-based coordination for intelligent agents. In *Proc. 3rd Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS–2004)* (New York, USA, July 19–23, 2004), ACM Press, vol. 1, 286–293.

[15] Pavón J., Gómez-Sanz J., and Fuentes, R. The INGENIAS Methodology and Tools. In Henderson-Sellers, B., and Giorgini, P., editors: *Agent-Oriented Methodologies*. Idea Group Publishing, chapter IX, pp. 236–276, 2005.

[16] Vygotsky, L. S. *Mind and Society*. Harvard University Press, Cambridge, MA, USA, 1978.

Designing Agent Chips

Insu Song
School of Information Technology & Electrical Engineering
The University of Queensland
Brisbane, QLD, 4072, Australia
insu@itee.uq.edu.au

Guido Governatori
School of Information Technology & Electrical Engineering
The University of Queensland
Brisbane, QLD, 4072, Australia
guido@itee.uq.edu.au

ABSTRACT

We outline meta-encoding schemas for compiling nonmonotonic logic theories into Verilog HDL (Hardware Description Language) descriptions. These descriptions can be synthesized into gate level specifications for direct fabrication of silicon chips[1]. The method is applied for designing agent chips incorporating similar features found in the BDI (Belief, Desire, and Intention) and Brooks' subsumption architectures.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Artificial Intelligence

General Terms

Agent architecture, Agent Chips

Keywords

agent programming languages

1. INTRODUCTION

We develop a formal declarative specification language called Layered Non-monotonic Logic (LNL) and meta-encoding schemas to generate two types of executable specifications (general logic programs and synthesizable Verilog descriptions). The reasoning mechanism of each layer is an argumentation system and lower layers are subsumed by higher layers. The semantics of the language is based on the ambiguity blocking Dung-like argumentation system [2]. The language can represent relative certainties among the components of a system and provides behavior based decomposition of a system similarly to the subsumption architecture [1]. The subsumption architecture decomposes a system into (possibly prioritized) parallel behaviors rather than the standard functional decomposition. Benefits of this approach in designing control systems are discussed in [1]. For instance, a typical functional decomposition of a cleaning robot might resemble the sequence:

perception→planning→task selection→motor control.

The decomposition of the same system in terms of behaviors would yield the following set of parallel behaviors:

1:avoid objects < 2:avoid water < 3:clean < 4:wander

where the numbers denote the layers and < denotes increasing levels of competence (i.e., more specific tasks). However, the layers of LNL represent relative confidences between layers such that layer-n is more confident than layer-$(n+1)$. Thus, each behavior will be represented by one or more layers of LNL as shown in Figure 1. The figure also shows that less competent behaviors tend to be related with more confident LNL layers. The reason is because less competent behaviors are usually more reactive or urgent behaviors. LNL supports the behavior based decomposition and conceptual decomposition by *conclusion subsumption* and *rule-subsumption* which will be noted when we give the formal definition of LNL.

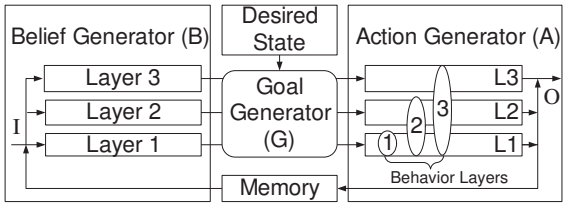


Figure 1: Conceptual view of an agent block.

2. AGENT BLOCK

We briefly discuss an agent architecture called *Agent Block* to which the specification language has been applied. Each agent block is an autonomous system that performs actions in order to achieve a certain desired state. As we can see in Figure 1, an agent block receives signals from its input (I), and reasons about the current state of the world with its background knowledge in the belief generator (B). The background knowledge is decomposed into several layers. The goal generator (G) in the figure then decides what goals to be achieved in order to make the world conform to the desired state. The set of goals then instantiates a particular behavioral specification from a set of plans in the action gen-

[1]Refers to large scale integrated circuits such as ASICs (Application Specific Integrated Circuits) and FPGAs (Field Programmable Gate Arrays).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

erator (A) and produces appropriate actions for the output (O).

Given a desired state (or a set of desired states), the goal generator can be automatically created from the plans in A and the background knowledge in B. In a very simple case, to maintain a desired state d, whenever d is not true, we can activate a set of goals defined in A that are known to be related to achieving d. Suppose the relations between each goal of A and d are described in the background knowledge B. Then, in this case G consists of rules ($\neg d \Rightarrow s$) for each goal $s \in S$ where S is the minimal subset of goals defined in A such that $S \cup B$ entails d.

An agent block specification consists of descriptions of its input, output, memory, desired state, belief generator, goal generator, and action generator. In addition, connection details between these components need to be given for a complete specification of an agent block. However, in this paper, we only outline how to specify and compile the three main parts of an agent block: belief generator, goal generator, and action generator.

Example 1. Let us consider an example specification for the part that controls a vacuum cleaning unit of the cleaning robot. Suppose that the robot performs vacuuming action ($aVac$) if it detects (on sensor sA) that the area is dirty, but it stops the action if it detects some water on sensor sB. That is, we have two parallel behaviors: avoiding water and cleaning. However, avoiding water has higher priority than cleaning. This specification can be represented as a set of defeasible rules decomposed in to two layers as follows (each number of a rule denotes the confidence level of the rule):

B={$sA \Rightarrow_1 dirty$, $sB \Rightarrow_1 wet$}
G={$wet \Rightarrow_1 \neg gclean$, $dirty \Rightarrow_2 gclean$}
A={$gclean \Rightarrow_2 aVac$}

B, G, and A represent the belief, goal, and action generator specifications, respectively. The arrows represent defeasible inferences. For instance, $sA \Rightarrow_1 dirty$ is read as 'if sA is true, then it is usually dirty'. The numbers represent relative confidences between layers such that layer-n conclusions are more confident than layer-$(n+1)$ conclusions. Then, if an area is both dirty and wet, the vacuuming unit will be turned off: $aVac$ is not true. The reason is that since wet is a layer-1 conclusion in B, $\neg gclean$ is a layer-1 conclusion in G. As layer-1 conclusions are more confident than layer-2 conclusions, $\neg gclean$ is also a layer-2 conclusion overriding $gclean$ in G. Thus, in A we cannot conclude $aVac$.

3. LAYERED NON-MONOTONIC LOGIC

We now give a formal definition of the language and its proof theory. From now on, we use *level-n* to mean the degree of confidence of layer-n. A rule r consists of its antecedent (or body) $A(r)$ which is a finite set of literals, an arrow $\Rightarrow_n$ (n is a positive integer denoting a level-n rule), and its consequence (or head) $C(r)$ which is a literal. Level-n rules are more confident than level-$(n+1)$ rules. If l is a literal, $\sim l$ is its complement. An LNL theory is a tuple $T = (R, N)$ where $R = R_1 \cup ... \cup R_n \cup ... \cup R_N$ is a finite set of rules where R_n is a set of level-n rules and N is the number of layers. Then, $\mathbb{R}_n = R_1 \cup ... \cup R_n$ is the set of layer-n rules. That is, layer-n subsumes (includes) layer-$(n-1)$ rules: *rule-subsumption*.

We obtain the meta-program representation of an LNL theory from the meta-program formalization of Defeasible Logic (DL) given in [3] by removing defeaters, strict rules, priority relations, and converting the relationship between strict rules and defeasible rules in DL in to the relationship between layer-$(n-1)$ and layer-n in LNL. Thus, it is also an ambiguity blocking Dung-like argumentation system [2]. The details of the meta-program representation of an LNL theory $T = (R, N)$ and its layers are now described. First, we obtain the *conclusion-meta-program* $\Pi_C(T)$ consisting of the following general logic programs of each layer-n where $1 \leq n \leq N$:

C1. $\mathsf{conclusion}_n(\mathsf{x})$:- $\mathsf{conclusion}_{n-1}(\mathsf{x})$.

C2. $\mathsf{conclusion}_n(\mathsf{x})$:- $\mathsf{supported}_n(\mathsf{x})$, not $\mathsf{supported}(\sim\mathsf{x})$, not $\mathsf{conclusion}_{n-1}(\sim\mathsf{x})$.

where not denotes the negation as failure, $\sim$ maps a literal x to its complement, and $\mathsf{conclusion}_n(\mathsf{q})$ means that q is a layer-n conclusion. C1 represents *conclusion subsumption* because layer-n includes layer-$(n-1)$ conclusions. Next, we obtain the *rule-meta-program* $\Pi_R(T)$ consisting of the following general logic programs for each rule $(q_1, ..., q_m \Rightarrow_i p) \in \mathbb{R}_n$ of each layer-n:

R1. $\mathsf{supported}_n(p)$:- $\mathsf{conclusion}_n(q_1)$,...,$\mathsf{conclusion}_n(q_m)$.

Then, the corresponding general logic program of T is

$$\Pi(T) = \Pi_C(T) \cup \Pi_R(T).$$

The meta-program counter part of the action generator of Example 1 is shown below:

$\mathsf{conclusion}_2(\mathsf{x})$:- $\mathsf{conclusion}_1(\mathsf{x})$.
$\mathsf{conclusion}_1(\mathsf{x})$:- $\mathsf{supported}_1(\mathsf{x})$, not $\mathsf{supported}_1(\sim\mathsf{x})$.
$\mathsf{conclusion}_2(\mathsf{x})$:- $\mathsf{supported}_2(\mathsf{x})$, not $\mathsf{supported}_2(\sim\mathsf{x})$, not $\mathsf{conclusion}_1(\sim\mathsf{x})$.
$\mathsf{supported}_2(aVac)$:- $\mathsf{conclusion}_2(gclean)$.

4. LOGIC PROGRAM REPRESENTATION

We now formally define the meta-encoding schema that translates LNL theories into compact propositional general logic programs. First, we define two literal encoding functions that encode literals in an LNL theory to previously unused positive literals. Let q be a literal and n a positive integer. Then, these functions are defined below:

$$Supp(q,n) = \begin{cases} p_n^{+s} & \text{if } q \text{ is a positive literal } p. \\ p_n^{-s} & \text{if } q \text{ is a negative literal } \neg p. \end{cases}$$

$$Con(q,n) = \begin{cases} p_n^{+} & \text{if } q \text{ is a positive literal } p. \\ p_n^{-} & \text{if } q \text{ is a negative literal } \neg p. \end{cases}$$

$Supp(q,n)$ denotes a support of q at layer-n and $Con(q,n)$ denotes a conclusion q at layer-n: $Supp(q,n)$ corresponds to $\mathsf{supported}_n(\mathsf{q})$; $Con(q,n)$ corresponds to $\mathsf{conclusion}_n(\mathsf{q})$. Let $ConA(A,n)$ be a set of new positive literals obtained from a set A of literals by replacing each literal $q \in A$ by $Con(q,n)$: $ConA(A,n) = \{Con(q,n) | q \in A\}$. With these functions, we now define the meta-encoding schema.

Let $T = (R, N)$ be an LNL theory and $\Pi(T)$ the corresponding meta-program. Let H_T be the Herbrand universe of $\Pi(T)$ which is the set of all possible grounded terms in $\Pi(T)$. Then, the Herbrand base $H_T(T)$ of $\Pi(T)$ is obtained by applying all possible consistent substitutions of

ground terms in H_T with variables in $\Pi(T)$. Thus, the meta-encoded Herbrand base (denoted as $P(T)$) of $\Pi(T)$ is obtained according to the following guidelines for each layer-n $(1 \leq n \leq N)$:

P1: For each $q \in H_T$, add $Con(q,n) \leftarrow Con(q,n-1)$.

P2: For each $q \in H_T$, add

$Con(q,n) \leftarrow Supp(q,n), not\ Supp(\sim q,n), not\ Con(\sim q,n-1)$

P3: For each $r \in \mathbb{R}_n$, add $Supp(C(r),n) \leftarrow ConA(A(r),n)$.

For most of LNL theories, this direct translation of T results in a lot of redundant rules that will be never used for generating conclusions. For instance, we can safely remove the rules produced by P2 that will be never supported. Thus, we replace 'For each $q \in H_T$' with 'For each $q \in SL_n$' in P2 where SL_n is the set of all layer-n literals that might be supported in $P(T)$. SL_n can be obtained as follows:

$$SL_n = \{C(r) | r \in \mathbb{R}_n\}$$

Let $P2(T)$ be the set of rules introduced by P2 in $P(T)$. Then, we can also reduce the number rules in P1 by replacing 'For each $q \in H_T$' with 'For each $q \in CL_n$' where CL_n is the set of all conclusion literals in layer-n of T. This is obtained as follows:

$$CL_0 = \emptyset$$
$$CL_n = CL_{n-1} \cup \{q | Con(q,n-1) \in P2(T)\}$$

Let us consider an LAS theory $T = (R,N)$ containing x unique rules in each layer. The total number of rules is $X = xN$. Then, the number of rules and facts created by the guidelines is bounded by the following equation:

$$|P(T)| \leq X(0.5 + 1.5N)$$

That is, the size of $P(T)$ is linear to the size of T. Particulary, if $N = 1$, $|P(T)| \leq 2|T|$.

The general logic program corresponding to the action generator of Example 1 is shown below:

$aVac_2^+$:- $aVac_2^{+s}, not\ aVac_2^{-s}, not\ aVac_1^-$.
$aVac_2^{+s}$:- $gclean_2^+$.

5. VERILOG HDL DESCRIPTION

Verilog is one of widely used HDL languages. An HDL is a language for formal description of electronic circuits. Most of VLSI (Very-large-scale integrated) circuit design tools can import Verilog files to generate a netlist which can be directly used to fabricate ICs. A netlist is a list of logic gates and their interconnections which make up a circuit.

The meta-encoding of an LNL theory into a Verilog module is similar to the definition of $P(T)$ which is given in the previous section. The differences are that all the used literals (referred as 'signals' in Verilog) in rules must be declared separately and signals have binary values: 1 and 0. We use value 1 to represent that a literal (a signal) is 'true' and 0 to represent 'unknown'. All defined signals are assumed to have value 0 by default but can be changed to value 1 if it is derived by a logic gate with the output value of 1. The exact implementation of this property is device specific, and thus it will be ignored in this paper.

Let $T = (R,N)$ be an LNL theory. The corresponding Verilog description $V(T)$ is obtained from T by adding the Verilog statements according to the following guidelines for each layer-n $(1 \leq n \leq N)$:

V1: For each $q \in CL_n$, add
wor $Con(q,n)$;
assign $Con(q,n) = Con(q,n-1)$;

V2: For each $q \in SL_n$, add
wor $Supp(q,n), Supp(\sim q,n), Con(q,n)$;
assign $Con(q,n) = Supp(q,n) \wedge \sim Supp(\sim q,n) \wedge \sim Con(\sim q,n-1)$;

V3: For each $r \in \mathbb{R}_n$, add
wor $Supp(C(r),n), Supp(\sim C(r),n), Con(C(r),n)$;
assign $Supp(C(r),n) = \bigwedge_{q \in A(r)} Con(C(r),n)$;

The Verilog module description of the action generator of Example 1 is shown below:

```
module A (input gclean2+, output aVac2+);
    wor aVac2+, aVac2+s, aVac2-s, aVac1-, gclean2+;
    assign aVac2+ = aVac2+s & ~aVac2-s & ~aVac1-;
    assign aVac2+s = gclean2+;
end module
```

In addition to the above description, we also need to add some device specific codes to set the default values (0) of the signals and port mapping codes between the components (B, A, G, and the memory) of the agent block. The generated Verilog descriptions can be directly compiled to a netlist using a freely available HDL synthesizer Xilinx^{TM} ISE Webpack. In the module definition, both $aVac_2^{-s}$ and $aVac_1^-$ are not necessary as they will never be true. Moreover, in practice many of subsumed rules are redundant because only few of the literals in layer-$(n-1)$ are referred in layer-n. Thus, if such redundant codes are removed, the actual size of a Verilog description of an LNL theory can be made close to the size of the original theory independent of the number of layers.

6. CONCLUSION

We have devised a formal translation method which compiles high level (non-monotonic logic based) specifications of an agent into synthesizable Verilog HDL descriptions. In contrast to current agent systems, an agent chip is realized as a concurrent computational structure on a silicon chip. This makes agent chips very robust and reactive. Since the specification language is a consistent instantiation of an argumentation system, it also has a well defined semantics which is critical for system verification and maintenance. Thus, agent chips can be readily used for mission critical applications such as distributed plant controllers, mobile robots, and patient caring systems.

7. REFERENCES

[1] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.

[2] G. Governatori, M. J. Maher, G. Antoniou, and D. Billington. Argumentation semantics for defeasible logics. *Journal of Logic and Computation*, 14(5):675–702, 2004.

[3] M. J. Maher and G. Governatori. A semantic decomposition of defeasible logics. In *AAAI '99*, pages 299–305, 1999.

Multiagent Systems as Software Architecture

Another Perspective on Software Engineering with Multiagent Systems

Danny Weyns
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven, Belgium
danny@cs.kuleuven.be

Tom Holvoet
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven, Belgium
tom@cs.kuleuven.be

Kurt Schelfthout
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven, Belgium
kurts@cs.kuleuven.be

ABSTRACT

The trend in agent-oriented software engineering is to consider multiagent systems (MASs) as a *radically new* way of engineering software. This position isolates agent-oriented software engineering from mainstream software engineering and could be one important reason why MASs are not widely adopted in industry yet.

In this paper, we present another perspective on software engineering with MASs. We put forward MASs as *software architecture*. We give an overview of a reference architecture for situated MAS. This reference architecture extracts and generalizes common functions and structures from various applications we have studied and built. The reference architecture provides a blueprint for architectural design of MAS applications that share the come base of the systems it is derived from. Considering MASs essentially as software architecture paves the way to integration with mainstream software engineering.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; D.2.11 [**Software Engineering**]: Software Architectures

General Terms

Design

1. INTRODUCTION

Researchers and practitioners in agent-oriented software engineering generally consider MASs as a *radically new* way of engineering software. Here is a list of recent quotes from literature:

- There is a fundamental mismatch between the concepts used by mainstream software engineering and the agent-oriented view. Existing software development techniques are unsuitable to realize the potential of agents as a software engineering paradigm. [15]
- Whether agent-oriented approaches catch on as a software engineering paradigm [will depend on] the degree to which agents represent a radical departure from current software engineering thinking. [5]
- We are on the edge of a revolutionary shift of paradigm, pioneered by the multiagent systems community, and likely to change our very attitudes in software modelling and engineering. [17]
- Agent-based computing can be considered as a new general-purpose paradigm for software development, which tends to radically influence the way a software system is conceived. [16]

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

This vision has led to the development of numerous MAS methodologies. Some of the proposed methodologies adopt techniques and practices from mainstream software engineering such as object-oriented techniques, e.g. Prometheus [6]; the Rational Unified Process, e.g. Mase [14]; or the Unified Modeling Language, e.g. Adelfe [3]. However, nearly all of these methodologies start from the idea that MAS provides a new way to engineer software systems. The position of being a radically new paradigm for software development isolates agent-oriented software engineering from mainstream software engineering. This could well be one important reason why MASs are not widely adopted in industry yet.

This paper presents another perspective on software engineering with MASs. We put forward MASs as *software architecture*. Considering MASs as software architecture gives MAS a clear position in the software development process, and paves the way to integration with mainstream software engineering.

Overview. Section 2 explains our perspective of MAS as software architecture. In section 3, we give a brief overview of architectural design that provides a context to apply MAS as software architecture in software engineering. Section 4 gives an overview of a reference architecture for situated MAS that architects can use for the design of concrete software architectures. Finally, we conclude in section 5.

2. MULTIAGENT SYSTEMS AS SOFTWARE ARCHITECTURE

Instead of considering MASs as a radically new approach for software development, we aim to give MASs a position in a general software engineering process. Our perspective on the essential purpose of MASs is as follows:

Multiagent systems provide an approach to solve a software problem by decomposing the system into a number of autonomous entities embedded in an environment in order to achieve the functional and quality requirements of the system.

This perspective states that MASs are an approach to *solve* a software problem. In particular, a MAS is a specific *decomposition of interacting elements* of the system intended to achieve the *requirements* of the system. This is exactly what *software architecture* is about. [1] defines software architecture as: "the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them." Software elements provide the functionality of the system, while the required quality attributes are primarily achieved through the structures of the software architecture. As such, MASs are in essence a family—yet a large family—of software architectures. The elements of a MAS software architecture are agents, resources, services, environment, etc. Relationships among the elements are very diverse, ranging from indirect interaction through virtual pheromone trails to complex negotiation protocols. In short, MASs are a very rich family of architectural approaches with specific characteristics, useful for a diversity of chal-

lenging application domains. Based on the problem analysis that yields the functional and quality attribute requirements of the system, the architect may or may not choose for a MAS-based solution. Quality attribute requirements such as robustness, flexibility or openness may be arguments to choose for a MAS software architecture. As such, we consider MASs as one valuable family of approaches to solve software problems in a large spectrum of possible ways to solve problems.

3. ARCHITECTURAL DESIGN

We now give a brief overview of architectural design that provides a context to apply MASs as software architecture. In the software development life cycle, architectural design iterates with requirements analysis on the one hand, and with the detailed design and development of the system on the other hand. Architectural design includes the design, the documentation, and the evaluation of the software architecture.

Design. Architectural design requires a systematic approach to develop a software architecture that meets the system requirements. In our research, we use techniques from the Attribute Driven Design (ADD [2]) method to design the architecture for a software system. The ADD method is a recursive decomposition method that is based on understanding how to achieve quality goals through proven architectural approaches. One common architectural approach are architectural patters [7]. An architectural pattern is a description of architectural elements and their relationships that has proven to be useful for achieving particular qualities. Examples of architectural patterns are pipe–and–filter or blackboard. We have developed a reference architecture for situated multiagent systems as a reusable architectural approach. This reference architecture integrates a set of architectural patterns that have proven their value in various MAS applications we have studied and built. The reference architecture provides an asset base the architect can draw from to select suitable architectural solutions.

Documentation. A software architecture is described by *views* [1]. A view describes the architecture of a software system from a particular perspective. The main views are: the module view that documents a system's principal units of implementation; the component-and-connector view that documents the system's units of execution; and the deployment view that documents the relationships between a system's software and its development and execution environment. Documenting a software architecture comes down to documenting the relevant views of the software architecture for the application at hand.

Evaluation. A software architecture is the foundation of a software system, it represents a system's earliest set of design decisions. Due to its large impact on the development of the system, it is important to verify the architecture as soon as possible. Modifications in early stages of the design are cheap and easy to carry out. In our research, we use the Architectural Tradeoff Analysis Method (ATAM [8]). The ATAM incites the stakeholders to articulate and prioritize specific quality goals; it forces the architect to provide a clear explanation and documentation of the software architecture; and especially it uncovers problems with the architecture that can be used to improve the quality of the software architecture in an early stage of the development cycle.

4. REFERENCE ARCHITECTURE FOR SITUATED MULTIAGENT SYSTEMS

The reference architecture we present in this section generalizes and extracts common functions and structures from various applications we have studied and built, including the Packet-World, a P2P file sharing system, a number of basic robot applications, and an simulator for Automatic Guided Vehicle systems. Besides these basic applications, the reference architecture considerably draws from experiences with an industrial logistic transportation system for warehouses [11]. The reference architecture provides a blueprint for architectural design of MAS applications that share the come base of the systems it is derived from.

Fig. 1 shows an overview of the reference architecture. The architecture integrates two primary abstractions: *agents* and the *environment*.

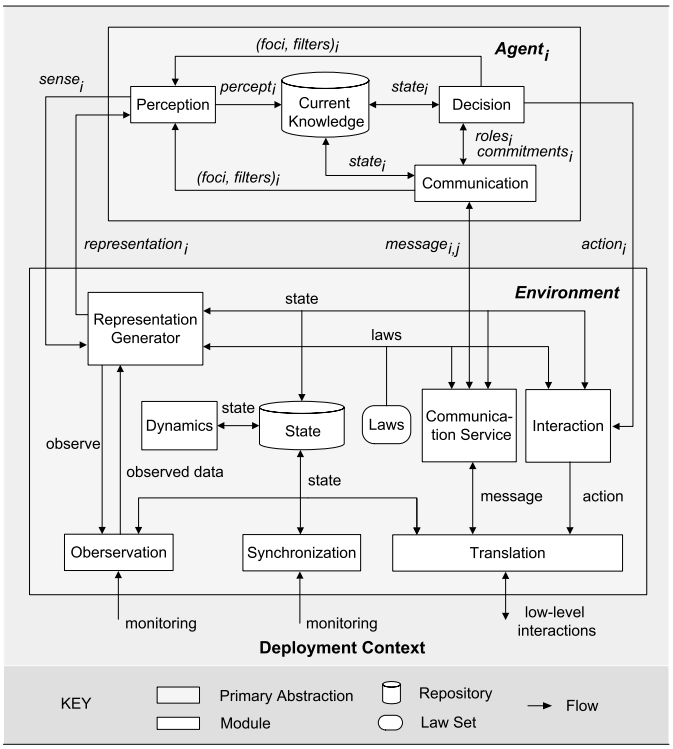


Figure 1: High-level overview of the reference architecture for situated multiagent systems

Agents. The agent architecture models perception, decision making and communication as separate modules of the agent. The *Perception* module maps a local representation of the state of the environment to a percept for the agent. We developed a model for *selective perception* that enables an agent to direct its perception at the most relevant aspects in the environment according to its current task. To sense its environment, the agent selects a set of *foci*. Foci allows the agent to sense the environment only for specific types of information. Sensing results in a representation of the agent's surrounding that can be interpret by the agent producing a percept. Finally, the percept is filtered by a set of selected *filters*, restricting the perceived data according to context relevant selection criteria. [13] zooms in on the perception module and discusses the different submodules in detail. The *Current Knowledge* module integrates percepts to update the knowledge of the agent.

The *Decision* module is responsible for action selection. We developed the decision module as a free-flow architecture, for details, we refer to [9]. Since existing free-flow architectures lack explicit support for social behavior, we introduced the concepts of a *role* and a *situated commitment*. A role covers a logical functionality of the agent, while a situated commitment allows an agent to adjust its behavior towards the role in its commitment. An agent can commit to itself, e.g. when it has to fulfill a vital task. However, in a collaboration, agents commit to one another via communication. Roles and situated commitments are building blocks for explicit collective behavior. The action selected by the decision module is passed to the environment for execution.

Communication enables agents to exchange information, and set up collaborations reflected in mutual situated commitments. We developed a communication module that processes incoming messages and produces outgoing messages according to well-defined communication protocols. The communication module (1) interprets messages and reacts to them according to the applicable protocol, and (2) initiates and

continues conversations when the conditions imposed by the applicable protocol are satisfied. Messages are passed to the message delivering system of the environment. The communication module is discussed in detail in [12].

Environment. The environment offers functionality to the agents for perception, communication and action, and it provides an interface to the *deployment context*. With deployment context, we refer to the given hardware and software and external resources with which the MAS interacts such as sensors and actuators, a printer, a network, a database, or a webservice. We now explain the different modules of the environment, for a detailed discussion of the environment model see [10].

The *State* repository represents the actual state of the environment. The environment state typically includes an abstraction of the deployment context possibly extended with other state related to the MAS environment. An example of state related to the deployment context is an abstract representation of a network topology. An example of additional state is the representation of digital pheromones that overlays the network.

The *Synchronization* module monitors specific parts of the deployment context and keeps the corresponding representation in the state repository up to date. An example is a synchronization module that monitors the topology of a dynamic network and maintains the representation of the network structure in the state repository.

The *Dynamics* module maintains dynamics in the environment that happens independent of the agents or the deployment context. A typical example is the the evaporation of a digital pheromone.

Laws represent application-specific constraints on agents' perception, interactions, and communication. We discuss examples of different types of laws below.

The *Percept Generator* generates percepts for the agents. In general, agents can perceive state of the environment, or observe elements in the deployment context. In this latter case, the percept generator requests the *Observation* module to retrieve the required data form the deployment context. Perception is subject to laws that provide a means to constrain perception. For example, for reasons of efficiency a designer can introduce default limits for perception in order to restrain the amount of information that has to be processed, or to limit the occupied bandwidth. [13] discusses the perception generator in detail.

The *Interaction* module deals with agents' actions. Actions can be divided in two classes: actions that may result in a modification of state of the environment, and actions that may result in the modification of elements of the deployment context. An example of the former is an agent that drops a digital pheromone in the environment. An example of the latter is an agent that writes data in an external data base. Actions are subject to interaction laws. For example, when several agents aim to access an external resource, an interaction law may impose a policy on the access of that resource. Actions related to the deployment context are passed to the *Translation* module that converts the high-level actions of agents into low-level interactions in the deployment context.

The *Communication Service* module collects messages and delivers messages to the appropriate agents. Message delivering can be subject to communication laws. Communication laws can just regulate the message stream, or they can impose application-specific regulations on exchanged messages. An example of this latter are a set of laws that impose agents to follow the prescribed steps of a particular protocol. The translation module converts the high-level message descriptions into low-level communication primitives of the deployment context.

Industrial Application. We have used the reference architecture for situated MASs for the architectural design of an industrial system for logistics services in a tobacco warehouse. This real-world application uses automatic guided vehicles to transport loads in the warehouse [11]. For the documentation of the software architecture of this application and a detailed report of the ATAM evaluation we refer to [4].

5. CONCLUSION

In this paper, we presented another perspective on software engineering with MASs. We put forward MASs as software architecture. Looking upon MASs as software architecture does not dilute MAS as a software engineering paradigm, on the contrary, it gives MAS a clear and prominent role in the software development process, paving the way to integration with mainstream software engineering.

We gave an overview of a reference architecture for situated MASs. The reference architecture integrates a set of patterns that architects can use during architectural design of concrete MAS applications.

Intensive experience with applying the reference architecture in a complex industrial application has convinced us that the MAS community would benefit a great deal from allocating a correct place for MAS in mainstream software engineering.

6. REFERENCES

[1] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison Wesley, April 2003.

[2] L. Bass, M. Klein, and F. Bachmann. Quality Attribute Design Primitives and the Attribute Driven Design Method. *4th International Workshop on Product Family Engineering*, 2001.

[3] C. Bernon, M.-P. Gleizes, S. Peyruqueou, and G. Picard. Adelfe: A Methodology for Adaptive Multiagent Systems Engineering. *Volume 2577 of Lecture Notes in Computer Science*, 2002.

[4] N. Boucke, T. Holvoet, T. Lefever, R. Sempels, K. Schelfthout, D. Weyns, and J. Wielemans. Applying the ATAM to an industrial multi-agent system application. In *CW-431, Technical Report, K.U.Leuven, Belgium*.

[5] N. R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4), 2001.

[6] L. Padgham and M. Winikoff. Prometheus: A Methodology for Developing Intelligent Agents. *Volume 2585 of Lecture Notes in Computer Science*, 2003.

[7] M. Shaw and D. Garlan. *Software architecture: perspectives on an emerging discipline*. Prentice-Hall, 1996.

[8] Software Engineering Institute. Carnegie Mellon University. *http://www.sei.cmu.edu/*.

[9] E. Steegmans, D. Weyns, T. Holvoet, and Y. Berbers. A Design Process for Adaptive Behavior of Situated Agents. In *Volume 3382 of Lecture Notes in Computer Science*, 2005.

[10] D. Weyns, A. Omicini, and J. Odell. Environment, First-Order Abstraction in Multiagent Systems. *Journal of Autonomous Agents and Multiagent Systems*, 2006 (to appear).

[11] D. Weyns, K. Schelfthout, T. Holvoet, and T. Lefever. Decentralized control of E'GV transportation systems. In *4th AAMAS Conference, Industry Track*, Utrecht, 2005.

[12] D. Weyns, E. Steegmans, and T. Holvoet. Protocol-Based Communication for Situated Multiagent Systems. *3th AAMAS Conference, New York*, 2004.

[13] D. Weyns, E. Steegmans, and T. Holvoet. Towards Active Perception in Situated Multi-Agent Systems. *Journal of Applied Artificial Intelligence, 18(8-9)*, 2004.

[14] M. Wood, S. A. DeLoach, and C. Sparkman. Multiagent Systems Engineering. *Software Engineering and Knowledge Engineering*, 11(3), 2001.

[15] M. Wooldridge, N. Jennings, and D. Kinny. The Gaia Methodology for Agent-Oriented Analysis and Design. *Autonomous Agents and Multi-Agent Systems*, 3(3), 2000.

[16] F. Zambonelli and A. Omicini. Challenges and Research Directions in Agent-Oriented Software Engineering. *Journal of Autonomous Agents and Multiagent Systems*, 9(3), 2003.

[17] F. Zambonelli and V. Parunak. *From Design to Intention: Signs of a Revolution*. 1st AAMAS Conference, Bologna, 2002.

Automated Analysis and Verification of Agent Behavior

Tibor Bosse
Vrije Universiteit Amsterdam
Department of Artificial Intelligence
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands
tbosse@cs.vu.nl

Dung N. Lam, K. Suzanne Barber
The University of Texas at Austin
Laboratory for Intelligent
Processes and Systems
1 University Station C5000
Austin, TX 78712-0240
{dnlam, barber}@lips.utexas.edu

ABSTRACT
Comprehending and analyzing agent behavior is an arduous task due to complexities in agent systems and sophistication of agent behaviors, in addition to the common difficulties with any complex software system. This paper presents an integrated approach for the analysis and verification of behaviors of agent-based systems. The approach is a result of collaboration between the Tracer Tool and the TTL Checker, which together automate the analysis and verification of agents in an implemented agent system with the aim of aiding the user in redesigning, debugging, and maintaining the software system. The Tracer Tool ensures that the user's comprehension of the system behavior is accurate and provides explanations of anomalous behavior, which can be detected as a failed behavioral property by the TTL Checker. The integrated approach has been applied successfully in a case study in the domain of Unmanned Aerial Vehicles.

Categories and Subject Descriptors
D.2.4 [**Software/Program Verification**]: *formal methods*; D.2.5 [**Testing and Debugging**]: *debugging aids, diagnostics, monitors, testing tools, tracing*.

General Terms
Documentation, Experimentation, Human Factors, Verification.

Keywords
Software comprehension, agent-oriented software engineering, reverse engineering, debugging, maintenance, tracer, verification of agent systems.

1. INTRODUCTION
Software comprehension is critical in debugging and maintaining complex software systems, though it has lacked appropriate attention in agent software systems. Most existing tools aimed at comprehending agent-based software can be classified either as reverse engineering tools or as model checking tools. According to [2], *reverse engineering* is "the process of analyzing a subject system to create representations of the system at a higher level of abstraction". Following this definition, a number of reverse engineering tools have been developed to assist the designer in analyzing and perusing the source code of a software system, e.g., [5]. In addition, various tools have been developed to perform *model checking* (i.e., to prove that the formal specification of a system satisfies certain behavioral properties) [3]. Nevertheless, despite the above tools to aid the user in understanding software behaviors, software comprehension has remained a time-consuming and mostly manual process. The obstacles faced by designers, developers, and end-users (who will be referred to simply as 'users') during software comprehension include (1) software complexity and sophistication, (2) the translation gap between low-level implementation structures and high-level design concepts, and (3) the large amount of time and effort demanded to gather a comprehensive picture of the software.

This paper demonstrates the Tracing Framework, which is supported by the Tracer Tool by [7] and the TTL Checker by [1], to reduce the effort in comprehending agent-based software systems. The approach to contend with the difficulties of software comprehension is to automate as many of the tasks involved in software comprehension as possible by presenting a visualization of what the agents are doing in terms of high-level agent concepts (e.g., beliefs, intentions, actions, and events) based on logs of system execution and comparing the visualization to the user's comprehension of the implemented agent system. The user's comprehension is explicitly modeled in order for the Tracer Tool to provide feedback as to the accuracy of the user's comprehension.

Additionally, the TTL Checker provides automated checking of agent or system behavioral properties against logs of system executions. This approach is similar to model-checking, except the input is a set of observations from the systems execution, rather than a formal model of the system. This is in accordance with Edmonds' emphasis on the need for empirical analysis of agent behavior, such as scenario-based analysis and field testing [4]. A user can discover behavioral anomalies from Tracer's graphical interface or be alerted by the TTL Checker. To aid the user in tracking down a problem, Tracer can generate explanations that reveal, for example, why an agent performed a certain action.

2. TRACER TOOL AND TTL CHECKER
Motivated by the difficulty of comprehending complex agent systems, the objective of the Tracing Framework [7] is to aid the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

user in understanding what is happening in the agent system and why such things are happening in terms of high-level agent concepts that are familiar to software designers, developers, and end-users. The Tracing Framework is established by the ***Tracer Tool***, which provides automation to some of the arduous, manual tasks involved for comprehension (see [7] for details):

Building Background Knowledge. In order to help the user build and maintain a comprehension of the system, the Tracer Tool requires a model of what the user understands. This model is called the background knowledge and is represented as a relational graph, where each node represents an agent concept and an edge represents a relation (e.g., causal or temporal relation) between two agent concepts. Given an empty or incomplete background knowledge, the Tracer Tool can offer suggestions about possible relations between agent concepts based on observations from the system's execution, thus building up the background knowledge.

Logging. Agent concepts (i.e., beliefs, goals, intentions, action, events, and communication messages) are captured as run-time observations using a logging mechanism.

Interpretation. The observations are a manifestation of agent behavior; actual agent behavior must be interpreted with respect to the user's knowledge of expected behavior. The Tracer Tool can create an interpretation of the observations using the modeled background knowledge. The interpretation is visualized as a relational graph, similar to the background knowledge.

Explanation. When the behavior of the implementation changes due to maintenance or redesign activities, the Tracer Tool provides suggestions to update the background knowledge. Since the user must review and accept all modifications to the background knowledge, the user's comprehension is also kept updated with respect to the current implementation. To help the user decide whether a suggestion is valid or to further clarify causal factors of a particular observation, explanations can be generated by the Tracer Tool.

Verification. Given the interpretation of a system execution, patterns of agent behavior can be readily discovered. As a result, anomalous behavior can also be detected as any observation that does not conform to a behavior pattern. Currently, anomaly detection is a manual task that involves discovering behavioral patterns in the interpretation, a task at which humans are extremely good. The Tracer Tool provides a capability to verify the expected behavior (represented as causal graphs) against the actual behavior of the implemented system.

For more complex behavioral properties, the ***TTL Checker*** by [1] provides automated verification of such properties using the observations collected by the Tracer Tool. The TTL Checker is a tool that enables the automated verification of complex dynamic properties against traces. It is based on the predicate logical Temporal Trace Language (TTL) [1]. In this language, to describe dynamic properties of complex agent systems, explicit reference is made to *states*, *time points* and *traces*. Dynamic properties can be formulated that relate a state at one point in time to a state at another point in time. A simple example is the following informally stated dynamic property for belief creation:

> if the agent observes at t1 that it is raining,
> then the agent will believe that it is raining

In formalized (TTL) form this property looks as follows:

$$\forall t1 \ [\ state(\gamma, t1) \models observes(A, itsraining) \Rightarrow$$
$$\exists t2 \geq t1 \ \ state(\gamma, t2) \models belief(A, itsraining)\ ]$$

To enable the automated verification of dynamic properties specified in TTL against formal traces (or logs), the TTL Checker takes a dynamic property and one or more formal traces as input, and checks whether the dynamic property holds for the traces.

3. INTEGRATED ANALYSIS APPROACH

Combining the advantages of the Tracer Tool and TTL Checker, an integrated approach is presented for the analysis and comprehension of agent-based software. This section describes the constituent steps in this approach.

1. ***Create a number of traces (or logs) of executions of a real agent system, in terms of agent concepts such as beliefs, desires, and actions***. This step is done using the Tracer Tool (in particular, using the logging mechanism mentioned in Section 2). Normally, the logs are represented in XML format. To facilitate user understanding of the logs, a translation program has been written that translates the logs to traces in TTL format. As a result, the traces can also be loaded into a specific TTL visualization tool.

2. ***Formalize a number of relevant dynamic properties in TTL***. This step is done using a graphical editor for TTL (see [1] for details). Two types of dynamic properties can be distinguished: *generic* dynamic properties and *domain-specific* dynamic properties. Generic dynamic properties are those that are independent of the domain. A simple example is 'Communication Correctness', i.e., "all messages sent from agent a1 to agent a2 are received by agent a2". In TTL, this property looks as follows:

$$\forall \gamma 1{:}TRACES \ \forall t{:}TIME \ \forall a1,a2{:}AGENT \ \forall m{:}MESSAGE$$
$$[\ [\ state(\gamma 1,t) \models sending\_message\_to(a1, a2, m) \wedge a1 \neq a2\ ]$$
$$\Rightarrow \exists \gamma 2{:}TRACES \ \ [\ state(\gamma 2,t) \models receiving\_message\_from(a2, a1, m)\ ]\ ]$$

The TTL checker contains a library of such generic dynamic properties, which can be re-used in any domain. Domain-specific dynamic properties have to be specified by hand for every new domain. Due to the expressiveness of the TTL language, complex behavioral properties can be specified, e.g., relating events at many different time points and different traces to each other.

3. ***Automatically check the dynamic properties created in step 2 against the logs created in step 1***. This step is done using the TTL Checker (see [1]). As a result of this step, the user knows exactly which properties hold and which do not hold for the traces. Moreover, in case a property does not hold, the checker can provide detailed information about the position in the trace where the property fails. Usually this information has the form of a counter example. For example, if the aforementioned property 'Communication Correctness' fails, then the TTL Checker shows an example of a message that is sent from agent a1 to a2, but it is not received by a2.

4. ***Explain failed properties***. This step is done using the Tracer Tool (in particular, the explanation mechanism described in Section 2). Given a counter example from the TTL Checker due to a failed property, the user can examine the exact observations and its relation to other observations in the Tracer Tool's visualization. Observation details provide further evidence to track down the source of the problem, such as the type of communication message, who sent it, and why it was sent.

4. SUMMARY

Software comprehension is an essential issue during the development, maintenance, and debugging of complex agent systems. Nevertheless, existing techniques (such as reverse engineering [2] and model-checking [3]) face a number of difficulties, in particular (1) the high complexity and sophistication of the software, (2) the translation gap between low-level implementation structures and high-level design concepts, and (3) the large amount of time and effort demanded to gather a comprehensive picture of the software.

To deal with these problems, this paper presents a novel approach to analyze the behavior of agent systems. The approach is the result of the collaboration between the Tracer Tool for comprehension of agent software [7] and the TTL Checker for automated checking of dynamic properties of agent systems [1]. The integrated approach comprises four important elements, i.e. (1) logging of agent system executions in terms of high-level design concepts, (2) formalization of desired dynamic properties, (3) automated verification of dynamic properties against logs, and (4) computer-aided explanation of unexpected properties.

To test the proposed approach, it has been applied in the domain of Unmanned Aerial Vehicles (UAV), see [8]. First, using the Tracer Tool, a number of logs of the UAV system were generated in terms of agent concepts. Next, generic and domain-specific dynamic properties were specified in TTL, and checked against the generated logs. Finally, several of the unexpected properties were explained using the Tracer Tool. Using this approach, the software developers succeeded in locating some undesired behavior, which subsequently allowed them to investigate the causes for such behavior. Thus, the approach turned out useful in the comprehension and analysis of agent software.

Concerning related work, existing research addresses the different elements of the presented approach separately. For example, in [6] several tools are suggested for the collection of trace files (consisting of events and corresponding time parameters) in a distributed system. To examine the collected traces, techniques for querying distributed databases are proposed. This approach is similar to our approach in the sense that both make use of logging and analyzing real executions of an implemented system. A difference is that our approach automates reasoning further by leveraging agent concepts of agent-based systems. Furthermore, various papers, such as [10], describe tools for the visualization of agent-based system behavior based on the presumption that visualization is essential to facilitate efficient debugging of agent-based systems. This approach is also adopted in this paper to aid rapid comprehension of agent behavior in the implemented system. In addition, a variety of analysis tools can process constructs in the visualization to further help automate agent software comprehension, as shown in this paper.

Obviously, in an ideal situation the verification of dynamic properties is done exhaustively (i.e., against all theoretically possible traces instead of a practically given set of traces). However, due to the expressiveness of the TTL language, this process is highly complex. Currently, the possibilities are investigated to combine recent model-checking techniques for real-time systems (such as [9]) with the presented approach.

5. ACKNOWLEDGMENTS

The authors are grateful to Jan Treur for his constructive comments, and to David Han for the development and analysis of the agents in the UAV system.

This research was funded in part by the Defense Advanced Research Projects Agency and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-00-2-0588. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed on implied, of the Defense Advanced Research Projects Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government.

6. REFERENCES

[1] Bosse, T., Jonker, C.M., Meij, L. van der, and Treur, J. A Temporal Trace Language for the Formal Analysis of Dynamic Properties. Technical Report, Vrije Universiteit Amsterdam, 2005.

[2] Chikofsky, E.J. and Cross, J.H. Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software*, vol. 7(1), 1990, pp. 13-17.

[3] Clarke, E.M., Grumberg, O., and Peled, D.A. *Model checking*. MIT Press, 2000.

[4] Edmonds, B. and Bryson, J. The Insufficiency of Formal Design Methods. In: *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'04*, 2004, pp. 938-946.

[5] Finnigan, P.J., Holt, R.C., Kalas, I., Kerr, S., Kontogiannis, K., Müller, H.A., Mylopoulos, J., Perelgut, S.G., Stanley, M., and Wong, K. The Software Bookshelf. *IBM Systems Journal*, vol. 36(4), 1997, pp. 564-593.

[6] Garcia-Molina, H., Germano, F., and Kohler, W. Debugging a Distributed Computer System. *IEEE Transactions on Software-Engineering*, SE-10 (2), 1984, pp. 210-219.

[7] Lam, D.N. and Barber, K.S. Comprehending Agent Software. In: *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'05*, 2005, pp. 586-593.

[8] Lam, D. N. and Barber, K. S. Debugging Agent Behavior in an Implemented Agent System. In: Bordini, R.H., Dastani, M., Dix, J., Seghrouchni, A.E.F. (Eds.), *Programming Multi-Agent Systems* (Selected Revised and Invited Papers), Springer LNCS, vol. 3346, 2005, pp. 103-125.

[9] Larsen, K.G., Petterson, P., and Yi, W. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1-2), 1997, pp. 134-152.

[10] Ndumu, D., Nwana, H., Lee, L., and Collis, J. Visualising and Debugging Distributed Multi-Agent Systems. In: *Proceedings of the Third Annual Conference on Autonomous Agents*, 1999, pp. 326-333.

The hidden dangers of experimenting in distributed AI

Annika Smit *
Masja Kempen
Niek Wijngaards
DECIS Lab
Thales Research & Technology Netherlands
P.O. Box 90, 2600 AB Delft, The Netherlands
+31 15 2517{865, 861, 867}
{annika.smit, masja.kempen, niek.wijngaards}@icis.decis.nl

André Meyer
DECIS Lab
TNO Defence, Security & Safety
P.O. Box 96864
2509 JG The Hague,
The Netherlands
+31 70 374 01 78
andre.meyer@{decis.nl, tno.nl}

ABSTRACT
Research on multi-agent systems often involves experiments, also in situations where humans interact with agents. Consequently, the field of experimental (human) sciences becomes more and more relevant. This paper clarifies how things can and often *do* go wrong in distributed AI experiments. We show the flaws in methodological design in existing papers (both with and without humans) and work out an example involving human test-subjects to introduce the fundamental issues of experimental design. Furthermore, we provide researchers with an approach to improve their experimental design. We wish to stimulate researchers to conduct better experiments – which will benefit us all.

Categories and Subject Descriptors
C.4 [**Performance of Systems**]: *measurement techniques*

General Terms
Measurement, Performance, Design, Experimentation, Human Factors, Standardization, Theory.

Keywords
Human-agent experimental design.

1. INTRODUCTION
In general, computer systems are built to perform on behalf of human users. Even autonomous software agents depend on human interaction for critical actions, because they are not (yet) seen as legal entities. Validation of human-agent interaction through experimentation is getting more attention lately in AI and computer science. Research in multi-agent systems more often involves humans as full-fledged participants in mixed-initiative interaction, for example in negotiation (e.g. where humans can set orders), decision support, workflow systems, etc. Actual empirical (human) research is conducted – which is an interesting and valuable trend – but needs to produce useful results.

Papers presenting experiments are seen more frequently as beneficial to our whole research community. At the same time, this community does not have a tradition in experimental design, especially not with humans. Valid and objective results are not easily gathered. Experiments need to be set up in such a way that peculiarities in human nature do not interfere with the experiment at hand. Psychology as a discipline has invested much time and effort into human experimenting – sound experimental methodology thus is available. It would be wise to apply this knowledge to experimentation in distributed AI.

Empirical research on human subjects is time consuming. It involves going through several necessary steps; Explicitation of the research question, or *hypothesis* formulation. Choice of testing method, or design. Decisions about *how* and *what type* of data will be gathered. What are the units of observation, and how many of them? Choice of data analysis methods. These issues are all crucial for choosing an *experimental design*, which refers to the formal structure of the experiment. The design *must* have at least the following properties: a) comparison of the observations under different levels of the independent variable should lead to non-ambiguous conclusions, and b) the comparison should be significantly testable.

Due to restrictions on the length of this paper we omitted the section on a brief overview of good experimental design. Instead we refer to [1], wikipedia.org, or any other good book on experimental design and statistics. Also unknown terms mentioned without explanation can be considered to be easily looked-up in such experimental handbooks. The important lesson to take home is that designing good experiments, best suited to your research question, is an art in itself. Invest time in it or you might end up with a meaningless experiment and a lot of time wasted. A final point we would like to make is that experiments should be reported in such a way that others can replicate the study.

In this paper, a number of erroneous experimental designs found in the latest AAMAS 2005 proceedings are used to illustrate common experimentation errors. We do not cite these researchers in detail, but appropriately name them 'Study 1, 2, etc'. Our aim is merely to use them as examples of well-intended, but wrongly applied efforts. Using a fictitious case, we show how proper experimentation methodology has to be applied and explain how to avoid some common pitfalls.

2. ERRORS in HUMAN EXPERIMENTS
We now focus on flawed human experiments to point out possible improvements. We confined ourselves to the papers of AAMAS

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

* Annika Smit currently works at ISC, The Netherlands; annika.smit@isc.politie.nl

2005, to show the relevancy of this subject. As we do not want to discredit researchers who are doing interesting research, we anonymously refer to the studies, which are numbered randomly.

People are adaptive systems who are constantly reacting to their environment and thus are prone to all kinds of influences – also the ones we do not want to measure. If a comparison is made between conditions (e.g. different interfaces) in terms of quality (e.g. people can work better with a certain kind of display), the different conditions cannot be presented in one specific order to all subjects. People are known to react differently in the beginning of an experiment than near the end. Numerous different factors play a role, such as practice effects, tiring effects, enthusiasm, time-of-day effects, etc. Therefore, the absolute basics of experimental design include a proper ordering of conditions. Unfortunately though, there are experimental studies that lack even this essential basic property [10]. Sometimes there seems to be a design, but it is not made explicit enough [6, 9].

Subjects need to be *randomly* assigned to the conditions. People differ from each other in many ways; these differences need to be averaged out by appointing people to different conditions in a random manner. Also important is what psychologists call 'external validity': The type of subjects used determines whether the findings can be extrapolated to other populations. It is a common mistake to use colleagues for experimentation [8]. Also the number of subjects is important. It is not useful to compare different conditions while using only two or three subjects. This may seem obvious, but experimental populations of this size are not uncommon at all [10].

Obviously, the period before actually conducting research entails the important basics of testing. But things can also go wrong in later stages. The description of procedures in papers is often vague and lacks important details. Methods and procedures are often not orderly described. However, standardization of method formats is common in many research areas. A clear format is the following: *Participants*, *Design & procedure*, *Material and stimulus presentation*, *Results (Statistical analysis*), *Discussion*.

Statistics are related to the aspect of subject amount. If only two or three subjects are tested, results can never be statistically significant. Unfortunately, in distributed AI research the specific type of statistical analysis used is often not stated [9]; sometimes no statistical analysis is done at all [10, 8].

3. STUDYING HUMANS PROPERLY

We here give an example. The data reported are fictitious, but realistic (see e.g. [3]). Let's assume that, for a Dutch auction (in which the price of the goods decreases over time), recommending trade agents have been implemented. These trade agents support the human buyer by indicating when it is the best time to buy. Legislation requires the human still to 'push the button', as indication of an actual order to buy; software agents do not have the (legal) authority to make a contract. However, supporting agents can be used which may indicate when to buy.

Whenever the time is ready to buy, the user should able to push the button *as fast as possible*. The researchers want to know with what type of interfacing human use the trading agents best. The clock used for the auction is shown on the user's screen. This graphical representation of the clock changes in some way, thereby showing the users the best moment to buy and push a button. An analogous clock and a digital clock are used; both count down prices. Four different conditions are compared: 1) analogous clock that changes background colour as indication to buy, 2) analogous clock that changes the clock-hand colour, 3) digital clock that changes background colour, 4) digital clock that changes the digits colours. The question is: Is there a significant difference in response time in the different conditions?

Experiment 1

Method

Participants. Eight students/… (female/male, average age) participated in this study (give background information).

Design & procedure. There were four 30-min conditions; a 10-min break followed after each condition. See Table 1 for the order of conditions. Using this type of counterbalancing, four subjects would have been enough, but probably *not* for significant results.

Table 1. Order of conditions (Balanced Latin Square design)

Subject	Order of conditions			
1, 5	1	2	4	3
2, 6	2	3	1	4
3, 7	3	4	2	1
4, 8	4	1	3	2

Material and stimulus presentation. This section should entail a description of all the materials and how stimuli were presented.

Results

A repeated measures ANOVA (Analysis of Variance) was used, with within-subject factor: Condition (the four different clock types). Usually data of one or more subjects cannot be analyzed: Equipment may fail or subjects do not finish the task. This is usually reported in order to inform researchers how many subjects may be 'lost' along the way. We assume that data of one of the subjects was 'lost'. Averaged RT data are shown in Table 2.

Table 2. Mean Reaction Times (RTs, in ms) of seven subjects to the buy-signal in the four different conditions

	Condition			
Subject	*1*	*2*	*3*	*4*
1	410	460	490	500
2	490	440	450	520
3	430	480	430	480
4	470	410	510	490
5	440	510	470	470
6	460	500	520	440
7	450	420	420	460

It is not enough to just present average results of the different conditions. Let's see what the data look like when averaged over all subjects (Figure 1). Pay attention to the distorting effect on the y-axis (that starts at 435).

Condition 1 seems to yield the fastest responses, but the ANOVA reveals no significant Condition effect ($p > 0.1$): The independent variable Condition (with four levels in this case) did not have a significant effect on RT (dependent variable).

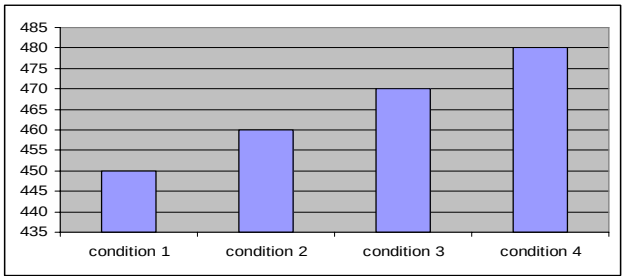


Figure 1. Performance measured in Reaction Times (ms), comparing four conditions, averaged over all seven subjects

Experiment 2: using the same example without methodology

If the four conditions were in the same order to all subjects (condition 1, 2, 3, 4.), the results would have been contaminated by fatigue effects, which have been found in the literature [4]. RTs may slow down 23% for the second and third presentation (compared to the first), and by 49% for the fourth presentation (compared to the first). If we now 'control' the data in Table 2 for these effects, the picture is totally different (see Figure 2).

The ANOVA yields a significant effect of condition; Condition 4 has the fastest response and Condition 1 the slowest [all p-values ≤ 0.01]. So now there indeed is a significant condition effect, but the effect is completely different from the picture in Figure 1. Of course, we do not know whether fatiguing effects occurred. Furthermore, there are a lot more effects that play a role [2]. All these effects are compensated for by using a proper design (see section 2).

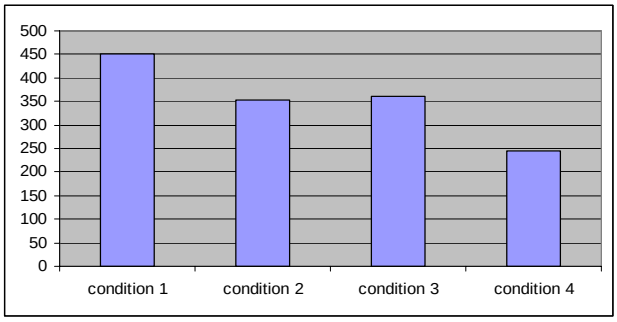


Figure 2. Performance measured in Reaction Times (ms), comparing four conditions, averaged over seven subjects (hypothetically 'controlled' for effect of fatigue, see text)

4. DISCUSSION & RECOMMENDATIONS

Experimentation is extremely useful, as it aids us to better understand strengths and weaknesses of our, models, theories and implementations. However, conducting experiments is very difficult. The difference between solid and weak methodology is the difference between valuable data and the absolute waste of time. Statistics can never make up for experimental design flaws. It is beyond the scope of this paper to provide a complete tutorial on sound experimental design. We have provided an introduction, including examples of common errors and a (fictional) case with a better experimental design. To support researchers in improving their experiments, we present a list of relevant issues and actions:

1. Think carefully about what questions you want to answer.
2. Understand that the area of experimental methodology is a specialized field; conducting experiments is very hard.
3. Accept that there are more effects out there that may harmfully influence your experimental results, therefore:
4. *Consult an expert* on methods & statistics (e.g. in the Psychology dept.). Do not try to do it all yourself.
5. Read appropriate literature on the basics of experimenting (ask the expert what would be useful in your case).
6. Let your expert analyze and comment on the design, methodology and statistical analysis you've chosen.
7. Report not only your results, but also your experimental design and procedure and chosen statistical analyses.

It is important that the distributed AI community improves its use of experimental designs. There is already research on software agents that have human-like capabilities, e.g. agents with memories, learning and adaptivity. Although agents do not get 'tired' as humans, they do live in resource-restricted environments (e.g., processing time, memory, etc.), which influence their performance. Conducting experiments with individual or groups of software agents is not an easy job, and many techniques used for human experimentation will also be necessary for experimentation with agents. This overlap in research areas is a topic we wish to explore in future publications and experiments.

5. ACKNOWLEDGMENTS

The research reported here is part of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024 (http://www.icis.decis.nl). The authors wish to thank the participants of the 'Creating Bonds with Humanoids'-workshop at the AAMAS 2005 conference, for raising awareness on the important issue of sound experimental design.

6. REFERENCES

Please note that the AAMAS 2005 papers are referred to anonymously; numbers are used in a random fashion (i.e. the order and numbers are not related to the papers they refer to).

[1] Elmes, D. G., Kantowitz, B. H., Roediger, III, H. L. *Research methods in psychology*, West Publishing Co: Minnesota (USA), 1985, 2nd edition.

[2] Kraepelin, E. *Die Arbeitscurve* [The working curve]. Leipzig: Wilhelm Englemann, 1902.

[3] Smit, A. S., Eling, P. A. T. M., & Coenen, A. M. L. Mental effort causes vigilance decrease due to resource depletion. *Acta Psychologica*, 2004, 115, 35-42.

[4] Smit, A. S. *Vigilance or availability of processing resources: A study on cognitive energetics*. Dissertation Thesis, Nijmegen, 2004, Chapter 3.2, pp 39-54.

[5] Study 1, in *Proceedings of the AAMAS 2005*, by ACM Press.

[6] Study 2, in *Proceedings of the AAMAS 2005*, by ACM Press.

[7] Study 3, in *Proceedings of the AAMAS 2005*, by ACM Press.

[8] Study 4, in *Proceedings of the AAMAS 2005*, by ACM Press.

[9] Study 5, in *Proceedings of the AAMAS 2005*, by ACM Press.

[10] Study 6, in *Proceedings of the AAMAS 2005*, by ACM Press.

[11] Study 7, in *Proceedings of the AAMAS 2005*, by ACM Press.

Agent-Community based Peer-to-Peer Information Retrieval – an Evaluation

Tsunenori Mine
Faculty of ISEE, Kyushu University
mine@is.kyushu-u.ac.jp

Akihiro Kogo
Graduate School of ISEE, Kyushu University
kogo@al.is.kyushu-u.ac.jp

Makoto Amamiya
Faculty of ISEE, Kyushu University
amamiya@is.kyushu-u.ac.jp

ABSTRACT

The Agent-Community-based Peer-to-Peer Information Retrieval (ACP2P) method uses agent communities to manage and look up information of interest to users. An agent works as a delegate of its user and searches for information that the user wants by communicating with other agents. The communication between agents is carried out in a peer-to-peer computing architecture. The ACP2P is implemented using the Multi-Agent Kodama framework. This paper presents how the ACP2P method works in an agent community network and show the experimental results to illustrate the validity of this approach.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.4 [**Information Storage and Retrieval**]: Systems and Software

General Terms

Algorithm, Design, Experimentation, Performance

Keywords

Multi-Agents, Information Retrieval, Peer-to-Peer

1. INTRODUCTION

Peer-to-peer (P2P) computing is an appealing approach not only to file sharing, but also to searching for information relevant to users over the large networks. Recently several studies on making use of document content information have been proposed[1][3][4]. They assume some cooperative environments where gathering all the documents from every peer in the network is possible for classifying all the peers in the network and for creating routing information between peers. However, in uncooperative environments such as business transaction environments, only the information exchanged between peers according to their transaction rules is available to them and such information is acquired only through their transaction process. In the case of Information Retrieval (IR) in such uncooperative environments, a peer can obtain other peer's documents only through their retrieval process. In other words, when a peer issues a query to other peers, the peer can receive only the documents returned by the peers. The retrieved documents a peer acquired can be re-delivered to other peers according to a handling rule of the documents. We are interested in applying agent technologies to such uncooperative environments. Considering the above things, we proposed an Agent-Community-based Peer-to-Peer information retrieval method called ACP2P method, which uses agent communities to manage and look up information related to a user query[6][5]. The agent communities can reflect the structures of human groups or societies such as laboratories, departments, institutions, research groups and so forth, where the people with the same or similar interests, objectives or aims stay together. In the prior research, we have demonstrated the method's effectiveness for reducing communication loads through several experiments[5]. However, we have not so far well discussed the accuracy performance of the method. This paper considers the ACP2P method and discusses the experimental results to illustrate the validity of this approach.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. OVERVIEW OF THE ACP2P METHOD

The ACP2P method employs three types of agents: user interface (UI) agent, information retrieval (IR) agent and history management (HM) agent. A set of three agents (UI agent, IR agent, HM agent) is assigned to each user. An IR agent communicates with other users' IR agents not only in the community it belongs to, but also in other communities, to search for information relevant to its user's query. A pair of a Query and Retrieved Document Histories (Q/RDH) and a Query and Sender-Agent address Histories (Q/SAH) and retrieved document content files are managed by the HM agent. When receiving a query from a UI agent, an IR agent asks an HM agent to look up target agents with its histories or asks a portal agent to multicast a query to all the IR agents in its community. Where a portal agent is the representative of all IR agents in its community and can multicast a message to them. Any IR agent in a community can ask the portal agent to multicast its query with requesting the number of relevant information to be returned. After multicasting a query, the portal agent only receives the requested number of 'Yes' messages from the IR agents in the order the

messages are received and returns them to the query-sender IR agent. When receiving a query from other IR agents, an IR agent looks up the information relevant to the query from its original document and retrieved document content files, sends an answer to the query-sender IR agent, and also sends a pair of the query and the address of the query-sender IR agent to an HM agent so that it can update Q/SAH. The returned answer is either a pair of a 'Yes' message and retrieved documents or a 'No' message indicating that there is no relevant information, although retrieved documents are not returned when the query comes through a portal agent. When receiving answers with a 'Yes' message from other IR agents, the IR agent sends them to a UI agent, and sends them with a pair of a query and the addresses of answer sender IR agents to an HM agent[5].

3. EXPERIMENTS

3.1 Preliminaries

We use the Web pages of Yahoo! JAPAN, which are broadly divided into five categories: animals, sports, computers, medicine, and finance. We select 20 smaller categories from each of them in descending order of the number of Web pages recorded in a category and assign an IR agent to each selected category. A category name is used as the name of an IR agent, and the Web pages in the category are used as the original documents of the agent. To perform the experiments, we compare three methods : (1) ACP2P using the two histories (wQ/SAH for short), (2) ACP2P using the Q/RDH, not using the Q/SAH (woQ/SAH for short), and (3) a simple method always employing a 'multicast' technique (MulCst for short). In the experiments, two query sets : QL=1 and QL=2, are used. QL=1 and QL=2 consist of 10 queries, whose query length is one and two, respectively, where query length means the number of terms in a query.

3.2 Relevance Judgement and Evaluation

In a P2P network environment, gathering all documents from every peer is not always possible, that is, indexing all documents is quite difficult. Thus the first goal for IR in the P2P network environment is to achieve a result comparable with a conventional IR method using a Centralized Indexing DataBase (CIDB method for short) because distributed information retrieval systems, which an IR system in P2P networks belongs to, are not yet better than the "single collection" baseline[3] and the validity of this method is empirically shown[4]; the automatically generated queries and the "single collection" baseline are useful resources in studying federated search (distributed information retrieval) in peer-to-peer networks[4]. As the CIDB method, we employ a probabilistic IR method that applies the BM25 [7] weighting function to all the documents collected from every peer. We use the following equation:

$$\sum_{T \in Q} \log \frac{n + 0.5}{N - n + 0.5} \frac{2tf}{\frac{dl}{avdl} + tf} \quad (1)$$

Where Q is a query that contains terms T. tf is the frequency of occurrence of the term within a specific document. N and n are the number of items (documents) in the collection and the number of documents containing the term, respectively. dl and $avdl$ are respectively the document length and average document length, where the document length is the number of terms in a document, and a term is a word detected by a morphological analyzer. In order to compare the ACP2P method with the CIDB method, we used the following equation:

$$\sum_{i=1}^{N_R} \frac{1}{r(i)} / \sum_{i=1}^{N_R} \frac{1}{i}$$

Where $r(i)$ is the CIDB method's rank of the document that is ranked by the ACP2P as the ith document. For example, if a document is ranked by the ACP2P as the 2nd document and the document's rank by the CIDB method is 3, then this means that $r(2)$ returns 3. We call this measure *Reciprocal Rank Similarity* (RRS for short). We can assume that RRS's denominator $\sum_{i=1}^{N_R} \frac{1}{i}$ represents the ideal value of a given model, where it is the CIDB method in this paper. As the ACP2P approaches the given model, the RRS value becomes higher. Thus, the RRS can measure the similarity between ranks generated by the ACP2P and by the CIDB and returns a higher score the smaller $r(i)$ $(1 \leq i \leq N_R)$ is, i.e., the higher the rank. For example, if a user wants to find 3 documents relevant to his/her query and we suppose the top 3 ranked documents' rank returned by his/her agent to be 3, 5 and 1, then the RRS returns $\frac{1/3+1/5+1/1}{1/1+1/2+1/3} = 0.84$, and the top 3 ranked documents' rank to be 3, 5 and 2, then the RRS returns $\frac{1/3+1/5+1/2}{1/1+1/2+1/3} = 0.56$. In the experiment, we use the average RRS: $\frac{1}{N_a} \sum_{i}^{N_a} RRS(i)$, where N_a is the number of all IR agents and $RRS(i)$ is the RRS of the ith IR agent.

3.3 Experimental Results and Discussions

We compare the RRS values of the three methods. All 100 IR agents are assigned to a single community so that we can consider the basic performance of the ACP2P method. The results are shown in Figure 1. In the figure the vertical axis is the average RRS and the horizontal axis is the number of queries issued by each IR agent. As the value of N_R increases, the RRS value also increases and the curve of the graphs becomes flatter. We can see that the RRS value of MulCst increases as the number of queries sent increases. Considering this phenomenon, we surmise that original documents assigned to IR agents will gradually be spread over the community through the document retrieval process of each IR agent. Thus even though a portal agent selects target agents in the order their 'Yes' messages are received, the probability that higher weighted documents will be returned rises. For the same reason, since the RRS value of the MulCst increases as N_R increases, the difference of the three methods decreases. When using QL=1, the wQ/SAH almost achieves higher retrieval accuracy than the other two methods, although the RRS value is unfortunately not so high because the records stored in the content files and the two histories are originally acquired by a portal agent using the query multicasting technique and its RRS value is not so high. We think there are at least two reasons why the RRS score was not so high: 1) a portal agent selects target agents in the order their 'Yes' messages are received, and 2) the ranking results calculated by each IR agent is different from those by the CIDB method even though they use the same equation (1) in Section 3.2 because N used by each IR agent of the ACP2P and that by the CIDB method are different. The first one might be able to be settled by making the portal agent receive the 'Yes'/'No' messages of all IR agents and send them to the query-sender agent although it consumes much more time.

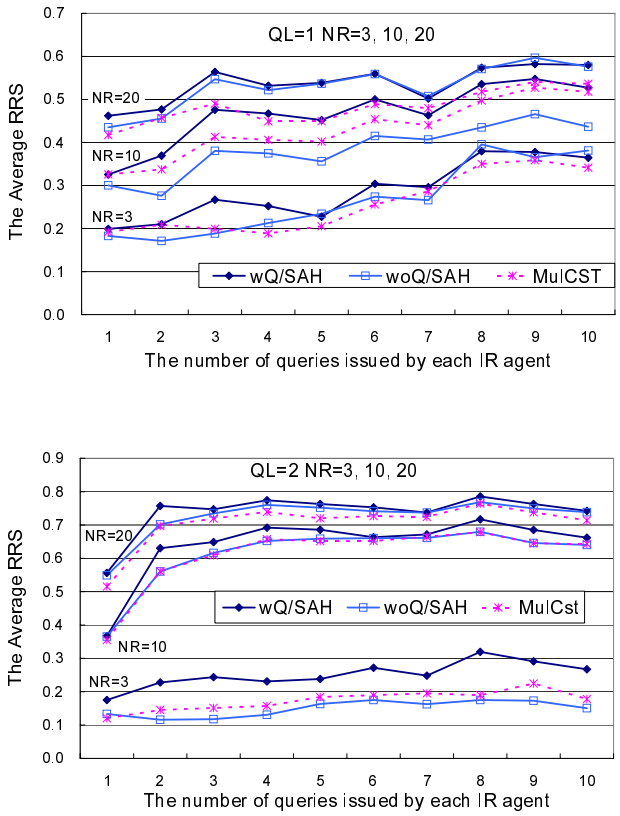


Figure 1: The comparison of average reciprocal rank similarity (RRS) of each IR agent for every query input using 3 different N_R values : N_R=3, N_R=10 and N_R=20. The query belongs to QL=1(Top) and QL=2(Bottom).

However, the second one is not so easy because N used by each IR agent will be changed through its retrieval process. Although there are several methods proposed for distributed information retrieval to estimate the number of documents in each database (DB size for short),[2], [8],[9], they assume such DB size is not varied, thus the number of documents in the network, i.e. N will not be varied either. On the other hand, we assume the DB size will be varied through the retrieval process of IR agents. Therefore a method for estimating N is necessary. In order to estimate N, we first need to estimate the total number of original documents in the network (N_{od}). As a straightforward method, we are trying to use the following simple equation. $N_{od} = \sum_{i \in YES_{IRA}} \frac{N_{QSA_o}}{n_{QSA}} * n_i + N_{QSA_o} * (N_{IRAs} - |YES_{IRA}|)$, where YES_{IRA} is the set of IR agents returning a 'Yes' message o a query issued by a query-sender IR agent. N_{QSA_o} is the number of original documents the query-sender agent has. n_{QSA} is the number of the query-sender agent's original documents relevant to the query. n_i is the number of $IR\ agent_i$'s original documents relevant to the query. N_{IRAs} is the number of IR agents in the network, where 100 in this paper. We assume that N_{IRAs} can be told by the portal agent. $|YES_{IRA}|$ is the number of IR agents in YES_{IRA}. This equation assumes that for every IR agent returning a 'Yes' message, the ratio of the number of original documents relevant to a query and the number of the original documents is the same and the number of the original documents of the IR agent returning a 'No' message is the same as the number of the query-sender agent's original documents. As well as N, we also need to estimate n and $avdl$ in equation (1). n can be estimated by letting the query-sender agent issue a query to all the IR agents returning a 'Yes' message, receive all the documents relevant to the query and count the number of original documents of the IR agents. Where every document should have the information whom it is originally owned or created by. $avdl$ can be set to the average document length of the whole documents the query-sender IR agent has. These are quite rough assumptions, but we have already received positive response from preliminary experimental results, although we need further investigation to show the detail.

4. CONCLUSIONS AND FUTURE WORK

We presented the ACP2P method and discussed the experimental results to illustrate its validity. To do the experiments, we implemented the method with Multi-Agent System Kodama. The experimental results showed that Q/SAH history help to have a higher accuracy in retrieving documents than a method without using the history. Discussing experimental results using more than one hierarchical agent community under more dynamic environments is future work.

Acknowledgment

This research was partly supported by the Grant-in-Aid for Scientific Research (C) (16500082) from the JSPS and the SCOPE-C (052310008) from the MIC, Japan.

5. REFERENCES

[1] M. Bawa, G. S. Manku, and P. Raghavan. Sets: search enhanced by topic segmentation. In *Proc. of SIGIR03*, pages 306 – 313, 2003.

[2] K.-L. Liu, C. Yu, and W. Meng. Discovering the representative of a search engine. In *Proc. of CIKM01*, pages 577–579, 2001.

[3] J. Lu and J. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *Proc. of CIKM03*, pages 199–206, 2003.

[4] J. Lu and J. Callan. Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In *Proc. of ECIR05*, 2005.

[5] T. Mine, D. Matsuno, A. Kogo, and M. Amamiya. Design and implementation of agent community based peer-to-peer information retrieval method. In *Proc. of CIA04, LNAI 3191*, pages 31–46, 9 2004.

[6] T. Mine, D. Matsuno, K. Takaki, and M. Amamiya. Agent community based peer-to-peer information retrieval. In *Proc. of AAMAS04*, pages 1484–1485, 7 2004.

[7] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi/keenbow at trec-8. In *NIST Special Publication 500-246: TREC-8*, pages 151–162, 1999.

[8] L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In *Proc. of SIGIR03*, pages 298–305, 2002.

[9] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *Proc. of SIGIR05*, pages 449–456, 2005.

An Overview of Business Process Adaptations via Protocols *

[Extended Abstract]

Nirmit Desai Amit K. Chopra Munindar P. Singh
Department of Computer Science
North Carolina State University, Raleigh, NC 27695-8206
{nvdesai, akchopra, singh}@ncsu.edu

ABSTRACT

Business process management in open environments poses special challenges. In particular, such environments are dynamic, thereby requiring frequent changes in business processes. Current business process modeling approaches handle such changes in an ad hoc manner, and lack a principled means of determining what needs to be changed and where.

This paper provides an overview of process adaptability through a novel application of business protocols, especially of protocol composition, introduced in our previous work. Through a real business scenario of auto-insurance claim processing, this paper briefly describes how a wide range of adaptations can be handled naturally and systematically via protocol composition. The illustrated adaptations have been evaluated via a prototype.

1. INTRODUCTION

Successful business process management requires supporting three key properties of open environments, namely, *autonomy*, *heterogeneity*, and *dynamism* [4, pp. 7–10]. Supporting autonomy means modeling and enacting business processes in a manner that offers maximal flexibility to the participants by only minimally constraining their behavior. Supporting heterogeneity means making as few assumptions as possible about the construction of the components for the parties, concentrating instead on characterizing the interactions among them. For open environments, interconnections among components, not the components themselves, are critical.

Dynamism has two flavors. *Membership dynamism* means that the set of components may change. Components may be added and removed at either design-time (*design-time membership dynamism*) or run-time (*runtime membership dynamism*). *Structural dynamism* means that the set of interconnections is not fixed. Dynamism reflects changes in the business requirements; modern businesses face intense pressure and must repeatedly reconfigure themselves in order to thrive, if not to survive. As changes in requirements are routine, an elegant way of handling such changes is vital. Thus, dynamism poses a difficult challenge, one that has not been adequately addressed in the literature, and is the theme of this paper. Supporting dynamism implies supporting process adaptability.

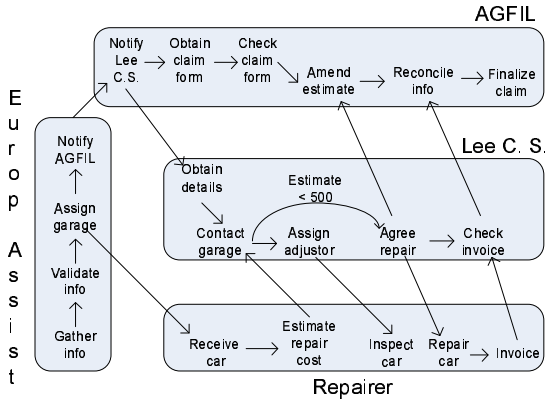


Figure 1: CrossFlow Insurance Claim Processing

Previously, Desai et al. [2] reported a methodology (named OWL-P) for developing business processes that supports autonomy and heterogeneity. The essence of OWL-P is an abstraction of business protocols. A *business protocol* is an abstract, modular, and publishable specification of rules that govern a business interaction among two or more *roles*. A protocol is abstract because it does not model the proprietary business logic of the agents enacting the roles. It is modular because it achieves a specific goal which may be a part of another bigger goal. Due to their abstract and modular nature, business protocols, each achieving a goal, can be combined to form a *composite protocol* that achieves a bigger goal. Each role's perspective of a protocol is captured in its *role skeleton*. A role skeleton, when integrated with the private business logic of an agent enacting the role, yields the agent's *local process*. A *business process* is an aggregation of the local processes of all the agents involved.

In this way, protocols partition a process in a cross-cutting manner whereas conventionally, flows are taken as partitions. For instance, peeking ahead on an example studied in great detail below, each of the boxes in Figure 1 represents a flow. Such boxes would be the modules of abstraction in conventional approaches. By contrast, OWL-P focuses on the interconnections among the boxes; such interconnections are a natural fit for the interorganizational nature of SOC environments [2]. As a striking illustration of the

*This research was partially supported by the NSF under grant DST-0139037 and by DARPA under contract F30603-00-C-0178.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

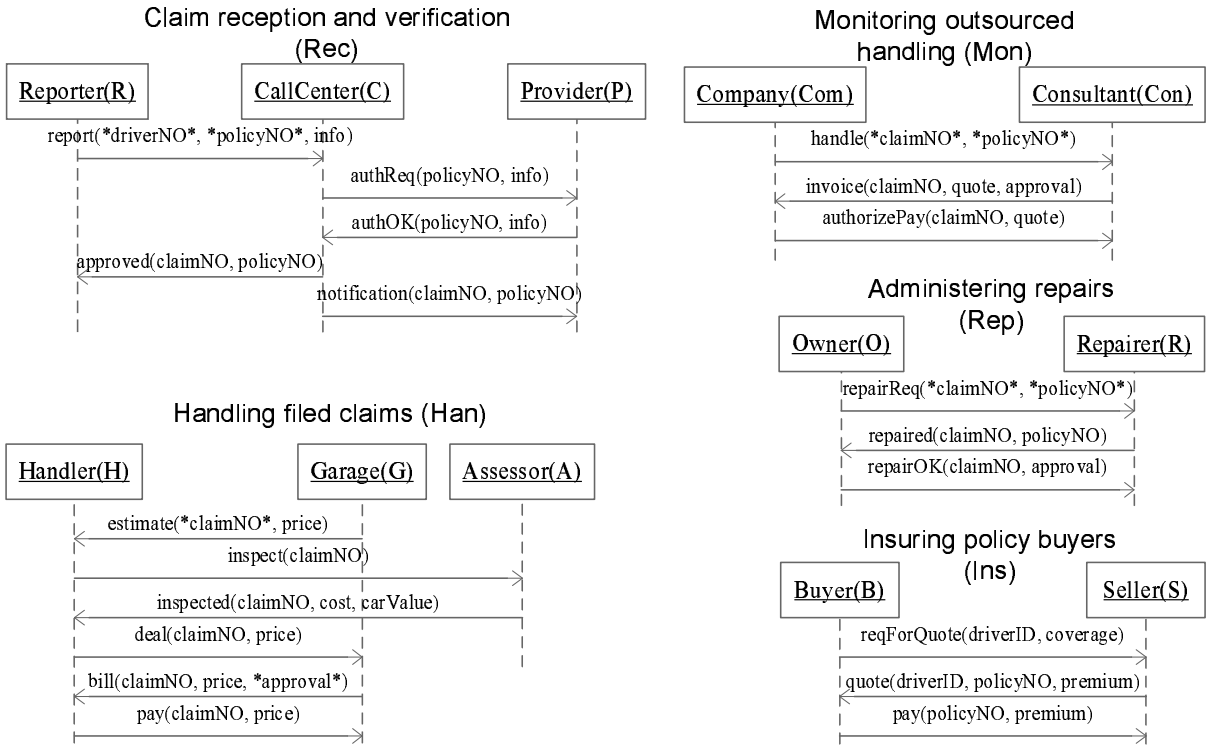


Figure 2: Scenarios from the protocols corresponding to the insurance claim process of Figure 1

importance of focusing on the interconnections, notice that the process flow (taken from [1]) in Figure 1 misses the insurance holder and its interactions altogether. Although the internal flow of the insurance holder's process (its box) may not be important to AGFIL, its external interactions with other parties (i.e., the insurance holder's interconnections) are crucial for modeling the business of AGFIL. Figure 2 shows how the above example is modeled via protocols.

Previous work on protocols provides the basic mechanisms to specify, compose, and enact them to produce software engineering benefits such as reusability and flexibility. This paper is a sketch of protocol-based treatment of dynamism in a real business scenario. Adaptability in general involves a broad variety of research challenges. Hence, this paper concentrates on three kinds of adaptations: (1) exceptions, (2) changes in business policies, and (3) changes in business models.

To illustrate the protocol-based approach, this paper considers the above insurance claim processing case studied under the CrossFlow project [1]. Being a real-life business scenario previously studied by others, this example provides an independent and significant test-case for this approach. Figure 1 shows the parties involved in the process and the process flow. AGFIL is an insurance company in Ireland. The present scenario deals with automobile insurance. AGFIL underwrites the insurance policy and covers any losses incurred. Europ Assist provides a 24-hour help-line service for receiving claims. Lee CS is a consulting firm that coordinates with AGFIL and deals with repairers to handle the claims. A network of approved repairers provide repair services. AGFIL holds ultimate control in deciding if a given claim is valid and if payment will be made to the repairer.

2. PROTOCOLS

Figure 2 shows scenarios from the protocols involved in the above insurance example. Such protocols are independently developed and stored in a repository for reuse.

Any realistic business process would involve multiple interaction protocols. For example, AGFIL would need protocols for selling insurance policies and for receiving and handling claims. Consider how protocols can be combined to form a composite protocol. Let's assume AGFIL outsources the help-line service for receiving claim reports to a call center, but handles the claims itself. The requisite process can be obtained by combining the *Ins* and *Rec* protocols in a certain way yielding a new *Bas* (Basic insurance claim processing) composite protocol.

As the ontologies of the protocols may not match, and they may have interdependencies, simply unioning the rules of the component protocols is not enough. The constraints on combining protocols are specified as a set of *composition axioms* in a composition profile. These axioms identify roles across protocols, specify data flow dependencies, event ordering constraints and ontological dependencies. Composite protocols can be treated and enacted like other protocols. The details of protocol specification and composition are discussed in our previous work [2].

Typically, the protocols are enacted with some of the roles having business relationships established via participating in some other protocol, e.g., *Ins* establishes the contract between the *buyer* and the *seller*. This business relationship is a contract represented by a set of commitments. For brevity, this presentation skips such relationship formation protocols. Figure 3 shows all such contractual commitments among the roles after all the relationships are formed and commitments delegated.

3. ADAPTING PROCESSES

Here, we describe what kinds of adaptations can be handled

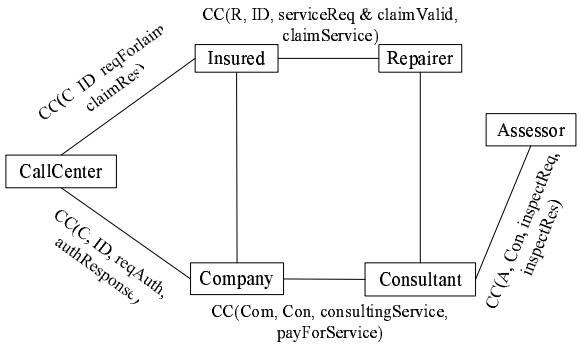


Figure 3: Commitments among parties

by applying the mechanism of protocol composition. Each of the adaptations feature varying degrees of autonomy, heterogeneity, structural dynamism, and design-time membership dynamism. The following assumes that a composite protocol *Picp* (Partial insurance claim processing) is constructed from the composition of *Bas*, *Mon*, *Han*, and *Rep* of Figure 2.

3.1 Exception Handling

Exceptions are (typically) uncommon conditions arising during a business interaction, e.g., fraudulent auto-insurance claims. Protocols facilitate incorporating the handling of an exception into an existing process.

A composite protocol *Ficp* (Fraudulent insurance claim processing) can be constructed by combining *Picp* with a protocol *Fra* (fraud detection and reporting) When AGFIL and Lee CS form their relationship, a conditional commitment CC(Com, Con, consultingService, payForService) is created meaning that AGFIL will authorize payments for handling individual claims if Lee CS provides the consulting service. In case of fraud, the *company* cancels the policy coverage of the *insured* and the *consultant* releases the *repairer* from the commitment to perform repairs. The *assessor*'s advising of a fraud should discharge CC(A, Con, inspectReq, inspectRes) and the *consultant*'s forwarding it to the *company* should mean the consulting service was provided.

The resulting interaction allows AGFIL to end the process by canceling the commitments in case of a fraud. Simply adjusting the commitments can allow greater flexibility without affecting other parties adversely. Here, the support for structural dynamism comes from the added interconnections among the existing partners to handle the exceptions. Support for autonomy comes from the ability to control the effects of foul behavior of a participant.

3.2 Changes in Business Policy

In handling claims where the value of the car is less than the estimated cost of repairs, the *company* may want to scrap the car, i.e., declare it a total loss. To settle this, it pays the *insured* a sum equal to the value of the car and takes possession of the car instead of administering repairs. Also, if the damage is minor, the *insured* may accept a cash settlement instead of having the car repaired. A protocol *Pcsc* (Pay cash and scrap car) can specify the interactions needed for such a policy.

AGFIL would have delegated its conditional commitment CC(S, B, serviceReq $\wedge$ claimValid, claimService) to Lee CS, thereby making Lee CS responsible for servicing claims, who would in turn delegate it to the *repairer*. When the *consultant* advises scrapping the car, it fulfills its commitment to provide the consulting service and the commitment to service the claim falls back to the *company* as the car is not to be repaired but to be paid for. Similarly, when the *consultant* advises a cash payment for a minor damage, and the *insured* agrees to it, it fulfills the commitment for the consulting service and the *company* becomes committed to paying the *insured*. This can be modeled by delegation. Assuming that *Picp* is the current interaction protocol among the partners, *Pcsc* can be composed with *Picp* yielding the composite protocol *Icp* (Insurance claim processing) that handles this policy change.

In *Icp*, a business policy change internal to AGFIL is accommodated across the business process. Here, support for structural dynamism comes from the modified interconnections among the existing partners to handle the policy change. Support for heterogeneity comes from the fact that a change of business logic internal to AGFIL could be accommodated.

3.3 Changes in Business Model

Quite often, enterprises change their business models to respond to changes in their business environment. Business process outsourcing (BPO) [3] has recently become a popular approach for conducting business.

The composition of *Bas* discussed in Section 2 already illustrated this type of adaptation. Let's assume AGFIL could handle claims itself. AGFIL outsourced its call center to Europ Assist to focus on its core business of selling insurance policies. Thus, the business model of AGFIL changed to have a partnership with Europ Assist.

Obviously, a change in the business model is a big shift for an enterprise; new partnerships are formed and new interconnections among the parties emerge. Thus, such a change features not only structural dynamism, but also design-time membership dynamism. Both are supported via composition with protocols having new roles.

4. REFERENCES

[1] CrossFlow consortium / AGFIL. Insurance (motor damage claims) scenario. Technical report, CrossFlow consortium, 1999. Document identifier: D1.a, http://www.crossflow.org.

[2] N. Desai, A. U. Mallya, A. K. Chopra, and M. P. Singh. Interaction protocols as design abstractions for business processes. *IEEE Transactions on Software Engineering*, 31(12):1015–1027, 2005.

[3] P. Harmon. *Business Process Change: A Manager's Guide to Improving, Redesigning, and Automating Processes*. Morgan Kaufmann, 2002.

[4] M. P. Singh and M. N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. Wiley, 2005.

Multi-agent based Community Computing System Development with the Model Driven Architecture

Yuna Jung
Artificial Intelligence Laboratory
Ajou University, Suwon
Korea(South)
+82-31-219-2442
serazade@ajou.ac.kr

Jungtae Lee
Programming Language Laboratory
Ajou University, Suwon
Korea(South)
+82-31-219-2436
jungtae@ajou.ac.kr

Minkoo Kim
Artificial Intelligence Laboratory
Ajou University, Suwon
Korea(South)
+82-31-219-2437
minkoo@ajou.ac.kr

ABSTRACT
Ubiquitous system can be considered as a system, in which many kinds of component are connected by networks and the components cooperate with each other for achieving some goals. To develop a ubiquitous system, several researches have used multi-agent models. Even thought current multi-agent models provide many benefits with ubiquitous system developments, they are not enough to develop ubiquitous systems. In this paper, we address the ubiquitous system which solves its problems with cooperative organization consisting of several heterogeneous computing components. We call such a ubiquitous system as a community computing system. In the community computing system, some goals are given to be achieved by the cooperative organization. In this paper, we propose a high-level abstraction model of multi-agent based community computing system. In addition, we propose the development process of the community computing system using MDA(Model Driven Architecture) approach.

Categories and Subject Descriptors
I.2.11 [**Distributed Artificial Intelligence**]: Distributed Artificial Intelligence – *Multi-agent systems, Coherence and Coordination.*

General Terms
Design, Languages

Keywords
Multi-agent systems, Ubiquitous, Community, MDA

1. INTRODUCTION
A ubiquitous system can be regarded as a multi-agent application which target support for daily-life activities. Multi-agent based ubiquitous systems are more than just an extension of multi-agent system to ubiquitous domain. General multi-agent models can not completely support specific features of ubiquitous systems. In this paper, we focused on the agent-based ubiquitous system solving the problems with mission-oriented organizations and called such ubiquitous system as the community computing system. In the community computing system, a community consists of several agents and provides ubiquitous services can not provided by individual computing component. The community computing system has features such as mission-oriented organization, role, goal, cooperation between roles or organizations, and so on. Such specific features of the community computing system can not be completely modeled using existing multi-agent models because existing models deal with general and low-level concepts. Therefore, in this paper, we propose a new high-level abstraction model using the community metaphor. In addition, we concern the development of the community computing system as well as high-level abstraction. In deed, it is hard to develop a highly complex distributed system using only a design. Therefore, we try to formulate the development process from high-level abstraction to implementation of a multi-agent based community computing system with model transformation of MDA.

The rest of the paper is organized as follows. We first introduce some related works and motivation in section 2. Then, in section 3, we propose the community computing system and its development process. Section 4 is dedicated to the conclusion and future works.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8–12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005…$5.00.

2. RELATED WORK
Gaia introduced a methodology for agent-oriented analysis and design [1]. In Gaia, a multi-agent system is regarded as a collection of computational organizations consisting of various interacting roles. Such a view of Gaia is analogous to our view of a cooperation based ubiquitous system. However, it is not adequate to apply agent-oriented methodologies including Gaia to develop a multi-agent based cooperative ubiquitous system. One reason is that they can not capture characteristics of the cooperation based ubiquitous computing. In the agent models, even in Gaia, an organization is only implicitly defined such structures within the role and interaction models. In the cooperation based ubiquitous computing systems, it is important to explicitly define organizations, because the objective of a system is realized by achieving organization's goals. Therefore, we need a high-level abstraction model to describe organization's structures and interaction behavior within and between them. To do this, we employ community metaphor. Community concept is introduced by several agent cooperation models [2]. In ubiquitous domain, PICO (Pervasive Information Community Organization) project [3] used the community concept. The community of PICO provides a framework for collaboration and coordination among

agents. The concept of community computing satisfies requirements of cooperation based ubiquitous computing such as proactive real-time collaborations for automated, continuous services providing in heterogeneous environment. Therefore, we choose the community as a metaphor describing organization's structure in a multi-agent based cooperative ubiquitous system, and then describe the interrelationships between communities. In this paper, we define that the ubiquitous system modeled by community concept as the community computing system.

Another reason is the lack of well defined and automated development process from design to implementation. Agent-oriented models only care about analysis and design of a system. This situation requires solid development process from analysis of system requirement to implementation. To define the development process of multi-agent based community computing systems, we employ MDA (Model Driven Architecture) [4]. MDA is an approach to system development, which increases the power of models in that work. Using MDA approach provides benefit to the development of community computing systems. It make the ubiquitous systems developers focus on describing the system using high-level abstraction primitives, community. Then, the high-level abstraction model is transformed to the more detailed model until a frame of source code is obtained [5].

3. COMMUNITY COMPUTING SYSTEM AND ITS DEVELOPMENT PROCESS

In this paper, we propose the high-level abstraction model and development process for multi-agent based community computing systems through the model transformation process of MDA. For applying MDA approach, we need several models representing different abstraction level and modeling languages for each model.

Before presenting each model in detail, it is worth introducing the basic concepts constructing the core of community computing.

- *Member*: it is the building blocks of a community computing system, and be implemented as an agent. It can represent a computing device, physical hardware, human, and software. At ordinary time, it performs its own action. Then, triggering a goal, it forms a community, and interact each other to achieve the goal. In this paper, we assume that all templates of member, which can be existed in a system, are predefined.
- *Community*: It is a metaphor to describe a proactive organization, and made up with agent members collaborating with others to achieve particular missions. A community has goals and protocols. A community can manage the life cycle of its own members. Offering a condition of joining into a community, a community can control entry and secession of each member. As a goal dynamically arises, a community is dynamically generated to achieve the goal. Then, the community is disorganized when the goal is attained. In this paper, we assume that community templates and the structure of each community are predefined. That is, a community, which is not predefined, is can not created in a system. As well as templates of community, goals and protocols of a community are also fixed.
- *Goal*: It is a particular condition that a community should satisfy. In short, it is the objective of a community.
- *Protocol*: It is a description of interaction between community's members to achieve a goal. It defined as a sequence of messages and member's actions. While a community performs a protocol, another goal can be issued.
- *Society*: It is a metaphor to represent U-space, the environment of a system. It is constructed by agent members and communities. At the first time, only members exist in society. After the start of the society, a community is generated if an agent recognizes a goal. When it attains its objective, a community is terminated, but the society is alive until the community computing system is exhausted.

First of all, as a most high-level abstraction model, CCM (Community Computing Model) describes the multi-agent based community computing system with community concept. Its objective is modeling the requirement and environment of a system. At runtime, a community computing system makes the space combined virtual environment and physical environment. We called this space as *U-space* (Ubiquitous space). *U-space* is a logical and physical area where ability of a ubiquitous system can influence. *U-space* can be defined as a collection of agents and communities in a CCM. To describe the CCM, we defined a modeling language, CML (Community computing Modeling Language). A CIM-PI considering implementation of Community computing system without concern about specific platforms. It shows the system in detail with computing elements and interoperation between them. A CIM-PI extends community descriptions and society descriptions of CCM in detail. In community description of CIM-PI, it is represented that mapping relationships between roles and member templates and interactions of protocols. Each role of the community is matched with certain member templates can playing the role. In addition, protocols of the community are detailed. The frame of CIM-PI (Platform Independent Community Computing Implementation Model) can be derived from CCM, and the rest should be filled by developer. To express a CIM-PI, we define CIL-PI as a modeling language. To describe implementation of a system depending on a specific platform, we also define a CIM-PS (Platform Specific Community computing Implementation Model). The skeleton of CIM-PS can be also derived from a CIM-PI, but the rest should be filled manually. A CIM-PS is represented as program codes in specific agent platform such as JAM or JADE.

Using the model transformation process from a CCM to a CIM-PS, we can obtain the frame of source code. This process allows community computing system development to be ease and systematic, because the gap between design and implementation is filled. Furthermore, we can guarantee consistency in the entire development process, because we can manage the process with coherent philosophies.

```
Space GHODAM_U-Space {
Community TRAFFICACCIDENT {
  Roles PATIENT : 1 ~ 500 {
    Attribute : BODYCONDITION={NORMAL|ABNORMAL};
                CALLAMBULANCE={YES|NO};
  STATUS={MOVE|STAY|CLASHED};
    Cast : STATUS= CLASHED; }
  Roles CAR : 1~ 500 {
    Attribute : STATUS={NORMAL|CLASHED};
                DRIVING MODE={PARKING|REVERSE|DRIVE|NEUTRAL|STOP};
    Cast : DRIVINGMODE=STOP; STATUS= CLASHED; }
```

```
Goals TRAFFICACCIDENTHANDLING {
    Protocol request_for_ambulance_service :
    (Initiator=PATIENT or CAR, Participantr= PATIENT or CAR) {
    Subprotocol : AMBULANCE-ASSOCIATION.AMBULANCE SERVICE,
                  patient profile; } }
  Ontology : TRAFFICONTOLOGY; }
Community AMBULANCE-ASSOCIATION { ..... }
Community EMERGENCYAMBULANCE { .....}
Society GHODAMCITY{
  Member GHODAMCITIZEN {
Attribute : ADDRESS-CITY= STRING; Cast : ADDRESS-CITY= GOTHAM; } } }
```

Figure 1. An example of CCM using CML

```
Platform Independent Community Implementation Description {
Space GHODAM_U-Space {
 Community TRAFFICACCIDENT {
    PATIENT:Human; CAR:Car ;
    Protocol request_for_ambulance_service {
      Communication of Initiator {
        SEQ
          make_profile_of_Patient;
          Send_Msg(request, Participant:1, profile_of_Patient);
        CASE(failure, Participant:1, All ambulance is buzy)
          Send_Msg(request, Participant:1, profile_of_Patient);
        CASE(inform, Participant:1, Ambulance_ID)
          Waiting_for_Ambulance(Ambulance_ID);   }
      Communication of Participant {
        CASE(request,Initiator:1, profile_of_Patient))
          SEQ
            Check_Idle_Instance(Ambulance,Driver,Paramedic);
            Select_Instance(Ambulance,Driver,Paramedic);
            ALT Send_Msg(failure, Initiator:1, All ambulance is buzy);
                Send_Msg(inform, Initiator:1,Ambulance_ID);}  } }
                  ..................................                    }
 Society: GHODAMCITY {
   Member Society Member {
    Attribute : LOCATION=GHODAMCITY; CITY_ADDRESS=GHODAM;}
   Member Animate Object extends Society Member {
    Attribute : SPECIES=STRING; GENUS=STRING; FAMILY=STRING;}
   Member Human extends Animate Object {.....}
   Member Inanimate Object extends Society Member {.....}
   Member Transportation Object extends Inanimate Object {.....}
    Member Car extends Transportation Object { .....}
 Member Ambulance extends Car {.....} }}
```

Figure 2. An example of CIM-PI using CIL-PI

```
GOALS: ACHIEVE trafficaccident_handling "patient";
FACTS:
Plan: { NAME: "request for ambulance service"
        GOAL: ACHIEVE trafficaccident_handling $PATIENT;
              ASSERT patient $PATIENT;
        BODY: }
Plan: { NAME: "communication of initiator"
        GOAL: PERFORM initiator;
        BODY: RETRIEVE patient $PATIENT;
                ASSIGN $SUCCESS_FLAG "Failure";
                WHILE : TEST (== $SUCCESS_FLAG "Failure")   {
               EXECUTE connectToAgentAsClient $PORT $HOST $IN $OUT;
                EXECUTE sendMessage $OUT $PATIENT;
                EXECUTE recieveMessage $IN $ANSWER;
                WHEN : TEST (== $ANSWER "Inform")   {
                  EXECUTE recieveMessage $IN $AMBULANCE_ID;
                  PERFORM waiting_for_ambulance $AMBULANCE_ID;
                  ASSIGN $SUCCESS_FLAG "True";  };  };     }
```

```
Plan: { NAME: "communication of participant"
        GOAL: PERFORM participant;
        BODY: EXECUTE connectToAgentAsServer $PORT $HOST $IN $OUT;
              EXECUTE recieveMessage $IN $PATIENT;
              ASSIGN $IDLE_FLAG "Failure";
              PERFORM check_idle_instance;
              WHEN : TEST (== $IDLE_FLAG "Inform")  {
                 PERFORM select_instance;
                 EXECUTE snedMessage $OUT "Inform";
                 EXECUTE snedMessage $OUT $AMBLUANCE_ID; };
              WHEN : TEST (== $IDLE_FLAG "Failure) {
                 EXECUTE snedMessage $OUT "Failure";  };  }
//...
```

JAM file of TRAFFICACCIDENT community template, "TRAFFICACCIDENT.jam "

Figure 3. An example of CIM-PS based on JAM[6] platform

4. CONCLUSION

In this article, we proposed a high-level conceptual model using community metaphor to support dynamic management of goal-oriented organization and cooperation in multi-agent based ubiquitous systems. Then, to fill a gap between a design and implementation, we propose the entire development process through model transformation from a high-level conceptual model to an implementation model. However, there are several issues remaining for future work.

- Capability verification of community computing model
- Policy of community computing model
- Dynamic structuring of community
- Dynamic cooperation of member agents
- Test on variety platforms

5. ACKNOWLEGEMENT

This research is supported by the ubiquitous Autonomic Computing and Network Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea

6. REFERENCES

[1] Jacques Ferber, *Multi-agent Systems*, Addison-Wesley, Reading, MA, 1999.

[2] Jennings, N. R., et. al. Transforming Standalone Expert Systems into a Community of Cooperating Agents, Int. Journal of Engineering Applications of Artificial Intelligence 6 (4), 2003, 317-331.

[3] Mohan Kumar, Behrooz A. Shirazi, Sajal K. Das, Byung Y. Sung, et. al. PICO: A Middleware Framework for Pervasive Computing, *Pervasive Computing*, 1536-1268, 2003, 72-79

[4] Object Management Group. Model Driven Architecture Guide, 2003

[5] Javier Munoz. Pervasive Systems Development with Model Driven Architecture, In *Proceedings of the 7th International Conference on the Unified Modeling Language (UML2004)* (Lisbon, Portugal, October 10, 2004).

[6] Huber M. j. JAM Agent in a Nutshell, version 0.65+0.76i, November 1, 2000

Enhancing Agent-Oriented Models with Aspects

Alessandro Garcia
Computing Dept, Lancaster Univ.
South Drive, InfoLab21
Lancaster, LA14WA, UK
garciaa@comp.lancs.ac.uk

Christina Chavez
Computing Dept, UFBA
Av. Adhemar de Barros, s/n
Salvador/BA, 40170-110, Brazil
flach@dcc.ufba.br

Ricardo Choren
SE/9, IME/RJ
Pça General Tibúrcio 80
Rio de Janeiro/RJ, 22290-270, Brazil
choren@de9.ime.eb.br

ABSTRACT

The modular representation of concerns is imperative in the design modeling of multi-agent systems (MASs). However, MAS designers typically face a number of concerns that inherently affect several system agents and their respective internal elements, such as actions and goals. These crosscutting concerns encompass both internal and systemic properties, such as learning, mobility, error handling, and security. Without an explicit modeling of such MAS properties, designers can not properly communicate and reason about them and their broadly-scoped effects. The result is a multitude of agent-oriented design breakdowns: poor traceability from requirements to design models, hindering of reuse opportunities, and decreased evolvability. This paper presents an enhancement of an existing agent-oriented modeling language with aspects. We illustrate the use of our approach through some classical examples of crosscutting concerns in agent-oriented design models.

Keywords

Agent-oriented software engineering, aspect-oriented software development, modeling languages, agent-oriented design.

1. INTRODUCTION

Several systemic and internal agent properties in the design of a multi-agent system (MAS) are typically *crosscutting concerns* [1, 5, 6, 8], such as learning, code mobility, context-awareness, error handling, and security. These concerns are MAS properties that involve and affect a consistent number of goals and actions, which are scattered and tangled to each other over an agent-oriented design. For example, the learning concern is composed of a number of specific goals and actions, which crosscut the goal hierarchies and actions' descriptions associated with several agents in an application. These crosscutting concerns often are fundamental driving factors in the software development process due to their broadly-scoped influence in the MAS design. Hence software engineers need to be empowered with modular representation of these concerns in order to properly communicate and reason about them and their crosscutting design implications.

Due to the very nature of some MAS concerns, existing agent-oriented modeling languages [2, 7, 9, 11] are not able to provide explicit support for the separation of crosscutting MAS-related concerns. As a result, their crosscutting effects are inevitably spanned over the resulting agent-oriented design artifacts. These languages provide the basis to separately represent several MAS functionalities. However, they are not aimed at supporting the handling of crosscutting MAS concerns as separate "aspects". Hence these concerns cannot be modularized with the abstractions and composition mechanisms available. In fact, the need for explicitly modeling crosscutting concerns seems evident in an age where we are facing a growing number of modeling extensions with the sole purpose of coping with a specific MAS concern, such as mobility, security, and autonomy. However, each of these approaches addresses a specific concern by extending a modeling language.

In this context, this paper presents aspect-oriented enhancements of an agent-oriented modeling language, called ANote[3], in order to cope with crosscutting concerns and promote MAS designs with superior modularity. The enhancements blind aspectual abstractions [2] with fundamental agent-oriented design elements, such as agents, goals and actions. We introduce a new notation element, called *aspect*, and additional composition mechanisms to promote improved separation of MAS concerns. An aspect is a first-class, nameable entity that provides modular representation for a crosscutting concern and localizes both (a) the specification of (sets of) join points, and (b) the enhancements to be combined at the specified join points. The enhancements may add new structure and behavior to agents, goals and actions, refine or replace existing behavior.

The paper is organized as follows. Section 2 illustrates some crosscutting concerns in agent-oriented modeling in terms of an example. Section 3 presents our proposed enhancements to ANote, and presents an aspectual notation compliant to this language. Section 4 presents concluding remarks and future work.

2. CROSSCUTTING CONCERNS IN AGENT-ORIENTED MODELING

Figure 1 shows a partial design representation of a multi-agent system [4] specified with the ANote modeling language [3]. ANote defines modeling views that allow for the expression of a MAS design from different perspectives. For instance, it includes an agent view (Figure 1a), a goal view (Figure 1b), and a scenario view (Figure 1c), which are recurring perspectives in several modeling languages [1]. They support the representation of agents, actions, and goals respectively. Modeling agent systems often includes other system concerns and views, such as interaction and organizational, which have been omitted from Figure 1 due to space limitation and simplicity purposes.

The system modeled in Figure 1 is a multi-agent application that supports the management of the reviewing process for research conferences. This system will be herein referred to as Expert Committee (EC). The EC system encompasses three types of

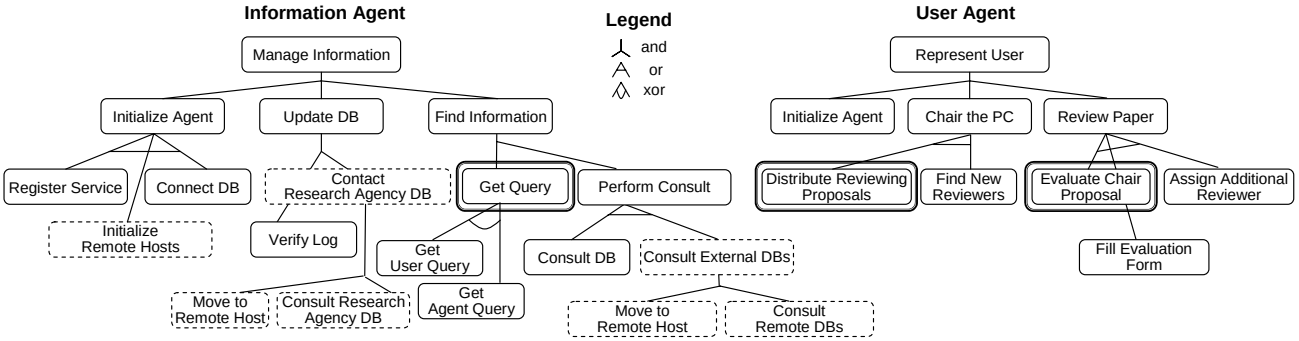
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

software agents (Figure 1a): (i) information agents, (ii) user agents, and (iii) the manager agent. Figures 1b and 1c illustrate some goals and actions associated with the information and user agents respectively. Information agents' goals are managing the system information that is mainly stored in a database, and providing information to the other system agents and users as requested. User agents are software assistants that represent system users in reviewing processes. Their basic functionalities are to infer and to keep information about the users' research interests and participations in scientific events.

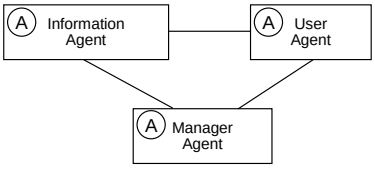
We can notice in Figure 1 that there are some concerns which cannot be modularly represented across the design views. Learning and mobility are two examples of crosscutting concerns in the agent-oriented design of the EC system. The set of goals and actions, which are both part of and affected by these broadly-scoped concerns, are spread over the agent-oriented design. Each set of goals and actions for a specific crosscutting concern is surrounded by dotted rectangles in Figure 1. The mobility concern consists of goals that crosscut distinct goal hierarchies, and of actions intermingled with diverse scenarios.

The crosscutting manifestation leads to two major problems at the agent-oriented design level, the so-called scattering and tangling [10]. *Scattering* in agent-oriented models is the manifestation of design elements, which belong to one specific concern, over several modeling units referred to other MAS concerns. *Tangling* in agent-oriented models is the mix of multiple concerns together in the same modeling elements

Existing agent-oriented modeling abstractions and composition mechanisms are not aimed at supporting the isolate handling of crosscutting MAS concerns as exemplified in Figure 1. It brings in turn a number of substantial design pitfalls, such as: hindering of modular and compositional reasoning; replication of agent-oriented design elements; essential information loosing; and, reduced evolvability and reuse opportunities



(b) Goal view diagram



(a) Agent view diagram

Goal	Perform Consult
Agent	Information Agent
Precondition	Local DB is available Remote environments are available
Actions	1. process the query 2. consult DB 3. gather consult details 4. decide whether moving or not 5. notify manager agent 6. get remote permissions to move 7. move to remote environments 8. consult external DBs 9. return home 10. return result 11. gather user feedback 12. process information using ML strategies 13. adapt user/agent preferences
Interactions	Manager Agent

Goal	Evaluate Chair Proposal
Agent	User Agent
Precondition	Chair sent the reviewing proposal
Actions	1. receive the proposal 2. process the proposal 3. propose a decision to the user 3. get confirmation from the user 4. gather decision and proposal details 5. process information using ML strategies 6. adapt user preferences 7. return final decision to the chair
Interactions	User Agent

(c) Scenario view diagram

Legend
elements affected by or part of mobility
elements affected by or part of learning

Figure 1. Crosscutting Concerns in ANote Modeling Views.

3. AN ASPECT-ORIENTED NOTATION

This section describes our approach to support agent-oriented modeling enhanced with aspects. We show the use of our notation in the context of the ANote modeling language [3]. ANote has 7 diagrams divided into three packages. In this paper, we will focus on the structural package, which encompasses the Goal, Agent and Planning diagrams.

The goal view diagram presents the system goals, i.e. the agent goals (Figure 2). This diagram has been enhanced with an aspect-oriented notation to present how the goals of aspects influence the agent goals at well-defined join points. The extension of the goal view diagram notation has been based on and adapted from our aspect-oriented notation for object-oriented design proposed in [2]. An aspect is represented as a diamond. Some circles are attached to it. Each circle captures a way (interface) that an aspect crosscuts the agent goals.

Figure 2 shows that the Mobility aspect has an interface "IMobility" that introduces the goals "Move to Remote Host" and "Consult Remote DB" to the goals "Consult Research Agency DB" and "Consult External DB" of Information Agents. In this

case, the composition operation "introduce" (Table 1) is used to denote an aspect-agent crosscutting.

The agent view specifies the agent types that exist in a multi-agent application solution and their relationships. In ANote, an agent is the module that is able to perform actions. However, aspects also contain a number of actions that implement the crosscutting behaviors, thus becoming system modules. Figure 3 shows how the agent diagram has been enhanced with aspect-oriented notation. Here, the aspects are also represented as diamonds. The arrows represent the crosscutting relationship. We can see in Figure 3 that Information Agents and User Agents are influenced by the Learning aspect, while the Information Agent is also affected by the Mobility aspect.

The ANote dynamic package encompasses the scenario, interaction and planning diagrams that have also been enhanced with aspects, but for brevity reasons, are not presented here.

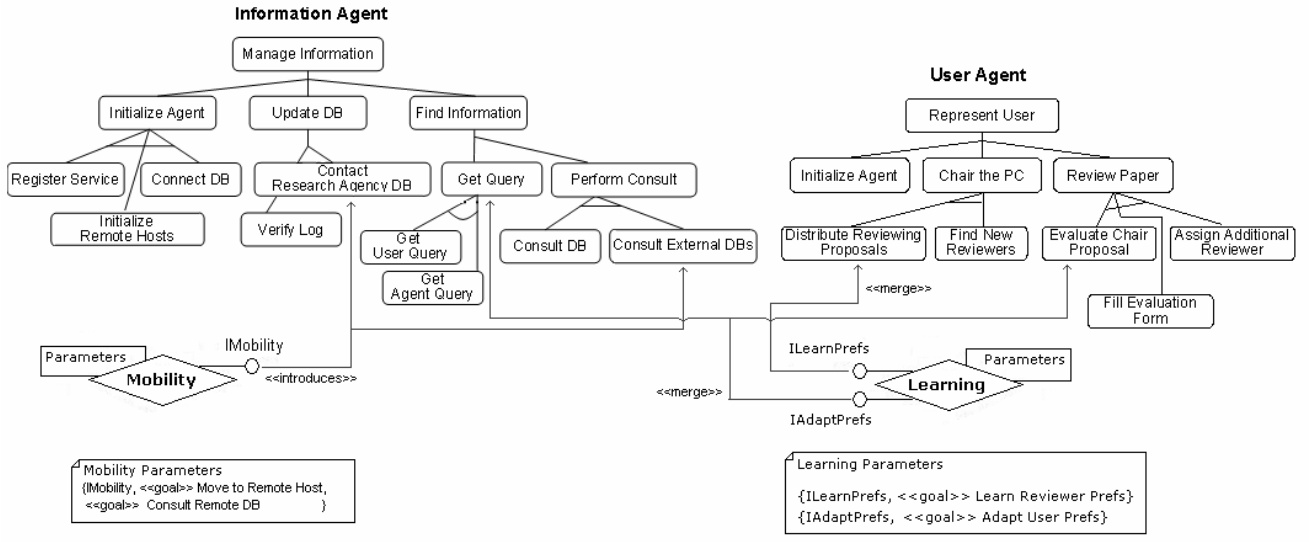


Figure 2. Aspect enhanced goal diagram.

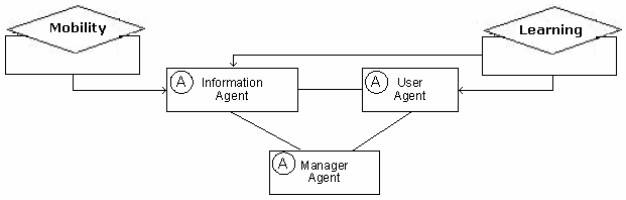


Figure 3. Aspect enhanced agent diagram.

4. CONCLUSIONS AND FUTURE WORK

This paper is a first attempt to systematically tame broadly-scoped concerns in agent-oriented design modeling. Many internal agent properties and systemic properties in the design of MAS are typically crosscutting, and they need to be handled as such. No matter what kind of decomposition and abstractions the agent-based software developers are relying on; there are always MAS concerns that crosscut the boundaries of other concerns. The lack of modularization support for crosscutting concerns generates undesirable burdens on agent-oriented design reuse and evolution.

This paper presented an aspect-oriented notation and integrated it into the ANote language. Our work is a first step towards enhancing agent-oriented models with aspects and we cannot guarantee that our set of diagrams is necessarily complete; more case studies are still necessary. Another issue is the behavioral modeling of agents, i.e. how to model crosscutting actions. We are currently working on enhancing the ANote planning view diagram to properly address other action-action composition categories.

5. REFERENCES

[1] Bergenti, F., Gleizes, M.-P., Zambonelli, F. (Eds.). Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook, volume 11. Springer-Verlag, 2004.

[2] Chavez, C., Lucena, C. A Metamodel for Aspect-Oriented Modeling. Proc. Workshop on Aspect-oriented Modeling with UML, 1st Intl Conf. on Aspect-Oriented Software Development, 2002.

[3] Choren, R., Lucena, C. Modeling Multi-agent Systems with ANote. Software Systems Modeling Journal 4, 2005, p. 199-208.

[4] Garcia, A. From Objects to Agents: An Aspect-Oriented Approach. PhD Thesis, Computer Science Dept., PUC-Rio, Brazil, April 2004.

[5] Garcia, A., Lucena, C., Cowan, D. Agents in Object-Oriented Software Engineering. Software: Pract and Experience 34(3), 2004, p. 489-521.

[6] Garcia, A. et al. Aspects in Agent-Oriented Software Engineering: Lessons Learned. Proc. 6th Workshop on Agent-Oriented on Software Engineering, AAMAS 2005, July 2005.

[7] Odell, J., Parunak, H., Bauer, B. Extending UML for Agents. In: Proceedings of the Agent-Oriented Information Systems Workshop, 17th Conf. on Artificial Intelligence (AAAI 2000), p. 3–17, 2000.

[8] Pace, A., Trilnik, F., Campo, M. Assisting the Development of Aspect-based MAS using the SmartWeaver Approach. In: Software Engineering for MASs, LNCS 2603, Springer, p. 165-181, 2003.

[9] Silva, V., Lucena, C. From a Conceptual Framework for Agents and Objects to a Multi-Agent System Modeling Language. Journal of Autonomous Agents and Multi-Agent Systems 9(1-2), 2004.

[10] Tarr, P. et al. N Degrees of Separation: Multi-Dimensional Separation of Concerns. Proc. 21st ICSE 1999, p. 107-119, 1999.

[11] Wooldridge, M., Jennings, N., Kinny, D. The GaiaMethodology for Agent-oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems 3(3), 2000, p. 285–312.

Agent communication

On the Semantics of Conditional Commitment

Shakil M. Khan
Department of Computer Science and Engineering
Toronto, ON, Canada M3J 1P3
skhan@cs.yorku.ca

Yves Lespérance
Department of Computer Science and Engineering
Toronto, ON, Canada M3J 1P3
lesperan@cs.yorku.ca

ABSTRACT

In this paper, we identify some problems with current formalizations of conditional commitments, i.e. commitments to achieve a goal if some condition becomes true. We present a solution to these problems. We also formalize two types of communicative actions that can be used by an agent to request another agent to achieve a goal or perform an action provided that some condition becomes true. Our account is set within ECASL [9], a framework for modeling communicating agents based on the situation calculus.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Theory, Design

Keywords

Agent communication, Semantics of requests, Conditional intentions, Conditional commitments, Logics for agent systems, Formal models of agency

1. INTRODUCTION

In recent years, the importance of agent communication in multiagent systems has been widely recognized. As a result, many researchers have developed communicative multiagent frameworks [3, 5, 7, 15, 18, 23, 25] and attempted to formalize various types of communicative actions in these frameworks. One important concept in these is the notion of *conditional commitment*. A conditional commitment is a commitment to achieve some goal if some condition becomes true (e.g. a commitment to ship some goods when payment of an agreed to amount arrives). Conditional requests are requests that seek to have the addressee acquire a conditional commitment. Any multiagent framework that deals with negotiation and cooperation ought to handle conditional commitments. Unfortunately, most definitions found in the literature (in [5, 28, 23, 1], for example) are inadequate: they either define conditional commitments as disjunctive goals, which makes the agents under-committed to the conditional goal, or define them as conjunctive goals, which renders the agent over-committed.

We will go over some examples to point out the problems associated with the disjunctive and the conjunctive accounts of conditional commitment. In these, we use the following modal operators: $\Diamond\phi$, i.e. ϕ eventually holds, Happens(α), i.e. the action α is performed next, ϕ Until ψ, i.e. eventually ψ becomes true, and as long as ψ is false, ϕ holds, and Before(ψ, ϕ), i.e. if ψ eventually becomes true, then ϕ becomes true before ψ does. The formal semantics of these operators are given in Section 2. In the following, we formulate these examples using internal/mental states semantics for communication acts, and use the terms 'intention' and 'commitment' interchangeably.

In the *disjunctive account*, a conditional commitment to achieve some goal provided that some condition holds is modeled as a commitment to achieve the goal if the condition holds, i.e. as a simple material implication. For example, consider an online marketplace domain. Suppose that there are two agents, a seller agent slr, and a buyer agent byr. If we use a disjunctive account, slr's conditional commitment to ship some goods to byr on the condition that byr pays can be modeled as follows:

$$\begin{aligned}&\text{CondInt}_{\text{dis}}(slr, GetPaid, \text{Happens}(shipGoods(slr, byr)))\\ &\doteq \text{Int}(slr, \neg\Diamond GetPaid \vee\\ &\qquad [\neg GetPaid \text{ Until}\\ &\qquad\quad (GetPaid \wedge \text{Happens}(shipGoods(slr, byr)))]).\end{aligned}$$

This says that slr's conditional commitment to ship the goods when byr pays amounts to slr having the intention that byr eventually pays and after that she ships the goods, if byr eventually pays (as mentioned earlier, the Until construct in the goal above implies that $\Diamond GetPaid$ and $\Diamond$-Happens($shipGoods(slr, byr)$)). One problem with this account of conditional intention is that there is a counter-intuitive way to satisfy the conditional intention, namely, the agent may commit to the triggering condition remaining false and deliberately perform some action that makes it remain false. Thus, in the example, to satisfy her conditional intention, slr may intentionally perform some action to stop byr from paying her, such as blocking debits from byr. In other words, there is nothing in this formalization of conditional intention that stops slr from intending not to

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

get paid and not to send the goods. However, this is counter-intuitive and a model of conditional commitment should not support this. Thus, with the disjunctive account of conditional commitment, the agent seems under-committed to the goal. Examples of accounts in the literature that formalize conditional commitments as disjunctive goals are [23] and [1].

In the *conjunctive account*, a conditional commitment to achieve a goal provided that a condition holds is modeled as a temporally ordered conjunctive commitment to the triggering condition and the conditional goal, where the triggering condition is achieved first. Although this model may seem appropriate in many cases, it often leads to problems. For example, suppose that *slr* has the conditional commitment to ship a replacement unit provided that *byr* reports and returns a defective good. If we use a conjunctive account, this can be modeled as follows:

$$\begin{aligned}&\text{CondInt}_{\text{con}}(slr, DefGoodRet,\\ &\qquad\qquad \text{Happens}(shipRepl(slr, byr))) \doteq\\ &\text{Int}(slr, \text{Before}(\text{Happens}(shipRepl(slr, byr)), DefGoodRet)\\ &\qquad \wedge \Diamond \text{Happens}(shipRepl(slr, byr))).\end{aligned}$$

This says that *slr*'s conditional commitment to ship a replacement unit provided that *byr* returns a defective good can be modeled as *slr*'s intention that *byr* returns a defective good before *slr* ships a replacement unit, and eventually *slr* ships a replacement unit. Note that, according to this definition, since *slr* has the intention that the defective product is returned before she ships the replacement unit, and that she eventually ships the replacement unit, it follows that *slr* has the intention that *byr* eventually returns a product, i.e. $\text{Int}(slr, \Diamond DefGoodRet)$. So *slr* may deliberately perform some action, such as shipping a defective good in the first place, to achieve this intention. Thus, the conjunctive account of conditional commitment results in over-committed agents. Both [5] and [28] formalize conditional commitments as conjunctive goals.

In this paper, we propose a solution to these problems (the *under/over-commitment problems*, henceforth). Our solution involves using an additional constraint with the disjunctive account to eliminate the under-commitment problem. We use the Extended Cognitive Agent Specification Language (ECASL) [9] as our base formalism for this. Our account is formulated for internal/mental states semantics for communication acts. Nevertheless, the same issues arise for public/social-commitment semantics (as discussed in Section 5). Once again, in this paper, we will use the terms 'intention' and 'commitment' interchangeably.

Using our definition of conditional intention, we then prove that a conditional intention becomes an ordinary intention when the associated condition is true, and that agents correctly introspect about their conditional intentions. We also formalize two types of communicative actions that can be used by an agent to request another agent to achieve a goal or perform an action provided that some confdition becomes true. Then we prove that performing such a conditional request leads to a conditional intention under the right conditions.

The paper is organized as follows: in the next section, we outline the ECASL framework. In Section 3, we present our model of conditional commitment and discuss some of its properties. In Section 4, we present some communicative acts that allow agents to make requests that result in conditional commitments. In Section 5, we compare our approach to previous work on conditional commitments. Finally in Section 6, we summarize our results and discuss possible future work.

2. ECASL

The Extended Cognitive Agent Specification Language (ECASL) [9], an extension of CASL [22, 24], is a framework for specifying and verifying complex communicating multi-agent systems that incorporates a formal model of means-ends reasoning. In this section, we outline the part of ECASL that is needed for our formalization of conditional commitment.

In ECASL, agents are viewed as entities with mental states, i.e., knowledge and goals, and the specifier can define the behavior of the agents in terms of these mental states. ECASL combines a declarative action theory defined in the situation calculus with a rich programming/process language, ConGolog [4]. Domain dynamics and agents' mental states are specified declaratively in the theory, while the agents' behavior is specified procedurally in ConGolog.

In ECASL, a dynamic domain is represented using an action theory [17] formulated in the situation calculus [12], a (mostly) first order language for representing dynamically changing worlds in which all changes are the result of named actions. ECASL uses a theory D that includes the following set of axioms:

- action precondition axioms, one per action a characterizing $Poss(a, s)$,
- successor state axioms (SSA), one per fluent, that encode both effect and frame axioms and specify exactly when the fluent changes [16],
- initial state axioms describing what is true initially including the mental states of the agents,
- axioms identifying the agent of each action,
- unique name axioms for actions, and
- domain-independent foundational axioms describing the structure of situations [10].

Within ECASL, the behavior of agents is specified using the notation of the logic programming language ConGolog [4], the concurrent version of Golog [11]. A typical ConGolog program is composed of a sequence of procedure declarations, followed by a complex action. Complex actions can be composed using constructs that include primitive actions, waiting for a condition, sequence, nondeterministic branch, nondeterministic choice of arguments, conditional branching, while loop, procedure call, nondeterministic iteration, concurrent execution with and without priorities, and interrupts. To deal with multiagent processes, primitive actions in ECASL take the agent of the action as argument.

The semantics of the ConGolog process description language is defined in terms of *transitions*, in the style of structural operational semantics [14]. The overall semantics of a program is specified by the *Do* relation:

$$Do(\delta, s, s') \doteq \exists \delta' \cdot (Trans^*(\delta, s, \delta', s') \wedge Final(\delta', s')).$$

$Do(\delta, s, s')$ holds if and only if s' can be reached by performing a sequence of transitions starting with program δ in s, and the remaining program δ' may legally terminate

in s'. Here, $Trans^*$ is the reflexive transitive closure of the transition relation $Trans$.[1]

The situation calculus underlying ECASL is a branching time temporal logic, where each situation has a linear past and a branching future. In the framework, one can write both state formulas and path formulas. A state formula $\phi(s)$ takes a single situation as argument and is evaluated with respect to that situation. On the other hand, a path formula $\psi(s_1, s_2)$ takes two situations as arguments and is evaluated with respect to the interval (finite path) $[s_1, s_2]$. A state formula ϕ may contain a placeholder constant *now* that stands for the situation in which ϕ must hold. $\phi(s)$ is the formula that results from replacing *now* by s. Similarly, a path formula ψ may contain the placeholder constants *now* and *then* that stand for the situations that are the endpoints of the interval $[now, then]$ over which ψ must hold. $\psi(s_1, s_2)$ denotes ψ with s_1 substituted for *now* and s_2 substituted for *then*. Where the intended meaning is clear, we sometimes suppress the placeholder(s).

ECASL allows the specifier to model agents in terms of their mental states by including operators to specify agents' information (i.e., their knowledge), and motivation (i.e., their goals or intentions). We use state formulas within the scope of knowledge, and path formulas within the scope of intentions. Following [13, 20], ECASL models knowledge using a possible worlds account adapted to the situation calculus. $K(agt, s', s)$ is used to denote that in situation s, *agt* thinks that she could be in situation s'. s' is called a *K-alternative situation* for *agt* in s. Using K, the knowledge or belief of an agent, $\text{Know}(agt, \phi, s)$, is defined as $\forall s'(K(agt, s', s) \supset \phi(s'))$, i.e. *agt* knows ϕ in s if ϕ holds in all of *agt*'s K-accessible situations in s. In ECASL, K is constrained to be reflexive, transitive, and Euclidean in the initial situation to capture the fact that agents' knowledge is true, and that agents have positive and negative introspection. As shown in [20], these constraints then continue to hold after any sequence of actions since they are preserved by the successor state axiom for K.

ECASL supports knowledge expansion as a result of sensing actions [20] and some *informing* communicative actions. Here, we restrict our discussion to knowledge expansion as a result of *inform* actions. The preconditions of *inform* are as follows:

$$Poss(inform(inf, agt, \phi), s) \equiv \text{Know}(inf, \phi, s) \\ \wedge \neg\text{Know}(inf, \text{Know}(agt, \phi, now), s).$$

In other words, the agent *inf* can inform *agt* that ϕ, iff *inf* knows that ϕ currently holds, and does not believe that *agt* currently knows that ϕ. The successor state axiom (SSA) for K can be defined as follows:

$$K(agt, s^*, do(a, s)) \equiv \\ \exists s'.\ K(agt, s', s) \wedge s^* = do(a, s') \wedge Poss(a, s').$$

This says that after an action happens, every agent learns that it was possible and has happened. Moreover, if the action involves someone informing *agt* that ϕ holds, then *agt* knows this afterwards. This follows from the fact that it is a precondition of $inform(inf, agt, \phi)$ that *inf* knows that ϕ, that what is known must be true (i.e. K is reflexive), and that the SSA for K requires the agent to know that $Poss(a, s)$ after a happens in s. Note that this axiom only handles knowledge expansion, not revision.

ECASL also incorporates goal expansion and a limited form of goal contraction. Goals or intentions are modeled using an accessibility relation W over possible situations. The W-accessible situations for an agent are the ones where she thinks that all her goals are satisfied. W-accessible situations may include situations that the agent thinks are impossible (i.e. that do not have a predecessor that is K-related to the current situation), unlike Cohen and Levesque's [2] G-accessible worlds. But intentions are defined in terms of the more primitive W and K relations so that the intention accessible situations are W-accessible situations that are also compatible with what the agent knows, in the sense that there is a K-accessible situation in their history. This guarantees that agents' intentions are realistic, that is, agents can only intend things that they believe are possible. Thus we have:

$$\text{Int}(agt, \psi, s) \doteq \forall s', s^*. \\ [W(agt, s^*, s) \wedge K(agt, s', s) \wedge s' \leq s^*] \supset \psi(s', s^*).$$

This means that the intentions of an agent in s are those formulas that are true for all intervals between situations s' and s^* where the situations s^* are W-accessible from s and have a K-accessible situation s' in their past. Intentions are future oriented, and any goal formula will be evaluated with respect to a finite path defined by a pair of situations, a current situation *now* and an ending situation *then*. This formalization of goals can deal with both achievement goals and maintenance goals. An achievement goal ϕ is said to be eventually satisfied if ϕ holds in some situation between *now* and *then*, i.e., if $\Diamond(\phi, now, then)$, which is defined as $\exists s'.\ (now \leq s' \leq then \wedge \phi(s'))$.[2] In [21], Shapiro showed how positive and negative introspection of intentions can be modeled by placing some constraints on K and W. To make sure that agents' wishes and intentions are consistent, W is also constrained to be serial.

ECASL provides an intention transfer communication action, *request*, which is defined in terms of *inform*. This is somewhat similar to Herzig and Longin's account [7], where a request is defined as informing about one's intentions, and the requested goals are adopted via cooperation principles. The *request* action can be used by an agent to request another agent to achieve some state of affairs. Formally, we have:

$$request(req, agt, \phi) \doteq inform(req, agt, \text{Int}(req, \phi, now)).$$

The SSA for W which handles intention change in ECASL, has the same structure as a SSA for a domain dependent fluent. In the following, $W^+(agt, a, s^*, s)$ ($W^-(agt, a, s^*, s)$, resp.) denotes the conditions under which s^* is added to (dropped from, resp.) W as a result of the action a in s:

$$W(agt, s^*, do(a, s)) \equiv W^+(agt, a, s^*, s) \vee \\ (W(agt, s^*, s) \wedge \neg W^-(agt, a, s^*, s)).$$

An agent's intentions are expanded when it is requested

[1]Since we have predicates that take programs as arguments, we need to encode programs and formulas as first-order terms as in [4]. For notational simplicity, we suppress this encoding and use formulas and programs as terms directly.

[2]We sometimes use $\Diamond$ with a path formula ψ argument, in which case, we mean that ψ holds over some interval $[s, then]$ that starts at some situation s between *now* and *then*; see Table 1 for the formal definition.

something by another agent. After the $request(req, agt, \psi)$ action, *agt* adopts the goal that ψ, unless she has a conflicting goal or is not willing to serve *req* for ψ. Therefore, this action should cause *agt* to drop any paths in W where ψ does not hold. This is handled in W^-:

$$\begin{aligned} &W^-(agt, a, s^*, s) \doteq \\ &\quad [\exists req, \psi.\ a = request(req, agt, \psi) \\ &\quad \wedge Serves(agt, req, \psi, s) \wedge \neg \mathrm{Int}(agt, \neg\psi, s) \\ &\quad \wedge \exists s'.\ K(agt, s', s) \wedge s' \leq s^* \wedge \neg\psi(do(a, s'), s^*)]. \end{aligned}$$

A limited form of intention contraction is also handled in ECASL. Agents intentions are contracted as a result of a *cancelRequest* action. ECASL also incorporates a formal model of means-ends reasoning and commitment to rational plans to achieve intentions. See [9] for the details of these.

Table 1 shows some abbreviations that will be used throughout the paper.

Table 1: Some Definitions of Temporal Operators

1. $\Diamond(\psi, now, then) \doteq \exists s'.\ now \leq s' \leq then \wedge \psi(s', then)$,
2. $\Box(\psi, now, then) \doteq \neg\Diamond(\neg\psi, now, then)$,
3. $[\phi \text{ Until } \psi](now, then) \doteq \exists s'.\ now \leq s' \leq then \wedge \psi(s', then) \wedge \forall s''.\ now \leq s'' < s' \supset \phi(s'')$,
4. $\mathrm{Before}(\psi, \phi, now, then) \doteq \exists s'.\ now \leq s' \leq then \wedge \psi(s', then) \supset \exists s''.\ now \leq s'' < s' \wedge \phi(s'')$,
5. $\mathrm{E}_\Diamond(\phi, now) \doteq \exists s.\ now \leq s \wedge \phi(s)$,
6. $\mathrm{A}_\Box(\phi, now) \doteq \neg\mathrm{E}_\Diamond(\neg\phi, now)$,
7. $\mathrm{Happens}(a, now, then) \doteq do(a, now) \leq then$,
8. $\mathrm{Happens}_G(\delta, now, then) \doteq \exists s'.\ s' \leq then \wedge Do(\delta, s', then)$.

3. CONDITIONAL COMMITMENTS

Having presented our framework, we now return to our discussion about conditional commitments. Informally, an agent *agt* has a conditional commitment or intention that ψ on the condition that ϕ if *agt* intends to achieve ψ as soon as the condition ϕ holds. In our specification, we assume that ϕ is a state formula, whereas ψ is a path formula and can represent any kind of goal (achievement, maintenance, etc.). In other words, the trigger condition ϕ of a conditional intention takes a single situation *now* as argument, unlike the goal formula ψ, which takes two situations *now* and *then* as arguments.[3] If one wishes to use an achievement goal ϕ' for ψ, one can use $\Diamond(\phi', now, then)$, i.e. eventually ϕ'.

So we now propose a formalization of conditional intentions that avoids the under/over-commitment problem:

$$\begin{aligned} &\mathrm{CondInt}(agt, \phi, \psi, s) \doteq \\ &\quad \mathrm{Int}(agt, DisjGoal(\phi, \psi, now, then) \\ &\qquad \wedge NoUnderComm(agt, \phi, now, then), s), \\ &DisjGoal(\phi, \psi, now, then) \doteq \\ &\quad [\neg\phi \text{ Until } (\phi \wedge \psi)](now, then) \vee \neg\Diamond(\phi, now, then), \\ &NoUnderComm(agt, \phi, now, then) \doteq \\ &\quad \Box([\mathrm{Int}(agt, \Box(\neg\phi, now, then), now) \supset \\ &\qquad \mathrm{Know}(agt, \mathrm{A}_\Box(\neg\phi, now), now)], now, then). \end{aligned}$$

That is, *agt* conditionally intends that ψ provided that ϕ, iff *agt* intends that the following conditions hold:

1. either (*a*) ϕ eventually holds, and ψ holds immediately from the time ϕ comes to hold, or (*b*) ϕ never holds, and
2. if in any situation *agt* intends that ϕ never comes to hold, she must also know in that situation that it can never become true.

Intuitively, this says that one way to fulfill an agent's conditional intention is to (1*a*) satisfy ψ after ϕ comes to hold, and a second way is that (1*b*) ϕ never comes to hold in the future. This part of our account is as in the disjunctive approach. However, we add to this that (2) the agent does not intend that ϕ never comes to hold unless she knows that it can never hold. Thus we require that if at some situation, *agt* intends that ϕ never comes true, it must be the case that she knows in that situation that ϕ can never become true, and she only intends this because it has become inevitable. So the additional constraint that $NoUnderComm(agt, \phi, now, then)$ ensures that *agt* will not do anything intentionally to make the triggering condition ϕ remain false. One might be tempted to define $NoUnderComm(agt, \phi, now, then)$ as $\Box(\neg\mathrm{Int}(agt, \Box(\neg\phi, now, then), now), now, then)$, i.e. *agt* never intends that ϕ never holds. However, since some event may make ϕ impossible to achieve, there is a possibility that *agt* may come to intend that ϕ always be false, if this becomes inevitable. The only case in which *agt* intends that ϕ always be false is when she knows that it can never become true.

Consider once again our online marketplace example given in Section 1 for the disjunctive account. Using this definition of conditional commitment, a seller *slr*'s intention to send the goods when a buyer *byr* pays, $\mathrm{CondInt}(slr, GetPaid(byr, slr), \mathrm{Happens}(shipGoods(slr, byr), now, then), s)$ can be formalized as follows:

$$\begin{aligned} &\mathrm{Int}(slr, DisjGoal(GetPaid(byr, slr), \\ &\qquad \mathrm{Happens}(shipGoods(slr, byr), now, then), \\ &\quad now, then) \\ &\wedge NoUnderComm(slr, GetPaid(byr, slr), now, then), s). \end{aligned}$$

slr's intention can be further expanded to:

$$\begin{aligned} &\mathrm{Int}(slr, [GetPaidAndThenSendGoods(byr, slr, now, then) \\ &\qquad \vee \neg\Diamond(GetPaid(byr, slr), now, then)] \wedge \\ &\quad [\Box((\mathrm{Int}(slr, \Box\neg GetPaid(byr, slr), now) \supset \\ &\qquad \mathrm{Know}(slr, \mathrm{A}_\Box(\neg GetPaid(byr, slr), now), now)), \\ &\quad now, then)], s), \end{aligned}$$

[3] We could also handle trigger conditions that are not state formulas. However, in these cases, since the trigger condition holds over a time interval, it is not always clear when exactly the triggering of the commitment to the conditional goal should occur. To avoid these complications, we stick to state formulas as triggers.

where,

$$GetPaidAndThenSendGoods(byr, slr, now, then) \doteq$$
$$[\neg GetPaid(byr, slr) \text{ Until}$$
$$(GetPaid(byr, slr) \wedge$$
$$\text{Happens}(shipGoods(slr, byr), now, then))]$$
$$(now, then).$$

From this, we can see that there are only two ways by which *slr* can satisfy this conditional intention: either at some future or current situation *byr* pays *slr* and then *slr* sends the goods to *byr*, or, *byr* never pays *slr*, and as long as *slr* does not know that *byr* will never pay her, she does not intend not to get paid. Since *slr* cannot intend not to get paid, she cannot deliberately perform anything (for example block payments from *byr*) to make the triggering condition remain false. If at a later situation, *slr* learns that it has become impossible for *byr* to ever pay her, *slr* will inevitably intend that *byr* never pays her, but otherwise she cannot acquire this intention. Thus, our formalization of conditional commitment does not suffer from the under-commitment problem.

Moreover, since we use the disjunctive approach, our account does not suffer from the over-commitment problem associated with the conjunctive approach. Consider the second example given in Section 1, where *slr* has the intention to ship a replacement unit when *byr* returns a defective good. Using our definition, this can be expanded to *slr*'s intention that either *byr* never returns a defective product, or *byr* returns a defective product and *slr* ships the replacement unit after that. Thus *slr* is not over-committed and will not perform something deliberately so that *byr* returns a product. The additional constraint that *slr* never intends that *byr* never return a product unless she knows that *byr* will never return a product does not lead to any over-commitment. Thus our formalization of conditional intention is also free from the over-commitment problem.

Note that our account allows the agent who has a conditional intention to intend not to know whether the condition holds. We could easily strengthen the definition to rule this out, but it is not clear that this is always appropriate.

Next, we show two simple properties of conditional intention. Assume that the domain theory D (as discussed in Section 2) includes our definition of conditional commitment given above. Then we have the following theorem that says that if an agent *agt* conditionally intends that ψ provided that ϕ in situation s, and if she knows that ϕ holds in s, then *agt* intends that ψ in s.

THEOREM 1.

$$D \models \text{CondInt}(agt, \phi, \psi, s) \wedge \text{Know}(agt, \phi, s) \supset$$
$$\text{Int}(agt, \psi, s).$$

So when the agent knows that the condition has become true, a conditional intention reduces to an ordinary intention. The second property states that agents are able to introspect their conditional intentions:

THEOREM 2.

$$D \models [\text{CondInt}(agt, \phi, \psi, s) \supset$$
$$\text{Know}(agt, \text{CondInt}(agt, \phi, \psi, now), s)] \wedge$$
$$[\neg\text{CondInt}(agt, \phi, \psi, s) \supset$$
$$\text{Know}(agt, \neg\text{CondInt}(agt, \phi, \psi, now), s)].$$

Thus, if an agent has a conditional intention (does not have a conditional intention, resp.) that ψ provided that ϕ, then she knows that she has (does not have, resp.) this conditional intention.

It would be interesting to prove additional results about conditional intentions, for instance, that a conditional intention persists as long as its condition is known to remain false and not known to have become impossible. We leave this for future work.

4. CONDITIONAL REQUESTS

We now discuss two communicative acts, *requestWhen* and *reqActWhen*, that can be used by an agent to request someone to achieve ψ or to execute a program δ respectively, on the condition that ϕ becomes true. Recall that, in ECASL the SSA for W determines whether an agent adopts a goal when requested; the requested goal is adopted by the requestee via cooperation principles encoded in the SSA for W. Thus, we model requests as informing about intentions, rather than as primitives. In the following, we use $CondIntCont(agt, \phi, \psi)$ as an abbreviation for the content of a conditional intention $DisjGoal(\phi, \psi, now, then) \wedge NoUnderComm(agt, \phi, now, then)$. Now, one simple way to model a requester *req*'s request to requestee *agt* to achieve ψ on the condition that ϕ is as follows:

$$requestWhen_{sim}(req, agt, \phi, \psi) \doteq$$
$$request(req, agt, CondIntCont(req, \phi, \psi)).$$

This says that, *req*'s conditional request to *agt* to achieve ψ provided that ϕ amounts to *req*'s request to *agt* to fulfill the content $CondIntCont(req, \phi, \psi)$ of her own conditional intention. Using the definition of *request*, this conditional request amounts to *req* informing *agt* that she currently intends to achieve ψ provided that ϕ. However, note that the content $CondIntCont(req, \phi, \psi)$ of this conditional intention includes mental attitudes that refer to *req*, rather than *agt*. Since the SSA for W does not automatically replace the agent parameters of mental state operators used in a goal formula, if we model conditional requests as above, given appropriate conditions (i.e., when *agt* agrees to serve *req* on $CondIntCont(req, \phi, \psi)$ and does not currently have the intention that $\neg CondIntCont(req, \phi, \psi)$), *agt* will adopt the intention that $CondIntCont(req, \phi, \psi)$, but not that $CondIntCont(agt, \phi, \psi)$. Thus she will not have the conditional intention to achieve ψ provided that ϕ after the conditional request is performed, and this simple definition is not quite correct.

For example, suppose that the manager agent *mgr* wants to conditionally request the seller *slr* in situation s to ship the goods when the buyer *byr* pays her. So *mgr* can do this by performing the following action in s:

$$requestWhen_{sim}(mgr, slr, GetPaid(byr, slr),$$
$$\text{Happens}(shipGoods(slr, byr), now, then)),$$

which can be expanded to:

$$request(mgr, slr,$$
$$CondIntCont(mgr, GetPaid(byr, slr),$$
$$\text{Happens}(shipGoods(slr, byr), now, then))).$$

After the request is performed, if *slr* agrees to serve *mgr* on $CondIntCont(mgr, GetPaid(byr, slr)$, Happens (*ship-*

$Goods(slr, byr), now, then))$, and does not intend that $\neg CondIntCont(mgr, GetPaid(byr, slr), \text{Happens}(shipGoods(slr, byr), now, then))$, the SSA for W will ensure that:

$$\text{Int}(slr, CondIntCont(mgr, GetPaid(byr, slr), \text{Happens}(shipGoods(slr, byr), now, then)), s_r),$$

which can be expanded to:

$$\text{Int}(slr, DisjGoal(GetPaid(byr, slr), \text{Happens}(shipGoods(slr, byr), now, then), now, then) \wedge NoUnderComm(mgr, GetPaid(byr, slr), now, then), s_r),$$

where s_r is the situation that results from performing the $requestWhen$ action in s. Now, using the definition of conditional intention, we can see that in s_r, slr does not have the conditional intention of sending the goods provided that byr pays her. The problem is with the mental state operators in the $NoUnderComm(mgr, \ldots)$ part of slr's intention: they say that mgr will not intend that the payment not occur unless she knows it can never occur. What we need is for this constraint to hold for slr.

To deal with this problem, we propose the following model of conditional requests:

$$requestWhen(req, agt, \phi, \psi) \doteq request(req, agt, CondIntCont(agt, \phi, \psi)).$$

This says that req's request to agt to conditionally achieve ψ provided that ϕ amounts to req's request to agt to fulfill the content of agt's conditional intention to achieve ψ provided that ϕ, i.e., $CondIntCont(agt, \phi, \psi)$. Using the definition of *request*, this can be further expanded to:

$$requestWhen(req, agt, \phi, \psi) \doteq inform(req, agt, \text{Int}(req, CondIntCont(agt, \phi, \psi), now)).$$

That is, req can request agt to achieve ψ on the condition that ϕ by informing agt that she intends that $CondIntCont(agt, \phi, \psi)$. Note that, the agent parameter of $CondIntCont(agt, \phi, \psi)$ is now the requestee agt, rather than the requester req. This guarantees that given that agt serves req and does not have the opposite intention, she will conditionally intend to achieve ψ provided that ϕ after req conditionally requests her this. Thus this formalization of conditional request does not suffer from the above mentioned problem.

We also define a special type of conditional request, namely, a request to perform an action when some condition holds:

$$reqActWhen(req, agt, \phi, \delta) \doteq requestWhen(req, agt, \phi, Happens_C(\delta, now, then)).$$

This states that req's conditional request to agt to execute the program δ provided that ϕ amounts to req's conditional request to agt to execute δ starting in the situation where ϕ holds.

Now consider what happens when mgr conditionally requests slr to ship the goods when byr pays her, that is, when mgr performs the $reqActWhen(mgr, slr, GetPaid(byr, slr), shipGoods(slr, byr))$ action. Given that slr agrees to serve mgr and does not have the opposite intention, the SSA for W will make slr adopt the following intention:

$$\text{Int}(slr, CondIntCont(slr, GetPaid(byr, slr), \text{Happen}_C(sendGoods(slr, byr), when, then)), s_r),$$

and thus, by the definition of conditional intention, she will conditionally intend to send the goods when byr pays her. Thus, our formalization of conditional requests allows the proper transfer of conditional intention from the requester to the requestee.

We next present a theorem that shows how agents' intentions are affected by the *requestWhen* action. Assume that the domain theory D includes our definition of these new communicative actions. We can show that:

THEOREM 3.

$$D \models [\neg\text{Int}(agt, \neg CondIntCont(agt, \phi, \psi), s) \wedge Serves(agt, req, CondIntCont(agt, \phi, \psi), s) \wedge Poss(requestWhen(req, agt, \phi, \psi), s)] \supset \text{CondInt}(agt, \phi, \psi, do(requestWhen(req, agt, \phi, \psi), s)).$$

This says that if in some situation s, an agent agt does not intend not to fulfill the content of a conditional intention to achieve ψ provided that ϕ, and if she serves another agent req on the content of this conditional commitment in s, then she will have the conditional intention to achieve ψ given that ϕ after req conditionally request her this in s, provided that the request is possible in s.

It would be useful to extend our framework with a communication act that allows a conditional commitment created as a result of a *requestWhen* to be cancelled. We believe that the existing ECASL *cancelRequest* action can be used to define such a conditional commitment cancelling act. We leave this for future work.

5. RELATED WORK

The under-commitment problem that we pointed out in Section 1 is related to another problem involving intentions discussed by Cohen and Levesque [2]. In that paper, they consider a robot who drops the intention of bringing a bottle of beer by breaking the last available bottle and thus making the intention impossible to achieve. Their solution was twofold: (1) they formalize intentions as persistent goals and (2) they assume that existing intentions act as a screen of admissibility over new intentions. In their framework, an agent's intentions persist until she knows that they have been achieved, or knows that it has become impossible to achieve them. Since the robot intends to bring a bottle of beer, she will not drop this goal until she achieves it or gets to know that it has become impossible to achieve. However, the robot can break the last available bottle to make her goal unachievable. But since an agent's current intentions provide a screen of admissibility for adopting new intentions, she cannot have these two conflicting intentions at the same time. Thus since she intends to bring a bottle, she cannot adopt the intention to break the only available bottle. Note that while this problem has similarities with the one addressed here, it does not involve conditional intentions.

In the literature, there has been some work on conditional intention. However, as mentioned earlier, all of the proposed treatments that we are aware of seem to suffer from the under- or over-commitment problems. Although it does not explictly address conditional intentions, the FIPA agent communication language specification [5] defines a type of communication act that leads to conditional intentions. In that framework, an agent can conditionally request another agent to execute an action when some condition holds. This

is modeled as follows: req's conditional request to agt to perform act when ϕ holds amounts to req informing agt that she has the intention that agt execute act and that ϕ be true just before that. Note that req's intention amounts to the conjunction that ϕ be true at some point and agt executes act right after that. Thus, this treatment of conditional intention can be viewed as a conjunctive account where the intention is to first achieve the triggering condition ϕ, and then to achieve the conditional goal. As discussed in Section 1, this leads to the over-commitment problem.

Yolum and Singh [28] present a different model of conditional commitment that relies on a social obligation-based semantics rather than a traditional one based on mental states. Their main concern was the study of communication protocols that accommodate exceptions and take advantage of opportunities. They model interaction protocols using *commitment machines* that supply a content to protocol states and actions in terms of the social commitments of the participating agents. In their formal semantics, which is only briefly described, they adopt a branching time temporal model. The semantics for commitments involves a modal accessibility relation for commitments C that relates a state of the protocol (i.e. a time-point) s, a debtor agent x, and a creditor agent y to a set of paths P. Intuitively, x is responsible to y for satisfying ϕ in state s iff ϕ holds at time-point s along all paths p that are C-accessible from (x, y, s). To model conditional commitment, they introduce a strict implication operator (denoted by $\rightsquigarrow$) that requires the consequent to hold when the antecedent holds. The strict implication is false when the antecedent is false. Their semantics says that $\phi \rightsquigarrow \phi'$ holds in a state s iff ϕ holds in s and for all s' that satisfy ϕ, every s'' that is similar to s' (i.e. $s'' \approx s'$) also satisfies ϕ'. What they mean by the similarity relation $\approx$ is not explained. Thus for them, a conditional commitment $C(x, y, \phi \rightsquigarrow \phi')$ holds in state s iff on all C-accessible paths p, ϕ holds at s, and whenever some s' satisfies ϕ, every s'' that is similar to s' satisfies ϕ'. Since they model conditional commitments using the $\rightsquigarrow$ operator, which behaves like a conjunction with some additional constraints, it appears that their formalization suffers from the over-commitment problem. It is also not clear how their formalization ensures that the goal is achieved *after* the condition along the paths.

Both [1] and [23] model conditional commitment as a disjunctive goal. In their social commitment and argument network based framework, Bentahar et al. [1] define conditional commitments as a simple implication. Their semantics of conditional commitment goes as follows: $M, s \models \text{CondInt}_{Ben}(agt_1, agt_2, \phi, \psi)$ iff $M, s \models \text{EF}^+\phi \Rightarrow M, s \models \text{ABC}(agt_1, agt_2, \psi)$, where s, E, F^+, and ABC denotes a time-point, there exists a path, sometime in the future, and absolute commitment, respectively. This says that agt_1 is committed to agt_2 to achieve ψ on the condition that ϕ means that agt_1 is unconditionally committed to agt_2 to achieve ψ if ϕ holds at some timepoint over some path in the future. Besides suffering from the under-commitment problem associated with disjunctive accounts, this seems to require commitment to the goal too early, before the condition becomes true.

In [23], Shapiro et al. describe a framework for specifying communicative multiagent systems using ConGolog [4] within the situation calculus, an early version of CASL. Since they were lacking a goal-revision mechanism at that point, they introduced a type of conditional request, the *requestUnless* action, in an attempt to avoid the need for goal-revision. $requestUnless(req, agt, \phi, \psi)$ means that req is requesting agt to adopt the goal that ψ unless ϕ is obtained. The execution of $requestUnless(req, agt, \phi, \psi)$ makes agt adopt the goal that $\phi \vee \psi$. This amounts to modeling conditional intentions as disjunctive goals, and hence the account suffers from the under-commitment problem.

6. CONCLUSION

In this paper, we identified some problems with many existing formalizations of conditional commitments. These seem to either have the agents over-committed, intending to achieve the condition under which the goal would have to be achieved, or under-committed, possibly intending that this condition remain false forever. We could not find any problem-free account in the literature. We presented a definition of conditional intentions that does not suffer from these problems. We then formalized two types of communicative actions that allow agents to make requests that lead to conditional commitments. We also proved some properties of conditional commitments and conditional requests. Finally, we discussed previous work on conditional commitments.

Note that, our framework allows an agent with a conditional intention to not intend that the trigger condition eventually becomes true. However, it does not allow her to intend that the trigger condition never comes to hold, without also knowing that it can never become true. In other words, in our framework, an agent's conditional intention that ψ provided that ϕ is not consistent with her intention that $\Box\neg\phi$, unless she already knows that this must be the case. This might be problematic in some cases. For instance, in our example where a seller has a conditional intention to ship a replacement unit when a buyer returns a defective product, we might want to say that the seller has the intention that the buyer never returns a defective good. However, it is not possible for an agent to consistently have both of these intentions in our framework. One way to overcome this limitation might be to adopt a richer semantic model of intention, where one allows different degrees of preferability, similar to the levels of plausibility in traditional belief revision frameworks such as [6]. Such semantic models and the resulting logics are more expressive, but much more complex to specify and reason in.

The theory presented here is a part of our ongoing research on the semantics of speech acts and agent communication in the situation calculus. In [8], we present an extended version of this work where we model some simple communication protocols that deal with conditional requests. Much work remains. In the future, we would like to prove other properties of conditional commitments, for example, about the persistence and revision of such commitments. We also plan to formalize complex interaction protocols, such as the Contract Net protocol [27] and the Net Bill protocol [26], using our formalization of conditional intention. It would also be interesting to try to use this formalization to implement flexible communication agents as in [19] and to develop tools to support multiagent programming.

7. ACKNOWLEDGEMENTS

We thank Hector Levesque, Pinar Yolum, and the review-

ers for useful comments on this work.

8. REFERENCES

[1] J. Bentahar, B. Moulin, J.-J. C. Meyer, and B. Chaib-draa. A logical model of commitment and argument network for agent communication. In *Proc. of AAMAS-04*, pages 792–799, 2004.

[2] P. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–361, 1990.

[3] P. Cohen and H. Levesque. Rational interaction as the basis for communication. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 221–255. MIT Press, Cambridge, Mass., 1990.

[4] G. De Giacomo, Y. Lespérance, and H. Levesque. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121:109–169, 2000.

[5] Foundations for Intelligent Physical Agents. FIPA communicative act library specification, document 37. 1997-2002.

[6] P. Gardenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press, Cambridge, Massachusetts, 1988.

[7] A. Herzig and D. Longin. A logic of intention with cooperation principles and with assertive speech acts as communication primitives. In *Proc. of AAMAS-02*, pages 920–927, 2002.

[8] S. Khan. A situation calculus account of multiagent planning, speech acts, and communication, MSc Thesis (in preparation), 2005.

[9] S. Khan and Y. Lespérance. ECASL: A model of rational agency for communicating agents. In *Proc. of AAMAS-05*, pages 762–769. Utrecht, The Netherlands, July 2005.

[10] G. Lakemeyer and H. Levesque. AOL: A logic of acting, sensing, knowing, and only-knowing. In *Proc. of KR-98*, pages 316–327, 1998.

[11] H. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. Golog: A logic programming language for dynamic domains. *J. of Logic Programming*, 31:59–84, 1997.

[12] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[13] R. Moore. A formal theory of knowledge and action. *Formal Theories of the Commonsense World*, pages 319–358, 1985.

[14] G. Plotkin. A structural approach to operational semantics. Technical Report DAIMI-FN-19, Computer Science Dept., Aarhus University, Denmark, 1981.

[15] A. Rao and M. Georgeff. Modeling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proc. of KR&R-91*, pages 473–484, 1991.

[16] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in the Honor of John McCarthy*. Academic Press, 1991.

[17] R. Reiter. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

[18] D. Sadek. Communication theory = rationality principles + communicative act models. In *Proc. of AAAI-94 Workshop on Planning for Interagent Comm.*, 1994.

[19] D. Sadek and P. Bretier. ARTIMIS: Natural dialogue meets rational agency. In *Proc. of IJCAI-97*, pages 1030–1035, 1997.

[20] R. Scherl and H. Levesque. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1-2), 2003.

[21] S. Shapiro. *Specifying and Verifying Multiagent Systems Using CASL*. PhD thesis, Dept. of C.S., U. of Toronto, 2005.

[22] S. Shapiro and Y. Lespérance. Modeling multiagent systems with the Cognitive Agents Specification Language - a feature interaction resolution application. In C. Castelfranchi and Y. Lespérance, editors, *Intelligent Agents Vol. VII - Proc. of ATAL-00*, volume LNAI 1986, pages 244–259, 2001.

[23] S. Shapiro, Y. Lespérance, and H. Levesque. Specifying communicative multi-agent systems. *Agents and Multi-Agent Systems – Formalisms, Methodologies, and Applications*, LNAI 1441:1–14, 1998.

[24] S. Shapiro, Y. Lespérance, and H. Levesque. The Cognitive Agents Specification Language and verification environment for multiagent systems. In *Proc. of AAMAS-02*, pages 19–26, 2002.

[25] M. Singh. *Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications*. LNAI 799, 1994.

[26] M. Sirbu. Credits and debits on the internet. *Readings in Agents*, pages 299–305, 1998.

[27] R. Smith. The contract net protocol: High level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1980.

[28] P. Yolum and M. Singh. Commitment machines. In J.-J. C. Meyer and M. Tambe, editors, *Intelligent Agents VIII : 8th Intl. Workshop, ATAL-01*, volume LNAI 2333, pages 235–247, 2002.

Contextualizing Commitment Protocols*

Amit K. Chopra*
North Carolina State University
Department of Computer Science
akchopra@ncsu.edu

Munindar P. Singh
North Carolina State University
Department of Computer Science
singh@ncsu.edu

ABSTRACT

Commitment protocols are modularized specifications of interactions understood in terms of commitments. Purchase is a classic example of a protocol. Although a typical protocol would capture the essence of the interactions desired, in practice, it should be adapted depending on the circumstances or *context* and the agents' preferences based on that context. For example, when applying purchase in different contexts, it may help to allow sending reminders for payments or returning goods to obtain a refund. We *contextualize* a protocol by adapting it via different transformations.

Our contributions are the following: (1) a protocol is transformed by composing its specification with a transformer specification; (2) contextualization is characterized operationally by relating the original and transformed protocols; and (3) contextualization is related to protocol compliance.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

1. INTRODUCTION

This paper studies commitment protocols (for short, "protocols"), which are modularized interactions characterized in terms of the commitments among the interacting agents. Protocols are key in designing multiagent systems, such as for implementing business processes. Protocols would be composed and instantiated to realize such systems. Protocols geared for business applications are increasingly being standardized and published, although existing approaches do not offer a formal semantics.

Although protocols are modular and reusable, one size does not fit all. Consider a business example such as a purchase protocol. A basic purchase protocol would capture the essence of how agents should interact to effect a purchase. However, it might not be able to accommodate the many variations needed to handle all the different contexts in which it might be applied. The correct interactions might depend upon the *context* where it is applied: the social or trust relationships between the agents (would the seller insist on advance payment?), the goods being sold (do they need a prescription?), the geographic location (is it legal to sell Schedule C pharmaceuticals without a prescription?), legal considerations (should the seller verify the buyer's age?), and so on. Further variations of context might be inherited from the rules of an electronic marketplace (the Uniform Commercial Code in the US [8]) or a business culture, e.g., can goods be returned for a refund with no questions asked or can goods be returned if they are faulty, and so on. A merchant that knows how to participate in a basic purchase protocol would need to interact differently in each of the above cases.

Developing protocols from scratch for each of the possible contexts would be a difficult undertaking, tedious and error prone at best. This paper proposes an approach wherein protocols are generated by taking an existing protocol and transforming it. To this end, protocols and *transformers* are specified formally and declaratively. Ideally, each transformer is designed for a particular aspect of context. For example, one transformer might accommodate reminders; another might accommodate returns and refunds. In this way, the task of engineering protocols is greatly simplified.

The main contributions of this paper include the following: (1) specifying protocols and transformers formally and independently of each other, (2) formally characterizing transformations based on how they modify a protocol, and (3) determining an agent's compliance with a protocol that takes transformations into account.

This paper advances our program of research that treats interaction as a key abstraction for multiagent systems and treats protocols—as abstractions of interaction—as first-class citizens in contrast to object classes. Modeling protocols enables sophisticated management of interactions and of associated representations such as commitments. Architecturally, an emphasis on interactions is essential for open systems, because openness arises from making the end points dynamically changeable. A system being designed is guaranteed to be open when the interactions among its participants are reified, and the participants are described not as specific entities but in terms of the roles they play in those interactions.

Organization. The rest of the document is organized as follows. Section 2 presents a purchase protocol which serves as our running example. Section 3 develops a language for specifying protocols and a computational model for them. Section 4 presents contexts and transformers relevant to our running example. Section 5 semantically characterizes transformations and defines compliance. Section 6 discusses some key literature and directions.

2. RUNNING EXAMPLE

This paper uses purchase as a running example. This section

*This research was partially supported by the National Science Foundation under grant DST-0139037 and by DARPA contract F30603-00-C-0178.

*Student author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

describes its "base" variant next. Figure 1 shows the base protocol in an informal graphical notation formalized later.

Commitments provide meaning to agent interaction. During the execution of a protocol, agents enter into, modify, and satisfy commitments. A commitment *C(x, y, p)* binds a debtor *x* to a creditor *y* for fulfilling the condition *p*. A conditional commitment *CC(x, y, p, q)* denotes that if condition *p* is brought about, then the commitment *C(x, y, q)* will hold. Following Singh [6], we use the following operations for the creation and manipulation of commitments: *create*, *discharge*, *cancel*, *assign*, *delegate*, and *release*. Commitment operations do not happen arbitrarily. They are typically governed by normative policies that characterize the set of circumstances under which the operations can happen.

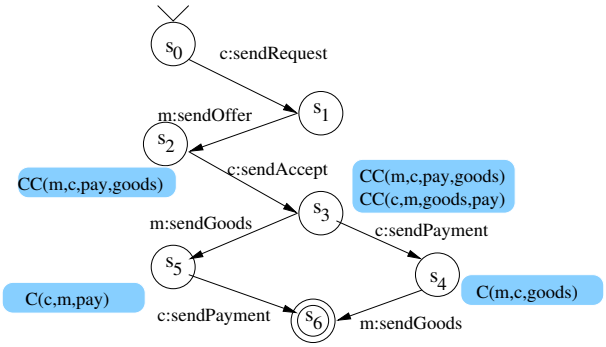


Figure 1: The base purchase protocol

The base purchase protocol has two roles: merchant and customer abbreviated m and c, respectively. For simplicity, we assume reliable, synchronous communication. Thus, sending a message is concurrent with its receipt. The protocol has the following steps. (1) The customer begins the protocol by sending a request for quotes to the merchant. (2) The merchant sends an offer in response. By sending an offer, the merchant creates the conditional commitment CC(m,c,pay,goods) meaning that if the customer pays, then the merchant will send the goods. (3) The customer accepts the offer, thereby creating another conditional commitment CC(c,m,goods,pay) meaning that if the merchant sends the goods, then the customer will pay. (4) If the customer sends a payment to the merchant, then CC(c,m,goods,pay) is discharged and CC(m,c,pay,goods) is reduced to C(m,c,goods) meaning that the merchant is now committed to sending the goods. Conversely, if the merchant sends goods to the customer, then CC(m,c, pay,goods) is discharged and CC(c,m,goods,pay) is reduced to C(c,m,pay) meaning that the customer is now committed to paying for the goods. (5) If the customer has paid in the previous step, then the merchant sends the goods, thereby discharging its commitment. Conversely, if the merchant has sent the goods in the previous step, then the customer sends the payment, thereby discharging its commitment. In either case, no commitments or conditional commitments hold in the resulting state, which is indicated as the final state of the protocol.

3. PROTOCOLS, FORMALLY

We specify protocols declaratively in $C+$ [2], an action description language based on the nonmonotonic causal logic, which captures *universal causation*—every fact that obtains must be caused.

We specify protocols in $C+$ because it supports *elaboration tolerance*. This means that a specification can be refined merely by adding to the existing specification; nothing need be removed from it even if the desired effect is that some inferences are disabled. Elaboration tolerance is a major motivation behind nonmonotonic reasoning. For protocols, elaboration tolerance means that certain interactions can be removed or modified simply by adding axioms to an existing specification. As Section 4 shows, the parts that are to be added correspond to protocol transformers.

3.1 The Action Description Language C+

A specification in $C+$ consists of a set of causal laws. The *signature* of a $C+$ specification is the set of constants that occur in it. ^{act} and ^{fl} represent the sets of actions and fluents, respectively. Each constant c is assigned a nonempty finite domain *Dom(c)* of symbols. An *interpretation* of is an assignment of values $c = v$ for each $c \in$ where $v \in$ *Dom(c)*. A parameterized constant represents a family of constants, the members of which are all its ground instances.

An action description in $C+$ describes a transition system, a graph with states as vertices and actions as edges. Here, a state s is an interpretation of ^{fl} and a transition is a triple $\langle s, e, s' \rangle$ where s and s' are states, and e is an interpretation of ^{act}.

Below, we explain informally the meanings of the kinds of laws we use. Here $a, b \ldots$ represent boolean action literals, and $f, g, \ldots$ represent boolean fluent literals (except in declarations, where they are treated as names); $A, B, \ldots$ and $F, G, \ldots$ represent formulas (combinations using $\wedge, \neg, \vee$) of actions and fluents, respectively.

SCHEMA 1. *a :: exogenousAction.*

Schema 1 declares an exogenous action a, which means that there is a default cause for a to happen or not happen. In other words, a simply happens or does not happen. We model *protocol actions* such as sendRequest and sendOffer as exogenous actions.

SCHEMA 2. *a :: action.*

Schema 2 declares an action that is not exogenous. Such actions are termed *auxiliary actions*. Auxiliary actions must be caused by protocol actions. Commitment operations are auxiliary actions.

Let P_A and A_A denote the sets of protocol and auxiliary actions, respectively. Then $P_A \cup A_A =$ ^{act}, the set of actions of an action description. All actions are atomic—they either happen or do not happen.

SCHEMA 3. *f :: inertialFluent.*

Schema 3 declares a fluent with the property that its value persists from one state to the next unless changed by some other law.

SCHEMA 4. *f :: sdFluent.*

Schema 4 declares f as a static fluent, that is, the value of f depends only on other fluents, not on actions.

SCHEMA 5. *nonexecutable a if b.*

Schema 5 ensures that a and b do not happen concurrently. We use axioms of this schema for every pair of distinct protocol actions, thereby ensuring that no two protocol actions are concurrent.

SCHEMA 6. *a causes b if F.*

Schema 6 says that in a state where F (the precondition) holds, if a happens, then b happens concurrently. If F is the formula $\top$, the law reduces to a causes b. In this case, in a state where a happens, b happens concurrently.

We restrict the usage of Schema 6 so that a and b are protocol and auxiliary actions, respectively. This makes sense if we think of protocol actions causing commitment actions. Also, it does not make sense for b to be a protocol action as we apply Schema 5 to restrict protocol actions to be nonconcurrent.

SCHEMA 7. *a causes F if G.*

Schema 7 says that in a state where G holds, if a happens, then F holds in the next state. If G is the formula $\top$, the law reduces to a causes F. In this case, in a state where a happens, F holds in the next state.

SCHEMA 8. *nonexecutable a if F.*

Schema 8 specifies that in a state where F holds, a cannot happen. Axioms of this schema help impose an ordering on the protocol actions.

SCHEMA 9. *a causes a.*

Schema 9 specifies that the action a happens by default if a is a positive literal. If a is negative literal, then a does not happen by default.

SCHEMA 10. *caused f if f.*

Schema 10 says that the fluent f holds by default if f is a positive literal. If f is a negative literal, then f does not hold by default.

SCHEMA 11. *caused f if F.*

Schema 11 says that f holds in a state where F holds.

3.2 Protocol Specification

Listing 1 specifies commitments and commitment operations in $C+$. Only partial listings are shown due to space limitations. Complete listings for all specifications in this paper are available at http://research.csc.ncsu.edu/mas/code/. Listing 1 is independent of any particular protocol, and is incorporated into each protocol. Commitments and conditional commitments are modeled as inertial fluents (Lines 10–11). Commitment operations are modeled as simple (not exogenous) actions (Lines 12–13). Causal laws of Schema 7 are used to model commitment operations (Lines 18–19). By default, all commitment actions are disabled (Line 27). These laws are required because of universal causation—they explain situations in which the commitment actions do not happen.

```
1  % The listing referred to as 'com-ops' from other listings
2  :- sorts
3   role;
4   condition.
5  :- variables
6   x, y, z          ::role;
7   p,q              ::condition.

9  :- constants
10  ccommitment(role,role,condition,condition),
11  commitment(role,role,condition) :: inertialFluent;
12  create(role,role,condition) :: action;
13  discharge(role,role,condition) ::action;
14  ...

16 % '<>', '++', '&', '-' represent logical
17 % inequality, or, and, and negation, respectively.
18  create(x,y,p) causes commitment(x,y,p) where x<>y.
19  discharge(x,y,p) causes -commitment(x,y,p) where x<>y.
20 % Creation of conditional commitment
21  ccreate(x,y,p,q) causes ccommitment(x,y,p,q)
22                          where x<>y & p<>q.
23  ...

25 % By default, all commitment actions are disabled;
26 % they must be caused by protocol actions.
27  -create(x,y,p) causes -create(x,y,p).
28  ...
```

Listing 1: The basic commitment operations captured in $C+$

Listing 2 specifies the purchase protocol of Figure 1. Note that it includes Listing 1. Protocol actions are declared on Lines 11–12 as exogenous actions. They correspond to the labels on the edges in Figure 1. Fluents that capture the state of the protocol are declared in Lines 15–16. Line 21–22 capture the modeling of the sendRequest action. Lines 25–28 show a part of the modeling of the sendGoods action. Lines 26–28 capture the discharging of the merchant's commitment to send goods. Other protocol actions are appropriately captured, but are not shown here. Towards the end of the listing, the initial and final states of the protocol are defined. The initial state is one where none of the fluents (except initial) hold. The final state is one where a successful exchange of goods and payment has happened. Fluents initial and final are declared to be statically determined (Line 18) as their value depends only on other fluents. Conceptually, initial and final are metafluents: they are part of the ontology for specifying protocols. The nonexecutable axioms serve to impose an ordering on actions. For instance, a sendRequest can happen only in the initial state (Line 22).

```
1  % Include basic commitment operations from Listing 1
2  :-include 'com-ops'.

4  :- objects
5    merchant, customer ::role;
6    goodsc, payc, acceptc ::condition.

8  :- constants

10 % Protocol actions: parameter specifies the performer
11  sendRequest(role), sendOffer(role), sendAccept(role),
12  sendGoods(role), sendPayment(role) ::exogenousAction;

14 % Fluents to characterize the state of the protocol
15  request(role,role), offer(role,role), accept(role,role),
16  goods(role,role), pay(role,role) ::inertialFluent;

18  initial, final :: sdFluent.

20 % Request
21  sendRequest(customer) causes request(customer,merchant).
22  nonexecutable sendRequest(customer) if -initial.

24 % Partial modeling of goods
25  sendGoods(merchant) causes goods(merchant,customer).
26  sendGoods(merchant) causes cdischarge(customer,merchant,
27          goodsc,payc) if ccommitment(customer,merchant,
28          goodsc,payc).
29  ...

31 % Similar axioms for Offer, Accept, and Payment
32  ...

34 % By default, in initial state
35  caused initial if initial.

37 % Not in initial state if any fluent holds
38  caused -initial if request(x,y).
39  caused -initial if offer(x,y).

41 % Similar axioms relating initial and other fluents
42  ...

44 % By default, not in final state
45  caused -final if -final.

47 % In final state, if pay and goods
48  caused final if pay(customer,merchant) &
49                  goods(merchant,customer).
```

Listing 2: The base purchase protocol in $C+$

The sending of a message is captured by the occurrence of the corresponding protocol action. Receiving a message is assumed to be synchronous with its sending. The fluents describe the state of the protocol. Commitment actions give meaning to protocol actions in terms of how protocol actions affect commitments. Therefore, it makes sense to model a commitment action as happening concurrently with the protocol action that causes it.

3.3 Transition Systems

As noted earlier, an action description in $C+$ formally describes a transition system. The transition system of a protocol is based on the transition system of its $C+$ action description with the removal of redundant transitions and unreachable states.

DEFINITION 1. *The set of transitions of a protocol P specified as an action description D is $T_P = \{\langle s, e, s'\rangle \mid \langle s, e, s'\rangle \in T_D$ and $s \neq s'\}$, where T_D denotes the set of transitions of D.*

Thus the transition system of a protocol is different from its action description. In the transition system semantics of $C+$, a nonaction (i.e., the nonoccurrence of an action $a \in {}^{act}$, denoted by the interpretation $a =$ false), is also considered an action. T_D contains transitions where e is interpreted such that every $a \in {}^{act} =$ false. Effectively, no action happens in such transitions. T_D also contains transitions where e is interpreted such that some $a \in {}^{act} =$ true, but the state does not change. (Such transitions could be produced, for example, by axioms of Schema 7 when the action a happens in a state where G does not hold.) Both these types of transitions appear as self-loops around states and are computationally redundant. Definition 1 removes such transitions.

DEFINITION 2. *The set of states of a protocol P specified as an action description D is $V_P = \{s \mid s \in V_D$ and $\exists\langle s', e, s''\rangle \in T_P$ such that $s = s'$ or $s = s''\}$, where V_D denotes the set of states described by D, and T_P denotes the set of transitions of P.*

Definition 2 removes isolated vertices (states). Such vertices have no incoming or outgoing edges.

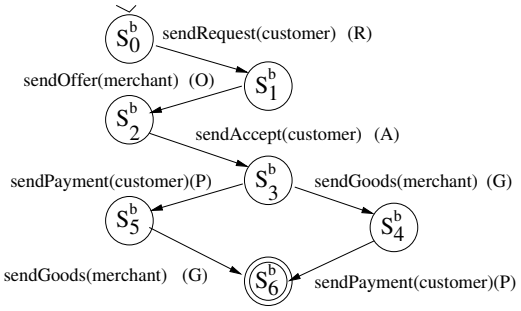


Figure 2: Transition system corresponding to Listing 2

Formally, then, a protocol P is described by $\langle T_P, V_P, V_I, V_F\rangle$, where $V_I \subseteq V_P$ such that $\forall v \in V_I : v \models$ initial, and $V_F \subseteq V_P$ such that $\forall v \in V_F : v \models$ final. Figure 2 shows the transition system corresponding to base purchase (Listing 2). Let's call this protocol P_B (B for base). Table 1 shows the interpretation of the states.

Lemma 1 shows that exactly one protocol action occurs in any transition. We will use this result later.

LEMMA 1. *In the transition system of a protocol P with protocol actions $\{p_1, \ldots, p_n\}$ and auxiliary actions $\{a_1, \ldots, a_m\}$, for every transition $\langle s, e, s'\rangle \in T_P$, exactly one $p_i =$ true $(1 \leq i \leq n)$, in interpretation e.*

Proof. The proof follows from the assumptions specified above about protocol actions and auxiliary actions. By Schema 5, no two protocol actions are concurrent. Therefore, at most one $p_i =$ true $(1 \leq i \leq n)$, in the interpretation e. If some $p_i =$ true, we are done. Else, Definition 1 removes nonaction loops, there must be auxiliary $a_j =$ true $(1 \leq j \leq m)$. However, by Schema 6, auxiliary actions are caused only by protocol actions. Thus some $p_i =$ true in e.

State	Fluents
s_0	*initial*
s_1	*request(customer, merchant)*
s_2	*request(customer, merchant), offer(merchant, customer), CC(customer, merchant, goodsc, payc)*
s_3	*request(customer, merchant), offer(merchant, customer), accept(customer, merchant), CC(customer, merchant, goodsc, payc), CC(merchant, customer, payc, goodsc)*
s_4	*request(customer, merchant), offer(merchant, customer), accept(customer, merchant), goods(merchant, customer), C(c, m, payc)*
s_5	*request(customer, merchant), offer(merchant, customer), accept(customer, merchant), pay(customer, merchant), C(m, c, goodsc)*
s_6	*request(customer, merchant), offer(merchant, customer), accept(customer, merchant), goods(merchant, customer), pay(customer, merchant), final*

Table 1: Interpretation of states in Figure 2: only fluents that obtain are shown.

4. CONTEXTS AND TRANSFORMERS

Section 3 showed how to specify protocols in $C+$, and developed a computational model for them. This section exploits the elaboration tolerance of $C+$ to support the independent specification of protocols and transformers. A protocol transformer is a $C+$ specification that encodes a generic way to handle an aspect of context. To apply a transformer to a protocol, we simply append it to the protocol specification.

A transformer specification may use *external* constants, i.e., fluents and actions that are not declared in it. External constants must be declared in a protocol to which the transformer is applied. Such constants represent part of the interface between the protocol and the transformer. Below we model a few transformers for the base purchase protocol (P_B).

4.1 The Return-Refund Transformer

EXAMPLE 1. *Sending returns and refunds: In some settings, a customer may return goods bought and obtain a refund (if a payment was made). Likewise, a merchant may refund the customer's payment unilaterally if it is unable to deliver the promised goods.* ∎

```
1  :- constants

3    sendReturn(role), sendRefund(role)::exogenousAction;
4    returned(role,role), refunded(role,role)::inertialFluent.

6  % Capturing the effects of the return action

8    sendReturn(customer) causes returned(customer,merchant)
9          & -goods(merchant,customer) if
10         goods(merchant,customer).
11   sendReturn(customer) causes create(merchant,customer,
12         goodsc) if pay(customer,merchant) &
13         goods(merchant,customer).
14   sendReturn(customer) causes cancel(customer,merchant,
15         payc) if commitment(customer,merchant,payc)
16         & goods(merchant,customer).
17   sendReturn(customer) causes ccreate(merchant,customer,
18         payc, goodsc) if commitment(customer,merchant,
19         payc) & goods(merchant,customer).
20   sendReturn(customer) causes ccreate(customer,merchant,
21         goodsc, payc) if commitment(customer,merchant,
22         payc) & goods(merchant,customer).

24 % Sending goods causes them to be not returned.
```

```
26   sendGoods(merchant) causes -returned(customer,merchant).

28 % Axioms for refunds
29   ...
```

Listing 3: Return-Refund transformer in C+

Listing 3 shows a transformer that adds refunds and returns to base purchase. The actions and fluents are declared in Lines 3–4. Lines 8–22 capture the semantics of return. sendReturn(customer) is successful only if goods have already been sent, and it undoes the effect of the sending of goods by the merchant (Lines 8–10). The remaining causal laws capture the effects of performing a return depending on whether payment has already been sent (Lines 11–13) or not (Lines 14–22). In addition, sending goods undoes the effect of returning them (Line 26). This enables cyclic behavior of sending and returning goods. Refunds (not shown in the listing) are captured similarly.

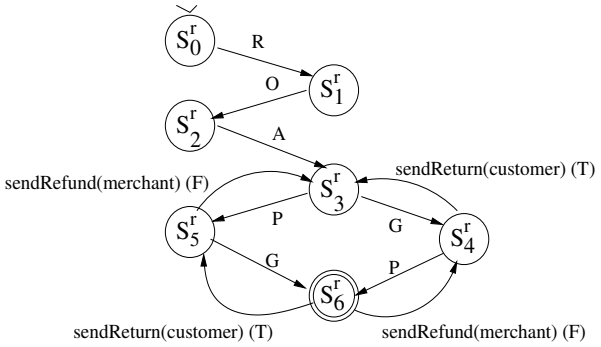


Figure 3: Transition system produced by applying the Return-Refund transformer

To apply this transformer to P_B (base purchase), we append it to the specification of P_B. We refer to the resulting protocol as P_R (for returns and refunds). Figure 3 shows the transition system of P_R. R, O, A, G, P mean the same as in Figure 2.

4.2 The Reminder Transformer

EXAMPLE 2. *Sending reminders: In some context where agents are notorious for not fulfilling their commitments on time, it would be useful for the creditor of a commitment to remind the debtor about the pending commitment. A customer can remind the merchant to send the goods and conversely, a merchant can remind the customer to send a payment.* ∎

```
1  :- constants

3    sendReminder(role) :: exogenousAction;

5  % Every protocol action must cause a transition into
6  % a new state. The num of the reminder serves to
7  % distinguish between two reminders.

9    reminder(role,role,num) :: inertialFluent;

11 % reminderNum keeps track of the number of reminders.
12 % To save space, we removed the lines that say
13 % maxReminders=6

15   reminderNum :: inertialFluent(0..maxReminders).

17 % Reminder: increment reminderNum after every reminder.
```

```
19 sendReminder(y)  causes reminder(x,y,i) & reminderNum=i+1
20         if commitment(x,y,p) & i=reminderNum where
21         i < maxReminders.
```

Listing 4: Reminder transformer in C+

Listing 4 shows a transformer that adds reminders to base purchase. Commitments are the only external constants in this transformer. Both the customer and the merchant can send reminders (limited to six maximum) depending on which commitment holds. The declaration of the fluent reminder(role, role, num) on Line 9 includes a parameter num. Recall that Definition 1 removes self-loops from the transition system. Therefore, protocol actions must be specified so that every time a protocol action happens, it causes a state change. Without the num parameter, reminders after the first reminder would not cause a state change, and would be removed by Definition 1.

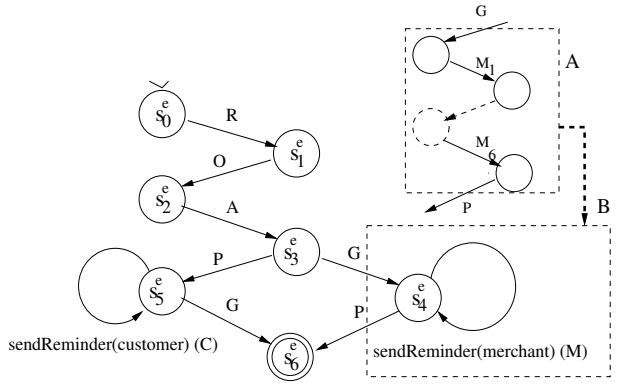


Figure 4: Transition system produced by applying the Reminder transformer

When we apply this transformer to P_B (by appending it to P_B), we obtain a protocol P_E (for reminders). Because the transition system obtained is large, Figure 4 shows the simplified transition system of P_E. Every reminder that the merchant sends transitions the protocol into a new state as shown in block A. We collapse block A into block B, thereby making it seem that reminders happen in a self-loop. We do the same for the customer's reminders. Note that such simplification is not possible in general. We make this simplification here, because we focus on commitments, and sending reminders does not change the commitments in any state.

4.3 The Pay-Before-Goods Transformer

This is an example of a transformer that *removes* some previously allowed interactions. Recall that base purchase allows sending goods and payment in any order.

EXAMPLE 3. *Pay first: In a context where customers are not trustworthy (but merchants are), merchants requires payment before delivery of goods.* ∎

```
1  % Goods sent only after payment.

3    nonexecutable sendGoods(merchant) if
4          -pay(customer,merchant).
```

Listing 5: Pay-Before-Goods transformer in C+

Listing 5 removes the path where the merchant sends goods before payment is received. When we append this transformer to P_B,

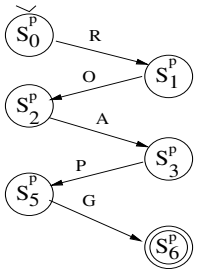


Figure 5: Transition system produced by applying the Pay-Before-Goods transformer

we obtain the protocol P_P (for pay first) as shown in Figure 5. Its transition system is similar to Figure 2, but with the goods-before-payment path removed.

4.4 The Discount Transformer

The previous transformers did not add any new commitments to base purchase; this does.

EXAMPLE 4. *As an incentive for the customer to pay and complete the transaction, the merchant promises to give a 10% discount on the customer's next purchase if the customer pays.* ▮

```
1  :- objects
2  % 10 percent discount on next purchase.

4    discountNextPurchase :: condition.

6  :- constants

8    sendDiscountOffer(role) ::exogenousAction,
9    discountOffer(role,role) ::inertialFluent.

11 % DiscountOffer

13   sendDiscountOffer(merchant) causes discountOffer(
14           merchant,customer).
15   sendDiscountOffer(merchant) causes ccreate(merchant,
16           customer,payc,discountNextPurchase).

18 % DiscountOffer only if not customer has not paid.

20   nonexecutable sendDiscountOffer(merchant) if
21           pay(customer,merchant) ++
22           -offer(merchant,customer).
```

Listing 6: Discount transformer in C+

Listing 6 shows the corresponding transformer. A new action sendDiscountOffer(merchant) is introduced which creates a conditional commitment that if the customer pays, then it receives a 10% discount on the next purchase. This action can happen only before pay, and only after offer.

Applying this transformer to P_B yields the protocol P_D (for discount). In the resulting transition system, the paths in P_B remain and new paths with sendDiscountOffer(merchant) are added. Figure 6 shows part of the resulting transition system with one such added path, between s_0 and s_8. Note that in s_8, a final state, the fluent C(merchant, customer, discountNextPurchase) holds.

5. SEMANTICS OF TRANSFORMATIONS

The base purchase protocol and its transformations described above offer differing functionality, yet are similar in that they are all purchase protocols. The notion of protocol subsumption [3] helps

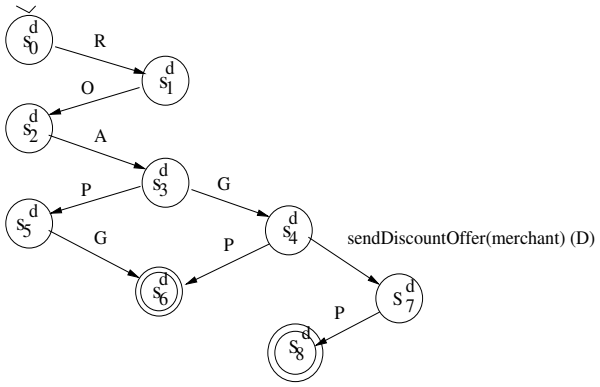


Figure 6: Transition system produced by applying the Discount transformer

characterize the relationship between a protocol and its transformations. The following reviews protocol subsumption, and applies it for understanding transformations.

5.1 Protocol Subsumption

Runs. A run represents a path in the transition system from a initial state to a final state.

DEFINITION 3. *A run of a protocol is a series of transitions* $\langle s_0, e_0, s_1\rangle$, $\langle s_1, e_1, s_2\rangle$, ...,$\langle s_{n-1}, e_{n-1}, s_n\rangle$ *such that* $s_0 \in S$, *the set of initial states, and* $s_n \in F$, *the set of final states.*

$\langle s_0, e_0, s_1, e_1, \ldots, e_{n-1}, s_n\rangle$ is an abbreviated representation of a run. Using Lemma 1, we further abbreviate a run by indicating only protocol actions instead of complete action interpretations, e.g., $\langle s_0, p_0, s_1, p_1, \ldots, p_{n-1}, s_n\rangle$ where $p_i \in P_A$, the set of protocol actions. For example, the set of runs corresponding to Figure 2 is $\{\langle s_0, R, s_1, O, s_2, A, s_3, G, s_4, P, s_6\rangle, \langle s_0, R, s_1, O, s_2, A, s_3, P, s_5, G, s_6\rangle\}$.

State Similarity. We introduce *state similarity* to compare states occurring in runs of different protocols. A state-similarity function f maps a state to a set of states, i.e., $f : \mathbf{S} \rightarrow 2^{\mathbf{S}}$. From f, we induce a binary relation $\approx_f \subseteq \mathbf{S} \times \mathbf{S}$, where $\approx_f = \{(s, f(s)) : s \in \mathbf{S}\}$. We require f to be such that $\approx_f$ is an equivalence relation. For example, commitments could be used to compare states. Two states are *commitment similar* if the same set of commitments hold in them. State-similarity forms the basis of protocol comparison.

Run Subsumption. Let $\prec_\tau$ be a temporal ordering relation on states in a run τ. $s \prec_\tau s'$ means that s occurs before s' in τ.

DEFINITION 4. *A run* τ_j *subsumes* τ_i *under a state-similarity function* f, *denoted by* $\tau_j \gg_f \tau_i$ *if for every state* s_i *that occurs in* τ_i, *there exists a state* s_j *that occurs in* τ_j *such that* $s_j \approx_f s_i$, *and for all* s'_i *that occur in* τ_i, *if* $s_i \prec_{\tau_i} s'_i$ *then there exists* s'_j *that occurs in* τ_j *such that* $s_j \prec_{\tau_j} s'_j$ *and* $s'_j \approx_f s'_i$.

For the sake of subsumption analysis, we omit the actions of a run, thus abbreviating a run $\langle s_0, p_0, s_1, p_1, \ldots, p_{n-1}, s_n\rangle$ to just $\langle s_0, s_1, \ldots, s_n\rangle$. Run subsumption is reflexive, transitive, and antisymmetric up to state similarity [3]. Longer runs subsume shorter runs, provided they have similar states in the same temporal order. For example, the run $\langle s^r_0, s^r_1, s^r_2, s^r_3, s^r_4, s^r_3, s^r_4, s^r_6\rangle$ in purchase with returns and refunds (Figure 3) subsumes under commitment similarity, the run $\langle s^b_0, s^b_1, s^b_2, s^b_3, s^b_4, s^b_6\rangle$ in base purchase (Figure 2, states with the same subscript being commitment similar). However, the run $\langle s^d_0, s^d_1, s^d_2, s^d_3, s^d_4, s^d_7, s^d_8\rangle$ in purchase with discount

(Figure 6) does not subsume under commitment similarity, any run in base purchase as state s_6^b does not have a corresponding commitment-similar state in purchase with discount.

Protocol Subsumption. Informally, a more *general* protocol allows more variations in its runs than a less general one. To help formalize this notion, we define a *span* of a protocol as a set of runs allowed by it. A transition system directly yields a span. We define the *closure* of a protocol as a span that is closed under run subsumption. That is, if a run is in the closure, then all the runs that subsume it (under some state-similarity function) are also in the closure. Closures are unique, and provide a firm basis for comparing protocols.

DEFINITION 5. *The transition-span of a protocol specification P, denoted by $[P]$, is the set of runs in its transition system.*

For example, $[P_B] = \{\langle s_0, s_1, s_2, s_3, s_4, s_6\rangle, \langle s_0, s_1, s_2, s_3, s_5, s_6\rangle\}$ is a transition-span of base purchase.

DEFINITION 6. *The closure of a protocol P under a state-similarity function f is given by $[\![P]\!]_f = \{\tau \mid \forall \tau' \in [P] : \tau \gg_f \tau'\}$.*

DEFINITION 7. *A protocol P subsumes a protocol P' under a state-similarity function f, denoted by $P[\![f]\!\rangle P'$, if $\forall \tau' \in [\![P']\!]_f$, $\exists \tau \in [\![P]\!]_f$, such that $\tau' \gg_f \tau$.*

Considering our example protocols, under commitment-similarity function f, (1) P_B (base purchase) subsumes P_R (with returns and refunds) and vice versa because $[\![P_B]\!]_f = [\![P_R]\!]_f$, (2) P_B subsumes P_E (with reminders) and vice versa because $[\![P_B]\!]_f = [\![P_E]\!]_f$, (3) P_B strictly subsumes P_P (pay-before-goods) because $[\![P_P]\!]_f \subset [\![P_B]\!]_f$, and (4) P_D (with discount) strictly subsumes P_B because $[\![P_B]\!]_f \subset [\![P_D]\!]_f$. (Comparing across Figures 2, 3, 4, 5, and 6, the states labeled with the same subscript are commitment similar.)

DEFINITION 8. *The set of induced runs of a protocol P under a state-similarity function f is $([P])_f = [\![P]\!]_f \setminus [P]$.*

In other words, the induced runs are not explicitly present in the transition system but are implicitly present in the specification.

5.2 Types of Transformations

A transformation can be viewed as a function that takes a protocol as input and produces a different protocol. In simple terms, a transformation either (1) adds, (2) removes, or (3) adds and removes runs from a protocol. The definitions below characterize transformations based on how they change the span of a protocol. The examples below use commitment similarity as the state-similarity function.

DEFINITION 9. *A transformation is closure expanding with respect to a state-similarity function f if the resulting transition-span has at least one additional run that does not belong to the closure of the original protocol.*

The transformation from P_B (Figure 2) to P_D (Figure 6) is closure expanding since it adds the run $\langle s_0^d, \ldots, s_8^d\rangle \notin [\![P_B]\!]_f$.

LEMMA 2. *A protocol resulting from a closure-expanding transformation with respect to a state-similarity function f strictly subsumes the original protocol under f.*

DEFINITION 10. *A transformation is closure preserving with respect to a state-similarity function f if the transition span of the resulting protocol is different from that of the original protocol, but their closures are identical (under f).*

No.	$[P]$		$[\![P]\!]_f$		Transformation
	Removes	Adds	Removes	Adds	
1	x	x	x	x	null
2	x	x	x	✓	Impossible
3	x	x	✓	x	Impossible
4	x	x	✓	✓	Impossible
5	x	✓	x	x	Closure preserving
6	x	✓	x	✓	Closure expanding
7	x	✓	✓	x	Impossible
8	x	✓	✓	✓	Impossible
9	✓	x	x	x	Closure preserving
10	✓	x	x	✓	Impossible
11	✓	x	✓	x	Closure contracting
12	✓	x	✓	✓	Impossible
13	✓	✓	x	x	Closure preserving
14	✓	✓	x	✓	Closure expanding
15	✓	✓	✓	x	Closure contracting
16	✓	✓	✓	✓	Not characterized

Table 2: Transformation possibilities

The transformations from P_B (Figure 2) to P_R (Figure 3) and to P_E (Figure 4) are both closure preserving since each transformation adds runs that were already present in $[\![P_B]\!]_f$. A transformation that removes runs from the transition span, but is still closure-preserving, is one that removes reminders from P_E.

LEMMA 3. *A protocol resulting from a closure-preserving transformation with respect to the state-similarity function f subsumes the original protocol under f, and vice versa.*

DEFINITION 11. *A transformation is closure contracting if the resulting transition-span is a proper subset of the transition-span of the original protocol.*

The transformation from P_B (Figure 2) to P_P (Figure 5) is closure contracting since it removes the run $\langle s_0^b, \ldots, s_4^b, s_6^b\rangle \in [P_B]$.

LEMMA 4. *A protocol subsumes a protocol resulting from a closure-contracting transformation under every f.*

Table 2 shows that the above-defined types of transformations represent an exhaustive list; there can be no other type of transformation. Each row is to be read as "a transformation that Removes (if ticked) and Adds (if ticked) from $[P]$, and at the same time Removes (if ticked) and Adds (if ticked) from $[\![P]\!]_f$ is of type Transformation". To understand the table, we present the following observations that follow from the definition of closure. (1) To change the closure, the transition span must be changed making the transformations in rows 2–4 impossible. (2) A transformation that only adds runs to the transition span cannot remove runs from the closure making transformations in rows 7–8 impossible. (3) Similarly, a transformation that only removes runs from the transition-span cannot add runs to the closure making rows 10 and 12 impossible. (4) The transformations of the type in row 16 cannot be generally characterized in terms of closures—depending on the specific transformation in consideration, the transformation may be expanding, preserving or contracting, or there may be no subset relation between the closures of the original and the resulting protocol—causing them to be labeled 'not characterized'. We lack the space to give detailed examples.

5.3 Compliance

How can we use the above characterization of transformations? One use is in determining compliance. In open systems, checking whether an agent complies with stated protocols is nontrivial. A simple approach for verifying compliance is to check an agent's enactment of a protocol against the protocol's state machine: any deviation from the state machine would be a violation. But some deviations may be necessary for variations in context. A commitment-based approach can allow flexible enactments while enabling verification of compliance [9]. Practical settings, however, often require more structure than general-purpose commitment reasoning allows. In particular, agreement among the interacting parties (expressed in a protocol) may be needed to facilitate their interactions. For example, in a certain context, the parties involved may find sending reminders appropriate, but not allowing arbitrary returns. Where do we get this structure from?

The closure of a protocol specifies which runs it allows. This is important not just for deviations but also for accommodating agents who participate in multiple protocols in a potentially interleaved manner. By changing the similarity function f, we obtain different closures. Thus, the state-similarity function leads to a configurable definition of compliance. An appropriate similarity function can be defined for each domain of interest or institution. Below, enactment refers to an agent's interactions.

DEFINITION 12. *An agent is compliant with a protocol P in a domain f as long as any enactment belongs to $[\![P]\!]_f$.*

Given Definition 12, we relate transformations to compliance as follows. Consider a domain f in which agents use protocol P. If an agent replaces P with another protocol P' and acts in a manner compliant with P', then under what conditions can we say that it is also compliant with P? To be compliant with both P and P', an agent's enactment must belong to both $[\![P]\!]_f$ and $[\![P']\!]_f$.

LEMMA 5. *An agent compliant with P' in a domain f is compliant with protocol P if P' can be obtained from P using a closure-preserving transformation with respect to f.*

In a domain with commitment-similarity, an agent that is compliant with P_R (with returns and refunds) or P_E (with reminders) is compliant with P_B (base purchase), but an agent using P_P (pay first) or P_D (with discount) is not.

6. DISCUSSION

This paper has formalized protocols and transformers, demonstrated how they work, and shown some of their useful properties. When protocols and transformers are treated as software components [7], we can think of creating libraries of them, and using them as off-the-shelf components to create the desired multiagent systems. The key requirement behind components is that they are sufficiently rigorously defined that third-party developers can compose them in ways that the original designers might not have anticipated. This requirement is met for protocols by the present approach.

Some interesting challenges remain. How do we know that a transformer does not conflict with a protocol? For example, applying a Goods-Before-Pay transformer (the inverse of the example of Listing 5) on the Pay-Before-Goods protocol of Figure 5 will result in a protocol with no runs. Clearly, we can build a transition system for the transformed protocol and observe that it allows no runs. However, a characterization based on the specification language would be more perspicuous and helpful during design. In general, what kinds of correctness guarantees can we make regarding a transformer based on the schemas used in it?

Engineering protocols, like any other artifacts, presupposes the existence of effective tools. Because, in practice, protocols would be developed in multiple ways, more than one kind of tool is needed. This paper has concentrated on supporting protocol design from existing protocols. Others have addressed this problem via a composition of existing protocols [10] or as a refinement of part of an existing protocol [4]. In specification terms, a transformer is not different from a protocol. Thus the present approach has elements of composition in a rigorous manner and shows in what cases the resulting protocols are refinements of the original protocols. An alternative approach is to support protocol design from first principles, i.e., based on requirements analysis of the desired class of applications, an approach such as Dooley graphs [5] or Tropos [1] that yields specifications from early requirements would be natural. It would be interesting to see how the approaches can be combined so that we can develop a library of protocols and transformers from first principles, and enable the development of additional protocols as needed from the components in the library.

7. REFERENCES

[1] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Journal of Autonomous Agents and Multiagent Systems*, 8(3):203–236, May 2004.

[2] Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153(1-2):49–104, 2004.

[3] Ashok U. Mallya and Munindar P. Singh. An algebra for commitment protocols. *Journal of Autonomous Agents and Multiagent Systems special issue on Agent Communication*, 2006. To appear.

[4] Hamza Mazouzi, Amal El Fallah Seghrouchni, and Serge Haddad. Open protocol design for complex interactions in multi-agent systems. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 517–526, 2002.

[5] H. Van Dyke Parunak. Visualizing agent conversations: Using enhanced Dooley graphs for agent design and analysis. In *Proceedings of the 2nd International Conference on Multiagent Systems*, pages 275–282. AAAI Press, 1996.

[6] Munindar P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 7:97–113, 1999.

[7] Clemens Szyperski. Components and the way ahead. In Gary T. Leavens and Murali Sitaraman, editors, *Foundations of Component-Based Systems*, chapter 1, pages 1–20. Cambridge University Press, 2000.

[8] Uniform commercial code. http://www.law.cornell.edu/ucc/ucc.table.html.

[9] Mahadevan Venkatraman and Munindar P. Singh. Verifying compliance with commitment protocols: Enabling open Web-based multiagent systems. *Journal of Autonomous Agents and Multi-Agent Systems*, 2(3):217–236, September 1999.

[10] Benjamin Vitteau and Marc-Philippe Huget. Modularity in interaction protocols. In Frank Dignum, editor, *Advances in Agent Communication*, volume 2922 of *Lecture Notes in Computer Science*, pages 291–309, 2004.

A Utility-Based Sensing and Communication Model for a Glacial Sensor Network

Paritosh Padhy, Rajdeep K. Dash, Kirk Martinez and Nicholas R. Jennings
School of Electronics and Computer Science,University of Southampton, Southampton, SO17 1BJ ,UK.
{pp04r,rkd02r,km,nrj}@ecs.soton.ac.uk

ABSTRACT

This paper reports on the development of a utility-based mechanism for managing sensing and communication in cooperative multi-sensor networks. The specific application considered is that of GLACSWEB, a deployed system that uses battery-powered sensors to collect environmental data related to glaciers which it transmits back to a base station so that it can be made available world-wide to researchers. In this context, we first develop a sensing protocol in which each sensor locally adjusts its sensing rate based on the value of the data it believes it will observe. Then, we detail a communication protocol that finds optimal routes for relaying this data back to the base station based on the cost of communicating it (derived from the opportunity cost of using the battery power for relaying data). Finally, we empirically evaluate our protocol by examining the impact on efficiency of the network topology, the size of the network, and the degree of dynamism of the environment. In so doing, we demonstrate that the efficiency gains of our new protocol, over the currently implemented method over a 6 month period, are 470%, 250% and 300% respectively.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence, Multiagent systems, Coherence and coordination.

General Terms

Algorithms, Design, Experimentation

Keywords

Agents and ambient intelligence, agents and novel computing paradigms, Agent-based sensor networks.

1. INTRODUCTION

Multi-sensor networks are being deployed in a wide variety of application areas and, in particular, they have recently been advocated for a number of environmental monitoring applications [8]. Moreover, there is an increasing interest in controlling these networks using multi-agent system techniques [7]. In this vein, we consider a particular sensor network, GLACSWEB [8], that we have deployed in the Brikdalsbreen glacier in Norway, and examine how it can be modelled as a (cooperative) multi-agent system. In this case, the two main tasks performed by the sensors (agents) are gathering data from the environment and communicating it towards a central sink node (i.e. an agent that harvests data from all other agents). In general, the agents work towards the predefined system goal of maximising data collection (hence the cooperative nature of the system). However, they are invariably constrained in at least one of the following dimensions: their available power, their communication bandwidth, their memory storage and their processing capability. Of these, power is the most important since it is required for everything else. Thus, it directly influences the life-span of the agents and, hence, that of the system as a whole. Given this, we focus on developing a communication and a sensing protocol for this network. Nevertheless, the solution we develop is also more broadly applicable to networks that have any form of limited power supply.

In more detail, the purpose of the GLACSWEB multi-sensor network is to monitor sub-glacial behaviour in order to understand climatic change. Figure 1 shows GLACSWEB's central base station that is located on top of the glacier and figure 2 shows a typical GLACSWEB node. The individual nodes each sense their own data and then communicate it directly to this sink node. As such, the system's communication protocol is energy inefficient since it lacks the energy savings that a multi-hop approach would provide [12] (i.e. one in which agents relay data for one another). Furthermore, at present, sensing in GLACSWEB is carried out at a pre-determined constant rate which is blind to the actual variations in the environment. This decoupling results in unnecessary sampling because, given the same energy expenditure, the information gained by sensing a slowly varying environment is less than what could be gained in a more dynamic situation.

Against this background, this paper develops a Utility-based Sensing and Communication protocol (called USAC). This consists of a sensing and a routing protocol that uses the cost of transmission and the value of observed data as utility metrics in the agents' decision-making process. In doing so, we advance the state of the art in the following ways :

- We develop a novel mechanism for adaptive sampling.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

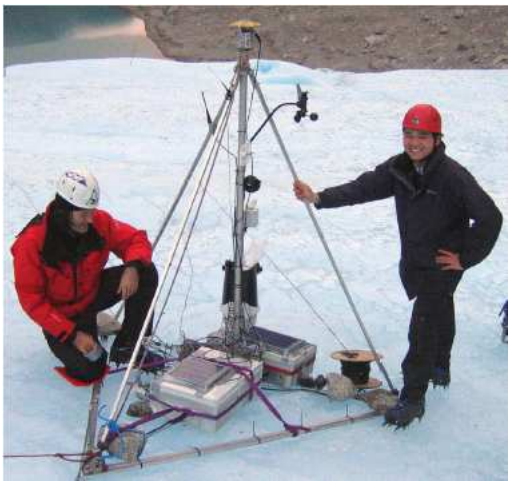


Figure 1: GLACSWEB base station

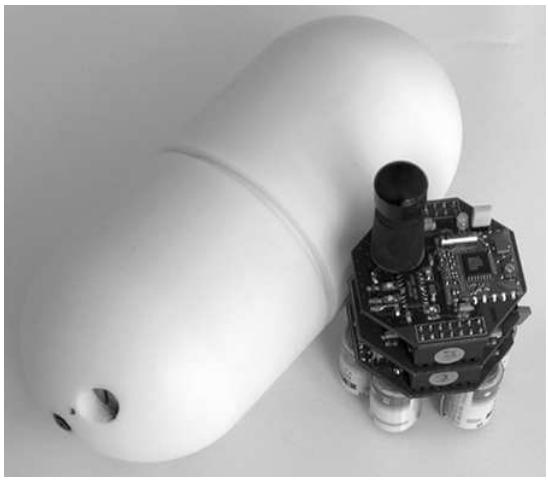


Figure 2: A GLACSWEB sensor probe

In this, each agent adjusts its rate depending on the rate of change of its observations and a valuation function (based on a sound information theoretic foundation) that the agents use for assigning a value to the data they observe.

- We devise a new routing protocol that finds the cheapest cost route from an agent to the centre. Here, the cost of a link from one agent to another is derived using the opportunity cost of the energy spent relaying the data (i.e. the value that a relay could have gained by using the energy in sensing instead of relaying).

- We empirically evaluate the USAC protocol against the current GLACSWEB protocol and show that it introduces a significantly higher gain in information, whilst reducing power consumption.

The remainder of this paper is organised as follows. Section 2 provides some basic background on GLACSWEB and a discussion about related work in the area of sensor networks modelled as multi-agent systems. We then detail the sensing protocol and the routing protocol of USAC in sections 3 and 4 respectively. The USAC protocol is then empirically evaluated in section 5 in a wide range of scenarios. We conclude in section 6.

2. BACKGROUND AND RELATED WORK

GLACSWEB is a deployed pervasive sensor network that directly monitors sub-glacial movement to determine how it is affected by climatic changes. In order to do this, it uses a network of sub-glacial probes which are placed at different locations inside the glacier as shown in figure 3.

The protocol currently followed by GLACSWEB is a simple one in which the probes sample the glacier every four hours and then transmit these readings directly to the base station located at the top of the glacier daily. At this time, there are twenty probes in the glacier with an additional batch of ten deployed every summer. However, as a result of the hostile environment (due to the low temperatures, the strain resulting from moving ice and the englacial water bodies) and the power hungry protocol, there is a high probe failure rate of around 50% per year. This provides the motivation for the research carried out in this paper.

Speaking more generally, it is widely recognised that there is a strong need for a well defined protocol stack in a sensor network that helps combine power and routing awareness, and that integrates data with the networking protocols. To this end, a number of routing protocols have been investigated in this area. In particular, flooding [4] is one of the oldest and most common routing techniques. In flooding, each agent receives an item of data and then repeats it by broadcasting unless the destination of the packet is the agent itself. This is a reactive technique and does not require costly topology maintenance or complex route discovery algorithms. However, it can easily cause data implosion and/or data overlap which results in unnecessary wastage in communication power. The SPIN [5] routing protocol adopts a publish-subscribe approach to the problem in which the agent nodes operate efficiently and conserve energy by all sending meta-data describing the sensor data instead of sending the actual data. Thus, this model is useful for those agents interested in the data advertised and is an effective protocol to minimise energy spent in consumption until the actual data is transmitted. However, it fails to place a limit on the energy consumed in wasted advertisements (i.e. for which there are no subscribers for the data). Moreover, the downside of both these models is that they involve significant amounts of communication and so consume large amounts of power which is often limiting.

To combat this, a number of researchers have focused on intelligent adaptive sensing [3]. Such work demonstrates that an active sensing approach can maximize a mobile sensor network's lifetime by sensing only during the most informative situations. However, our research differs from such approaches in that we focus on the sudden changes in the observed data rather than simply investigating regular patterns. In addition, our work reuses the utility derived from the sensing protocol in the communication protocol, thereby intertwining these two critical aspects of a sensor network. As a result, our model is sensor-centric, in contrast to the query-centric model adopted in [3].

The concept of utility has previously been used in the context of sensor networks for both cooperative [1, 6] and selfish

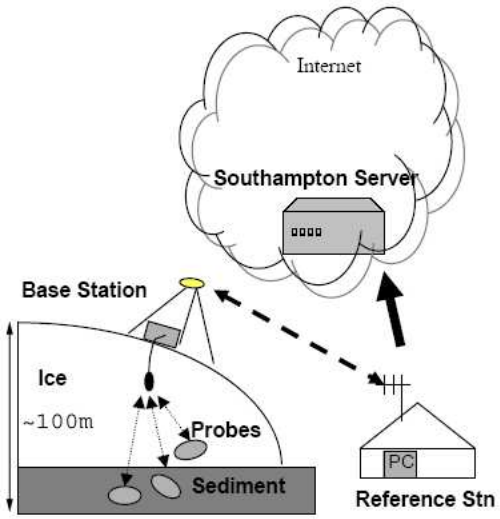


Figure 3: Architecture of the GLACSWEB network.

agents [2, 11]. In the latter case, the concept of selfishness arises mainly in those applications where agents are individually owned by different stakeholders (which is not true in our case). The focus of our work is on cooperative agents (since all nodes are owned by one stakeholder, the University of Southampton) and differs from existing research by combining the sensing and communication protocol via the utility function.

3. THE SENSING PROTOCOL

In this section, we detail the sensing protocol of USAC which provides the agents with a sensing schedule. We concentrate here only on the sensing actions of an agent (thereby ignoring the communication actions dealt with in section 4 so as to simplify the exposition). This focus results in a single agent model in which the agent has to make optimal local decisions concerned with the timing of its sensing.

In more detail, the model for the sensing protocol is as follows. From the remaining battery energy, E_i, and the energy, E_{sense}, each sensing action takes[1], agent i can calculate the maximum number of sensing actions, N_s^r, it has remaining ($N_s^r = E_i/E_{sense}$). Given this, the agent then needs to derive an optimal schedule for determining *when* to carry out these actions. In order to do this, a metric is required to determine how well a particular schedule does compared to another. The metric we use in this case is derived from *information-theory* because this enables us to have a principled means of obtaining maximum information from the environment.

3.1 The Valuation Function

An optimal sampling rate can only be derived if the agent has knowledge about the future data. However, this requirement is contradictory since in the case that the agent knows the future data, it does not need to sense the environment. As a result, an agent can only find an optimal sampling rate based on its forecast of the future data. Then, upon observing previously forecasted data, the agent gains information by reducing its uncertainty about this data to zero. Here the representation we use for the uncertainty of the data is the confidence interval of a predicted data point. Specifically, the x% confidence interval of a data point is a range of values centred around the mean of the forecast within which x% of the data is guaranteed to fall. For example, a 100% confidence interval is the range of values the data can only ever take, whereas a 10% confidence interval implies that only 10% of the forecasted data is guaranteed to fall in this interval. From this, it can be seen that a lower confidence interval is more useful since its range is smaller. Therefore, a good forecast is one that minimises the confidence interval.

The forecasting method we use here is based on a limited-window linear regression model. This is because it provides a good forecasting model for data that can be characterised as piecewise-linear functions of time with added Gaussian noise (which is the case for data collected in the GLACSWEB environment). Thus, the data can be represented by a set of equations of the form:

$$x_{T_i+\tau} = \alpha_i + \beta_i\tau + \eta \tag{1}$$

where $\{T_1, T_2, \ldots, T_I\}$ represent the times at which a phase change (i.e. a sudden variation in the data model) occurs, $\alpha_i + \beta_i\tau$ is the equation of the line after the phase change at time T_i, and η is noise.

Thus, there are three major aspects of the data which are unknown:

- The current line segment equation (i.e. $\alpha_i + \beta_i\tau$).
- The time at which a phase change occurs (i.e. $\{T_1, T_2, \ldots, T_I\}$).
- The level of noise in the environment (i.e. η).

Now, given observations $(\widehat{x}_1, \ldots, \widehat{x}_n)$ taken at time $(t_1, \ldots, t_n)$, a linear regression model of window size m would then estimate β_i as:

$$\overline{\beta}_i = \frac{\min(m,n)\sum_{i=lb}^{m}(t_i.\widehat{x}_i) - \sum_{i=lb}^{m} t_i \sum_{i=lb}^{m} \widehat{x}_i}{\min(m,n)\sum_{i=lb}^{m} t_i^2 - \left(\sum_{i=lb}^{m} t_i\right)^2} \tag{2}$$

where $lb = \max(n - m + 1, 1)$. Furthermore, using this model, we can estimate α_i as:

$$\overline{\alpha}_i = \frac{\sum_{i=lb}^{m} \widehat{x}_i - \overline{\beta}_i \sum_{i=lb}^{m} t_i}{\min(m,n)} \tag{3}$$

From this, the value at time interval ϵt after the last measurement is predicted as:

$$x_{t_n+\epsilon t} = \overline{\alpha}_i + \overline{\beta}_i(t_n + \epsilon t) \tag{4}$$

along with confidence intervals given by:

$$ci(t_n) = 2\sigma\mathcal{N}(c/2) \tag{5}$$

where σ is the standard deviation of the m samples and $\mathcal{N}(c)$ is the inverse of the positive part of the normal distribution function and c is the confidence level. Given this, we then calculate the value of a data point as:

$$V(data(t_s)) = \mid ci(t_{s-1}) - ci(t_s) \mid \tag{6}$$

[1]The sensors within GLACSWEB have the same E_{sense} and take the same amount of time to carry out one observation.

ALGORITHM 1.

1. *Observe data,* $n_{sample}++$
2. *If* $n_{sample} = 1$ *then* $V(data(t_s)) = V_{max}$
3. *If* $n_{sample} = 2$ *then* $V(data(t_s)) = V_{max}$, *calculate* $ci(t_s)$
4. *If* $n_{sample} > 2$ *then calculate* $ci(t_s)$ *and* $V(data(t_s))$
   - *if* $data(t_s) \in ci(t_{s-1})$ *then* $f_{sample}(t_s) = max(\alpha f_{sample}(t_{s-1}), f_{min})$
   - *else* $f_{sample}(t_s) = f_{max}$ *and* $n_{sample} = 1$
5. *Go to Step 2*

Figure 4: The sensing protocol algorithm.

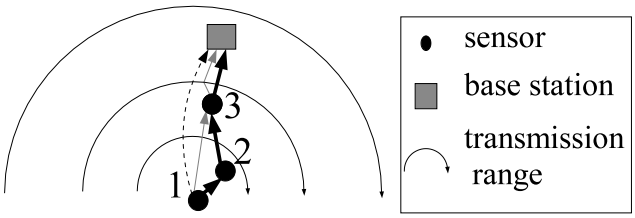
where $ci(t_{s-1})$ is the confidence interval before the data was sampled at time t_s and $ci(t_s)$ is the confidence interval at time t_s calculated using $data(t_s)$. Having defined the sensing protocol, we now turn to the algorithm to implement it.

3.2 The Algorithm

The algorithm we embed inside each agent for the sensing protocol is given in figure 4. The core concept behind this protocol is reflected in step 4. Here the reasoning is that if at time t_s the observed data falls outside the confidence interval, then the agent should set its sampling rate f_{sample} to a maximum f_{max}. This is because if the data falls outside the interval then the agent believes that there has been a phase change in the environment. Hence it starts sampling at the most frequent rate in order to better incorporate this phase change in its forecasting model. However, if data falls within the confidence interval, it implies that the agent already has a good regression model. Hence, it can afford to reduce its sampling rate (represented in step 3 by the multiplicative factor $\alpha \in [0, 1]$) until the lower bound f_{min} which has been set by the requirements of the glaciologists. Steps 2 and 3 enable the agent to obtain the two data points required to carry out a linear regression.

4. THE ROUTING PROTOCOL

Once a sensor has collected data from the environment, it needs to transmit it towards the base station. So far, in GLACSWEB this is done by direct transmission to the centre [10]. However, as discussed in section 2, this is inefficient since the power required to transmit data from one node to another is proportional to the square of the distance between the nodes (from basic radio transmission theory). As a result, the total energy spent by transmitting data directly to the centre via a single hop is more than the energy spent when the data is relayed via successive intermediaries to the centre. In order to see this effect, consider the example shown in figure 5. Here, sensor 1 could transmit data to the base station (bs) via the following three routes: $1 \rightarrow 2 \rightarrow 3 \rightarrow bs$ (bold), $1 \rightarrow 3 \rightarrow bs$ (grey) and $1 \rightarrow bs$ (broken line). The total energy consumed for the transmission of one packet of data would then be 12 $(4 + 4 + 4)$, 20 $(16 + 4)$ and 36 respectively, thereby suggesting the use of route $1 \rightarrow 2 \rightarrow 3 \rightarrow bs$.



sender \ receiver	2	3	base
1	4	16	36
2	X	4	16
3	4	X	4

Figure 5: Three possible routes via which sensor 1 can transmit its data to the base station. The concentric semi-circles show the range of sensor 1 with three power levels chosen such that the range grows linearly. The table shows the energy required for a sensor to transmit a packet directly to another.

However, an approach based solely on the transmission power is too naïve since it disregards the two following aspects:

1. **The opportunity cost of the energy used by each sensor.** If a sensor does not relay data, it could then use that energy in order to carry out additional sensing (which contributes towards the value of the network). Since each sensor is in a different local environment (due to the different placement of the sensors in the glacier), they derive different values by sensing the environment. Hence, it might be preferable for a sensor to transmit its data via a more energy-consuming route if the least energy-consuming route contains a sensor in a highly dynamic environment.
2. **The total power required to transmit along a particular route.** The transmission of data also requires the receiving node to be in a listening mode (i.e. the agent needs to switch on its antenna for receiving data which also consumes power). Thus the route $1 \rightarrow 2 \rightarrow 3 \rightarrow bs$ requires both sensor 2 and 3 to additionally spend energy receiving the data.

We tackle these two problems by developing a utility-based communication protocol. This protocol is based on the value of the data to be routed to the base station (which is derived according to section 3) and the cost of transmitting the data. We next detail how to calculate the cost of communication, before going onto the algorithm used for the communication protocol in section 4.2.

4.1 The Cost of Communication

The multisensor network is modelled as a multi-agent system consisting of a number of agents, $\mathcal{I} = \{1, \ldots, n\}$, that each have K different discrete power levels, $\{pt_i^1, \ldots, pt_i^K\}$, (with $pt_i^{k+1} > pt_i^k$) at which they can transmit. At each level, there is a set of neighbours $n_i(pt_i^k) \subseteq \mathcal{I}$ to which agent i can transmit data. Due to the nature of radio transmission, $n_i(pt_i^k) \subseteq n_i(pt_i^{k+1})$.

Thus, the direct communication of data from any agent i to another agent j, where $j \in n_i(pt_i^k)$ consumes a certain

amount of energy Et_i^j which is given by:

$$Et_i^j(data) = pt_i^{k*} \times t_i^j(data)$$

where pt_i^{k*} is the lowest power level at which $j \in n_i(pt_i^k)$ and $t_i^j(data)$ is the amount of time a data packet takes to transmit. Now, in this scenario the size of each sensed data packet and the bandwidth available to each agent is the same, so $t_i^j(data)$ is constant for all agents and sensed data packets. Therefore, by slight abuse of notation, we shall hereafter refer to $Et_i^j(data)$ as Et_i^j.

The cost of communication of an agent i to another agent j is then the opportunity cost of that decision. Now, there are two particular scenarios to consider when communicating data. If, on one hand, an agent is originating the data, then its cost of communication is given by:

$$c_i^j(originate) = \frac{Et_i^j}{Et_i^j + E_i^{sense}} \times v_i^{sense}(t_n) \quad (7)$$

where E_i^{sense} is the energy spent by i in sensing new data and $v_i^{sense}(t_n)$ is the value of the new data. On the other hand, if an agent is relaying data, then its cost of transmission is given by:

$$c_i^j(relay) = \frac{Et_i^j + E_i^{receive}}{Et_i^j + E_i^{sense}} \times v_i^{sense}(t_n) \quad (8)$$

where $E_i^{receive}$ is the energy spent by the agent receiving the data which it then relays.

Now, since it is not possible to assign $v_i^{sense}(t_n)$ before actually carrying out the observation, we need to estimate it. Due to the nature of the data (where sudden changes are possible) we estimate $v_i^{sense}(t_n)$ using a moving average with window size w. Thus at time t_n the estimated value of the data is given by:

$$\overline{v_i}^{sense}(t_n) = \frac{1}{\min(n, w)} \sum_{i=\max(n-w,0)}^{n-1} v_i^{sense}(t_i)$$

We choose such a forecasting method since it evens out the changes in value that random noise can introduce, whilst at the same time updating the value of the data fairly quickly as time progresses. However, it should be noted that this forecasting method (or for that matter any forecasting method) cannot guarantee to correctly predict the value of the data all the time. Also, the moving average only starts once the number of samples collected by the sensor $> w$. Up to that point, the estimated value is just an average.

Having thus explained how the cost of communication is calculated, we now detail the algorithm followed by each agent when communicating data.

4.2 The Algorithm

The algorithm we use for the communication protocol is given in figure 6. It consists of four main steps, namely:

1. **Initialisation.** In this phase, the network topology is discovered and each agent is made aware of the power level it must transmit at in order to reach each of its neighbours. This phase is run each time probes are deployed within the glacier.

ALGORITHM 2.

1. ***Initialisation.***

   Run a network discovery protocol that establishes $n_i(pt_i^k)\ \forall k, i$. Go to step 3.

2. ***Update transmission power.***

   At predefined intervals of time, t_{update} ($> 1/f_{min}$), the centre broadcasts a message $msg_0 = \langle P_0^{trans}, P_0^{rec} \rangle$ at the highest power level pt_0^K, where P_0^{trans} is the power at which the centre has transmitted this message and P_0^{rec} is the minimum power at which the centre can receive data. An agent can then calculate the minimum power required to transmit to the centre as:

   $$P_i^{min} = \frac{P_0^{rec} P_i^{rec}(msg_0)}{P_0^{trans}}$$

   assuming that the dissipation of power is symmetric between 0 and i. From P_i^{min}, an agent can then determine the minimum power level $pt_i^{k*}(0)$ required to transmit to the base station (since $0 \in n_i(pt_i^k)$ if $pt_i^k > P_i^{min}$). It can then determine $Et_i^0(data)$ and thus $c_i^j(originate)$.

3. ***Update cost of transmission to base.***

   Let $I(k*) \in I$ be the set of agents that require the minimum power level pt_i^{k*} to transmit to base ($I(k*) = n_0(pt_0^{k*}) - n_0(pt_0^{k*-1})$). Note that $pt_0^0 = \{0\}$. Agents $i \in I(1)$ can calculate their own cost of relaying data to the base station, $c_i^0(relay)$. Upon calculation, they broadcast the message $\langle c_i^0(relay) \rangle$ at power level pt_i^K.
   Then for $k = 2$ to K do*

   - *Agents $i \in I(k*)$ calculate the cost of relaying data $c_i^j(relay)$ to all agents $j \in I(k*-1)$.*
   - *They also update their cost of transmission, $c_i^0(originate)$, as $\min(c_i^l(originate) + c_l^0(relay))$ where $l \in \cup_{a=1}^{k*-1} I(k*-a)$*
   - *They then broadcast the message $\langle c_i^0(relay) \rangle$ at power level pt_i^K where $c_i^0(relay) = \min(c_i^l(relay) + c_l^0(relay))$.*

4. ***Transmit data.***

   Send the data packet through the lowest cost path if $V_i(data(t_s)) > c_i^0(originate)$. Update $c_i^j(originate), c_i^j(relay)$ from the value of newly sensed data.

5. ***Repeat Step.***

   ***If** time to update transmission power levels,*
   ***then** go to Step 2*
   ***else if** time to update relay and originate costs*
   ***then** go to Step 3*
   ***else** sense data and go to step 4.*

Figure 6: The routing protocol algorithm.

2. **Update energy band of agent.** This step is responsible for dividing the agents into different power level groups with respect to the base. This segmentation is then used in the next step in order to update the cost of relaying data.

3. **Update cost of transmission to base.** This step is required so as to find the minimum cost route from each agent to the centre. In order to do so, agents in each power level group successively transmit the cost of their cheapest route to the centre.

4. **Transmit data**. Having found the cheapest cost of transmission to the base, the agent then decides whether or not to transmit its observed data.

5. EMPIRICAL EVALUATION

The aim of these experiments is to determine the relative performance of USAC when compared to the currently deployed GLACSWEB protocol. Now, the performance of any protocol in this scenario is contingent on a number of factors such as the network topology, the number of agents within the network and the type of environment in which it is employed. Thus, in order to generalise our results, we benchmark the performance of USAC against GLACSWEB, whilst independently varying each of the factors mentioned above. We also calculate the *efficiency* (i.e. the value of the data gained over the energy consumed) of USAC relative to GLACSWEB in each case.

5.1 Experimental Setup

The GLACSWEB simulator lets each of the nodes take one of the following actions in a single time period: *sense*, *idle − listen* (where an agent enables its antenna so that it is ready to receive data), *transmit* a single packet, *receive* a single packet and *sleep*. With the exception of *transmit*, all actions have a set power consumption value affixed to them. The power consumption of the *transmit* action is dependent on the variable transmission power of the agent transmitter. Each such agent has five levels of transmission power to communicate with agents at different transmission ranges. The radio propagation model in the simulation was assumed to be symmetric. We decided to ignore the *processing* action of the agent due to its near negligible power consumption. Specifically table 1 shows the typical energy consumption of each action based on the values obtained from the fielded system. Furthermore, the energy capacity of the node was 77760 J and the confidence level within the sensing protocol was set to 10% (again based on our experience with the fielded system).

Table 1: Energy consumption of sensor node actions

Sensor Action	Energy consumption(Joules)
Transmission per packet	0.47
Reception per packet	0.37
Idle transceiver on	0.086
Sense	0.015
Sleep per second	0.000032

We also decided to use a simple version of the IEEE 802.11 Medium Access Control [9] where RTS/CTS[2] control packets are only used before transmission of the data packets that are sent in a burst. Also there was no difference between overhearing and receiving in terms of power consumption. However, agents were programmed to ignore packets that were not destined to them and thereafter lower their power consumption by switching to the idle transceiver mode. In order to obtain statistically significant results, we report average result and standard deviations of 200 simulations in each of the experiments carried out. The data used in our experiments are derived from segments of the data collected[3] by the fielded probes over the last year.

[2]Request-To-Send/Clear-To-Send.

5.2 Network Topology

In this experiment, we simulated the two protocols for a fixed number of agents (10) randomly distributed around the centre. The sensed data model for each agent and number of agents in the networks were fixed for each instance of the simulation. The purpose of this was to analyse how the two protocols fared against each other with the changing topology of the network. The results of these simulations are shown in the two plots in Figure 7.

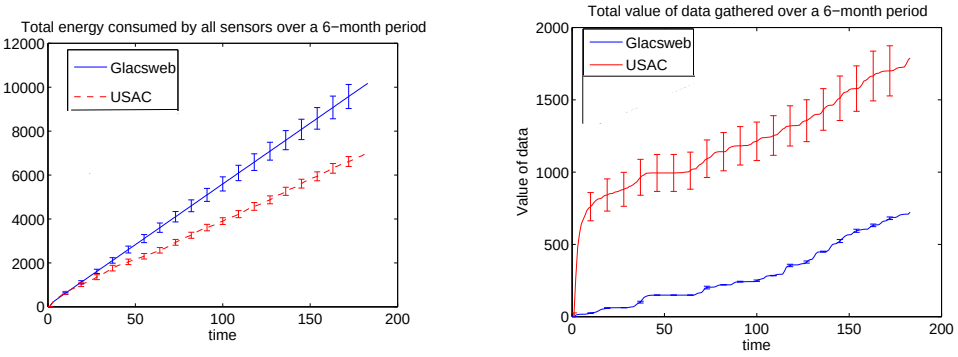
Both plots show the superiority of USAC over GLACSWEB. The big jump at the beginning of USAC's value graph is attributed to the maximum rate of sampling in the initial stages. Once a sufficient number of samples have been collected, the sampling rate is reduced and adjusted according to the change in observed data. The overall graphs of both protocols are linear. However, the gradient of USAC is steeper than that of GLACSWEB. This is because USAC increases its sampling whenever the observed data changes. The efficiency gain of USAC over GLACSWEB in this experiment was calculated to be 470%.

5.3 Network Size

In this experiment, we conducted simulations by varying the number of agents in the network from 1 to 20. Figure 8 illustrates the performance of USAC against GLACSWEB with varying network size. Due to the variability of the data, we concern ourselves with the average of the final total at the end of each simulation to ensure an effective comparison.

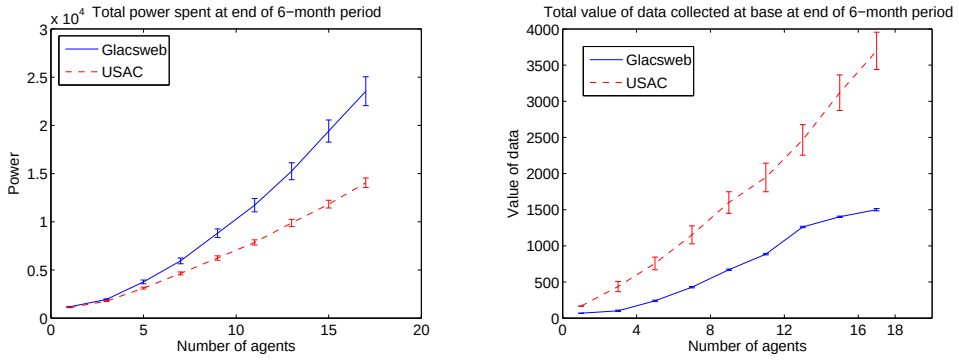
As can be seen, USAC consistently consumes less energy and still manages to collect more valued data at the sink node (even as the size of the network is increased). However, the random placement of the agent nodes results in a greater variance in the total value of data collected using USAC. This is due to the way that USAC reallocates the energy saved, when not relaying, to the task of sensing. In particular, the network topology in an instance of the experiment determines the amount of energy an agent spends on communication and therefore the amount of sensing it performs. In other words, an agent closer to the base station might relay and communicate more data since the value of the data derived might often be more than the cost of communication. On the other hand, an agent further away from the base station might be forced to communicate and relay less data because the cost of communication may now have surpassed the value of data it has sensed. The efficiency gain of USAC over GLACSWEB in this part of the experiment was calculated to be 250%.

[3]http://leo.ecs.soton.ac.uk/glacsweb/plotter.php



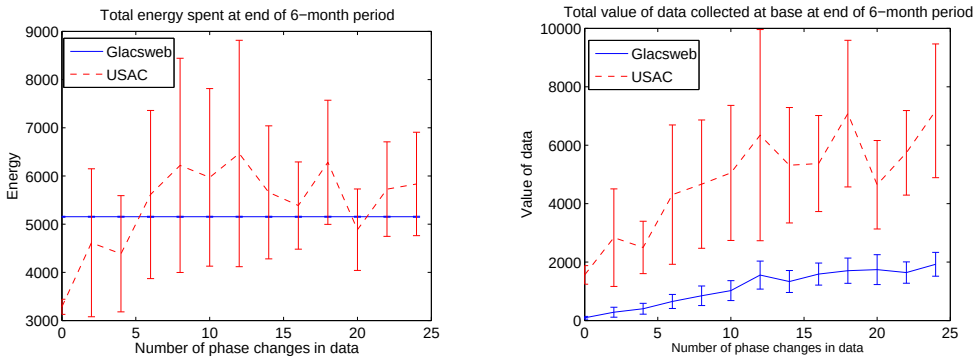
(a) Energy Consumption (b) Value Derived

Figure 7: The total energy spent and total data value gathered over a 6-month period plotted against time.



(a) Energy Consumption (b) Value Derived

Figure 8: The total energy spent and total data value gathered at the end of a 6-month period plotted against number of agents in the network



(a) Energy Consumption (b) Value Derived

Figure 9: The total energy spent and total data value gathered at the end of a 6-month period plotted against a measure of dynamism of data.

5.4 Dynamism of the Environment

In this experiment, simulations were carried out by varying the data model of the agent nodes whilst keeping the topology and size of the network constant. The degree of dynamism in the data model is defined as the number of phase changes that occur in the piecewise linear data model of the environment used by the agent. The results of these simulations are shown in figure 9.

The results from sub-figure 9(a) show that USAC adapts its power consumption based on the dynamism of the data (in contrast to GLACSWEB where the sensing protocol is static and unaffected by the type of data it is measuring). This results in the higher total value of data collected at the sink for USAC as shown in sub-figure 9(b). The high variance observed with USAC is due to the random times at which phase changes occur. Thus, if two phase changes occur very close to each other, there is a possibility of misrepresenting the changes which leads to low power consumption and low value. This experiment showed that USAC has an efficiency gain of 300% over GLACSWEB.

6. CONCLUSIONS AND FUTURE WORK

The protocol we have developed in this paper allows agents to act in a decentralised (based on the nature of their local environment) manner, while self-organising to form a network whose performance is high in terms of minimising energy consumption and maximising the value of data gained. It makes use of the localisation ability of individual agents to determine the cheapest cost path to the sink and incorporates the value of the observed data to calculate the cheapest path. We have also shown that our protocol is superior to the currently deployed GLACSWEB protocol, even when the distribution of agents around the sink and the nature of observed environment is varied.

Whilst we have specifically considered evaluating the effectiveness of our protocol in the GLACSWEB sensor network, the challenges involved here are very similar to those that occur in the design of many other sensor networks. For example, we are currently exploring the possibility of using it in the FloodNet system (a sensor network for monitoring river levels in which the sensors are solar powered)[4]. Also, in the current work, we have considered just one particular way of calculating the value of information. In future work, we wish to investigate whether more complicated information theoretic measures, such as Kullback-Liebler divergence and Mahanalobis distance, would be a better basis for measuring information and whether these more sophisticated measures would improve the system's efficiency. It would also be interesting to introduce additional factors in the simulation such as network openness and scalability (where agents come and leave the system) and extend our protocol to accommodate this aspect of dynamism in the network.

7. REFERENCES

[1] J. Byers and G. Nasser. Utility-based decision-making in wireless sensor networks. In *Proceedings of the first ACM international symposium on Mobile ad hoc networking & computing(MOBIHOC'00)*, pages 143–144, Piscataway, NJ, USA, 2000. IEEE Press.

[2] R. Dash, A. Rogers, S. Reece, S. Roberts, and N. R. Jennings. Constrained bandwidth allocation in multi-sensor information fusion: a mechanism design approach. In *Proceedings of the 8th International Conference on Information Fusion*, 2005.

[3] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-based approximate querying in sensor networks. *International Journal on Very Large Data Bases*, 2005. To appear.

[4] P. Downey and R. Cardell-Oliver. Evaluating the impact of limited resource on the performance of flooding in wireless sensor networks. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'04)*, pages 785–794, 2004.

[5] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, pages 174–185, 1999.

[6] B. Horling, R. Vincent, R. Mailler, J. Shen, R. Becker, K. Rawlins, and V. Lesser. Distributed sensor network for real time tracking. In *Proceedings of the Fifth International Conference on Autonomous Agents (AGENTS '01)*, pages 417–424, New York, NY, USA, 2001. ACM Press.

[7] V. Lesser, C. Ortiz, and M. Tambe, editors. *Distributed sensor networks: a multiagent perspective*. Kluwer Publishing, 2003.

[8] K. Martinez, J.K. Hart, and R. Ong. Environmental sensor networks. *Computer*, 37(8):50–56, 2004.

[9] LAN MAN Standards Committee of the IEEE Computer Society. *Wireless LAN medium access control (MAC) and physical layer (PHY) specification*. IEEE, New York, NY, USA, 802.11-1997 edition, 1997.

[10] P. Padhy, K. Martinez, A. Riddoch, J.K. Hart, and H.L.R. Ong. Glacial environment monitoring using sensor networks. In *Proceedings of the First Real-World Wireless Sensor Networks Workshop(REALWSN'05)*, pages 10–14, 2005.

[11] A. Rogers, E. David, and N. R. Jennings. Self organised routing for wireless micro-sensor networks. *IEEE Transactions on Systems, Man and Cybernetics (Part A)*, 35(3):349–359, 2005.

[12] A. Woo, T. Tong, and D.Culler. Taming the underlying challenges or reliable multihop routing in sensor networks. In *Proceedings of the First International Conference on Embedded Networked Sensor Systems (SenSys'03)*, pages 14–27, 2003.

[4]http://envisense.org/floodnet/floodnet.htm

Acknowledgments

We gratefully acknowledge BAE system and EPSRC funding for the ALADDIN project (EP/C548051/1) and EPSRC funding for the GlacsWeb project.

Specifying and Resolving Preferences Among Agent Interaction Patterns*

Ashok U. Mallya †
Veraz Networks
926 Rock Avenue, Suite 20,
San Jose, CA 95131, USA
amallya@veraznet.com

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
singh@ncsu.edu

ABSTRACT

A strength of commitment protocols is that they enable agents to act flexibly, thereby enabling them to accommodate varying local policies and respond to exceptions. A consequent weakness is that commitment protocols thus fail to distinguish between possible executions that are normal and those that may be allowed but are not ideal. This paper develops an approach for specifying *preferences* among executions that are allowed by a protocol. It captures sets of executions via an event constraint specification language and gives them a denotational characterization based on branching-time models. This paper develops algorithms for choosing the best execution path by considering the interplay between the preference specification of a protocol and local policies of agents interacting using the protocol, thereby giving the specifications a natural operational characterization. The value of the concepts developed is illustrated by its application to a recent practical framework for protocols called OWL-P. Further, the paper shows that the operational and denotational characterizations of preference specifications coincide.

Categories and Subject Descriptors

I..2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

1. INTRODUCTION

Commitment protocols capture the essence of the desired interactions in high-level terms. Protocols regulate the externally observable, social behavior of agents, distinguishing what is allowed from what is not. Current approaches, however, do not make any finer distinctions about what is *normal* and what is not.

This paper is about the main consequences of taking a knowledge engineering stance toward commitment protocols.

- To accommodate the openness of the given system, protocols must apply in a wide range of contexts. That is, they must generally allow multiple execution paths. For example, a purchase protocol should allow the possibility that payment might be made before delivery of goods, in which case a refund should be made if delivery is unsuccessful.
- To accommodate agent autonomy, protocols must enable the participating agents to choose their actions and responses to the above kinds of conditions as they see fit. For example, a purchase protocol should allow the possibilities that negotiations may fail, that the payment may be made via a third party, that an agent awaiting missing goods may send a reminder for them.
- To be realistic, the protocols themselves must be based on a study of different usage scenarios, wherein not all possible executions are considered equal. For example, as a practical matter, we would all recognize that it is more normal for a purchase protocol to lead to an exchange of goods and money than for a refund to be issued or for the goods to be repossessed by the merchant because of lack of payment.

In other words, protocols should be flexible enough to accommodate exceptions, but should be described in such a manner that exceptions are distinguished from normal executions. Moreover, protocol descriptions should ideally enable the specification of a hierarchy of exceptions, some being more acute (i.e., less desirable) than others.

Protocol Preferences and Agent Policies. To capture the above motivations, we propose that protocol specifications be enhanced with modular, pluggable descriptions of the *preferences* among executions of the protocol. These preferences are the protocol designer's view of what are the most normal or most desired executions. In this sense, they have a normative force. Individual agents would have their own *policies* for how they participate in a given protocol. The policies should generally be in line with the protocol preferences. (Verifying compliance with protocol preferences, however, is nontrivial—we return to this point in Section 6.)

Protocols can be refined to yield protocols that serve the same goal but impose additional requirements. For example, payment by cash is a refinement of payment (in general). Agents contemplating interacting can negotiate about the refinement of the protocol that they will enact. For example, if a merchant accepts only cash, payment by check or charge card is ruled out and payment by cash is the only kind of payment that will work.

More interestingly, agents who are participating in a protocol may negotiate about the specific actions that each would take (from among those that are allowed by the given protocol). For exam-

*This research was supported by the NSF under grant DST-0139037.

†Ashok Mallya was a full-time doctoral student at NCSU when this paper was submitted; he has since graduated.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

ple, given that two parties agree to participate as seller and buyer in a purchase protocol, they might then negotiate about whether the seller is willing to send the goods before the buyer makes a payment. Protocol preferences thus provide a basis for argumentation among the parties.

Approach. As explained in greater detail below, in our framework, protocols are denotationally characterized using sets of runs (i.e., computations), and are operationalized via translation into executable rulesets. With the above background, the approach of this paper proceeds as follows to enable augmenting protocol specifications with preferences.

- We augment an existing protocol specification language with an ability to specify preferences among different executions of protocols. For simplicity of specification, we use a simple language based on linear temporal logic to specify expressions over protocol events to identify executions that are of interest.
- We show how preferences based on such expressions can be operationalized into executable rulesets.
- We show how the above preferences are mapped into a graph whose points are sets of runs, and characterized via branching time models.
- We show that the model-theoretic and the operational characterizations coincide.

We provide the example of a typical interaction between a merchant and a customer through this paper to illustrate our approach.

Contributions. Our main contribution is in developing a new methodology for designing commitment protocols. Our methodology can be used for the following

Modeling exceptions. In traditional process models, exceptions are modeled in an ad hoc manner, when the designer demarcates blocks of the process and assigns exception handlers to those blocks. In our approach, exceptions can be defined separately from the specification of the process or the protocol. As a result, different exception conditions can be assigned to the same protocol based on the context of the protocol execution. This is similar in spirit to aspect-oriented software development.

Selecting protocols. When multiple protocols are available for an agent to realize a certain interaction, that agent can negotiate with the other parties involved in the protocol and, based on its preferences, find out if an execution of that protocol with those participants would be acceptable to it. For example, a customer agent that does not wish to enact a hotel booking protocol in which the hotel can cancel the room and award a refund can choose which hotel to interact with if its preferences are made clear before enactment and if the hotel and the customer try and negotiate the protocol to enact based on their preferences.

Organization. Section 2 introduces commitments and commitment protocols. Section 3 describes the language used to specify sets of execution sequences and its semantics. This section also describes how preferences among sets of execution sequences are specified. Section 4 gives an algorithm for resolving preferences among execution sequence sets so that agents can choose the best course of action. Section 5 lists important patterns that arise due to the interplay between local-actions, nonlocal actions, local policies, and protocol preferences. Section 6 concludes the paper with a summary of our contributions and a survey of related literature.

2. BACKGROUND

We formalize protocols as transition systems similar in spirit to commitment machines [9], which are declarative specifications. For compatibility with this model, we use the linear temporal logic-based event ordering language proposed by Singh [7] to specify preferences between sets of executions in that transition system. Although Singh's language is given semantics based on linear models, the models are related in an incremental manner to other possibilities: thus it supports a branching interpretation. We use expressions over events in this language to specify sets of runs of a protocol over which preferences can be expressed. This section briefly introduces the semantics of commitments to lay the groundwork for preference specification and introduces the language.

2.1 Commitments

A commitment $\mathsf{C}(x, y, p)$ denotes that the agent x is obliged to the agent y for bringing about the condition p. Here x is called the *debtor*, y the *creditor*, and p the *condition* of the commitment. The condition is expressed in a suitable formal language.

Commitment Operations. Commitments are created, satisfied, and transformed in certain ways. The following operations are conventionally defined for commitments, where $c \equiv \mathsf{C}(x, y, p)$: $\text{CREATE}(x, c)$ establishes the commitment c, and can only be performed by c's debtor *x*. $\text{CANCEL}(x, c)$ cancels the commitment c, and can only be performed by c's debtor *x*. Generally, cancellation is compensated by making another commitment. $\text{RELEASE}(y, c)$ releases c's debtor *x* from commitment c, and only can be performed by the creditor *y*. $\text{ASSIGN}(y, z, c)$ replaces *y* with *z* as c's creditor. $\text{DELEGATE}(x, z, c)$ replaces *x* with *z* as the c's debtor. $\text{DISCHARGE}(x, c)$ c's debtor x fulfills the commitment.

A commitment is *active* if it has been created, but not yet been operated upon by a *discharge*, *delegate*, *assign*, *cancel*, or *release*. A commitment is *satisfied* when its condition becomes true. Commitments can also be *conditional*, denoted by $\mathsf{CC}(x, y, p, q)$, meaning that x is committed to y to bring about q if p holds. A conditional commitment such as $\mathsf{CC}(x, y, p, q)$ becomes an unconditional commitment $\mathsf{C}(x, y, q)$ when its condition p holds. A commitment is *breached* when it is not possible that the commitment will be satisfied. Realistic settings assign deadlines to commitments to detect their breach or satisfaction [5].

Commitment Protocols. Commitment protocols are interaction patterns expressed in terms of the commitments that arise between the participants of the interaction. Commitment protocols have been operationalized via commitment machines. Commitment machines generate computations or *runs*, which are sequences of *states* that a valid protocol execution goes through. Propositions represent facts about the universe of discourse of the protocol such as the actions that the protocol participants have taken, commitments that have been created and operated upon, and messages that have been sent. Each state is a snapshot of the evolving state of the system (as the given agents interact), and is labeled by *propositions* that hold true there. State changes are caused by *messages* that the participants send to each other.

This paper builds on OWL-P [1, 2], which is a practical framework and an associated language for specifying, combining, and enacting commitment protocols. OWL-P (OWL for Protocols and Processes), uses OWL (Web Ontology Language) to specify protocol and protocol composition constructs. An OWL-P specification identifies roles that participate in the protocol, the messages that are exchanged (with the meanings of the messages in terms of commitments that are created), and a set of rules that constrain the set of

runs of the protocol by specifying ordering, data flow, and other constraints.

The flexibility of commitment protocols arises because they use commitments and other propositions to assign meanings to states and thus capture the essence of an interaction. Treating commitments explicitly enables further kinds of sophisticated reasoning, such as involving delegation and other kinds of manipulation of commitments. For example, a customer in a purchase protocol can delegate to a third party (such as a bank) the payment commitment that the customer has towards the merchant.

2.2 Running Example

We illustrate our approach with an example interaction in which a merchant sends some goods to a customer in exchange for a payment. The exact protocol for this exchange involves variations because the customer could pay before the merchant ships the goods and, if not satisfied, can return the item to the merchant for a refund of the payment.

3. SPECIFYING PREFERENCES

OWL-P specifies protocols declaratively and leverages the semantics of commitments and their operations to make protocols flexible. However, OWL-P does not distinguish between the normal and abnormal executions of a protocol. For example, executions in a purchase protocol that involve returns and refunds or missing shipments are treated on par with the normal executions involving no missing shipments or delayed payments. For this reason, we extend OWL-P to capture preferences among sets of executions. We first devise a language over events to concisely specify sets of protocol executions. Next, we specify preferences among such sets using this language.

3.1 The Constraint Specification Language

We use the event-based linear temporal logic $\mathcal{I}$ as introduced by Singh [7] for specifying sets of executions. We repeat here the syntax and semantics of $\mathcal{I}$ from [7]. *I* is the start symbol of the BNF for the language of $\mathcal{I}$. In this BNF, *slant* indicates nonterminals, $\longrightarrow$ and | are meta-symbols of the BNF, /* and */ begin and end comments, respectively, and all other symbols are terminals.

L_1. *I* $\longrightarrow$ *dep* | *dep* $\wedge$ *I* /*conjunction: interleaving*/

L_2. *dep* $\longrightarrow$ *seq* | *seq* $\vee$ *dep* /* disjunction: choice*/

L_3. *seq* $\longrightarrow$ *bool* | *event* | *event* $\cdot$ *seq* /* before: ordering*/

L_4. *bool* $\longrightarrow 0 \mid \top$

Dependency. A dependency is an expression generated by I. It specifies constraints on the occurrence and ordering of events.

Event Literal Set. $\Gamma \neq \{\}$ is the set of event literals as generated by the nonterminal *event.* Each event e literal has a *complement* $\overline{e}$. Intuitively, initially, neither an event nor its complement holds; ultimately, one of them must hold. Γ_D is the set of literals mentioned in a dependency D and their complements. For example, $\Gamma_e = \{e, \overline{e}\}$. For a set of dependencies $\mathbb{D}$, we define $\Gamma_{\mathbb{D}}$ as $\Gamma_{\mathbb{D}} = \bigcup_{D \in \mathbb{D}} \Gamma_D$.

In our example, we denote the payment made by the customer by p, the sending of goods by the merchant by s, the return of the goods by ret, and a refund by ref. Using these event literals, we can specify the constraint "refund can be done only after payment has been made" as a dependency $\overline{ref} \vee p \cdot ref$.

Since state labels in OWL-P capture the history of events in the system, the event-based language (and semantics) that we use here can be applied to the state-based execution semantics of OWL-P.

The formal semantics of $\mathcal{I}$ is based on runs, i.e., sequences of events. Legal runs satisfy the following requirements:

1. Event instances and their complements are mutually exclusive.
2. An event instance occurs at most once in a computation.

Universe of Runs. $\mathbb{U}_{\mathcal{I}}$ is the universe of runs; it contains all legal runs involving event instances from Γ.

3.2 Constraint Language Semantics

For a run $\tau \in \mathbb{U}_{\mathcal{I}}$ and $I \in \mathcal{I}$, $\tau \models I$ means that I is satisfied over the run τ . This notion can be formalized as follows. Here, τ_i refers to the ith item in τ and $\tau_{[i,j]}$ refers to the subrun of τ consisting of its elements from index i to index j, both inclusive. $|\tau|$ is the last index of τ and may be ω for an infinite run. We use the following conventions in the specification of semantics below: $e, f, \overline{e}, \overline{f}$, etc. are literals; D, E, etc. are dependencies; i, j, k, etc. are temporal indices; and τ, etc. are runs. The semantics of $\mathcal{I}$ is

M_1. $\tau \models e$ iff $(\exists i : \tau_i = e)$

M_2. $\tau \models I_1 \vee I_2$ iff $\tau \models I_1$ or $\tau \models I_2$

M_3. $\tau \models I_1 \wedge I_2$ iff $\tau \models I_1$ and $\tau \models I_2$

M_4. $\tau \models I_1 \cdot I_2$ iff $(\exists i : \tau_{[0,i]} \models I_1$ and $\tau_{[i+1,|\tau|]} I_2)$

Denotation. The denotation $[\![D]\!]$ of a dependency D is the set of runs that satisfy D, i.e., $[\![D]\!] = \{\tau : \tau \models D\}$.

For example, the denotation of the dependency $\overline{ref} \vee p \cdot ref$ is the set of runs in which there is no refund ($\overline{ref}$) or, if there is a refund, a payment precedes it ($p \cdot ref$).

3.3 Residuation

The residual of a dependency D by an event e is denoted by D/e and corresponds to the largest set of runs satisfying the given dependency. Formally, $\nu \in [\![D/e]\!]$ iff $(\forall v : v \in [\![e]\!] \Rightarrow (v\nu \in \boldsymbol{U}_{\mathcal{I}} \Rightarrow v\nu \in [\![D]\!]))$

The residual represents what remains in the dependency after a certain event has occurred for the dependency to be satisfied on any run. The following are residuation rules as given by Singh:

R_1. $0/e \doteq 0$

R_2. $\top/e \doteq \top$

R_3. $(E_1 \wedge E_2)/e \doteq ((E_1/e) \wedge (E_2/e))$

R_4. $(E_1 \vee E_2)/e \doteq ((E_1/e) \vee (E_2/e))$

R_5. $(e.E)/e \doteq E$ if $e \notin \Gamma_E$

R_6. $D/e \doteq D$ if $e \notin \Gamma_D$

R_7. $(e' \cdot E)/e \doteq 0$ if $e \in \Gamma_E$ where e' is any event literal

R_8. $(\overline{e} \cdot E)/e \doteq 0$

Consider the dependency $p \cdot ret \cdot ref$, which encodes "payment followed by a return followed by a refund." The residual of this dependency with p is $ret \cdot ref$, which is what is left to be done to satisfy the dependency after p occurs. The residual of this dependency with s, however, is the dependency itself, since the dependency does not specify whether s occurs.

3.4 Preferences

The specification language $\mathcal{I}$ supports dependencies to succinctly specify sets of runs. To induce a preference structure over such sets, i.e., to specify if one set is preferred over another, we introduce the *preference relation*.

Preference Relation. Let $\mathbb{R} \subseteq 2^{\mathbb{U}_\mathcal{I}} \times 2^{\mathbb{U}_\mathcal{I}}$ be the preference relation between sets of runs. $\mathbb{R}$ is irreflexive, transitive, and antisymmetric. $(S_i, S_j) \in \mathbb{R}$ means that any run in S_i is preferred over any run in S_j. Consequently, $\mathbb{R}$ is downward closed in the sense that if $(S_i, S_j) \in \mathbb{R}$, then for all subsets S_i' and S_j' of S_i and S_j, respectively, $(S_i', S_j') \in \mathbb{R}$. For convenience, we apply $\mathbb{R}$ to dependencies, meaning that their denotations are appropriately related. $(D_i, D_j) \in \mathbb{R}$ means that any run in the denotation of D_i is preferred over any run in the denotation of D_j, i.e., $\forall \tau_i, \tau_j : \tau_i \in [\![D_i]\!]$ and $\tau_j \in [\![D_j]\!], \tau_i$ is preferred over τ_j.

A commitment protocol is associated with a set of dependencies. For a protocol P specified in terms of a set of dependencies D_P, the denotation of the protocol is the set of runs that the protocol allows, and is given by $[\![P]\!] = \bigcup_{D \in D_P} [\![D]\!]$. We encode our merchant-customer example by the following dependencies

P_1. Refund only if payment has been done: $\overline{ref} \vee p \cdot ref$.

P_2. Return only if goods have been shipped: $\overline{ret} \vee s \cdot ret$.

P_3. If payment has been made and goods have been returned, then refund: $\overline{p} \vee \overline{ret} \vee ref$.

Preference Graph. A preference graph specifies preferences among a set of dependencies. Each dependency labels one node of the graph, and a preference relation specifies preferences among these dependencies, and consequently among the nodes they label. A preference graph $L = \langle D_x, R \rangle$ specifies preferences among the elements of the set of dependencies D_x using the partial order induced by R over D_x. For the above L, we define its event literal set to be the set of all events that are mentioned in the dependencies in D_x, and their negations. $\Gamma_L = \bigcup_{D \in D_x} \Gamma_D$.

Figure 1 shows a schematic representation of an example preference graph. This graph encodes the following:

D_1. The most preferred runs are those in which payment and shipping occur, and a return or a refund do not.

D_2. Less preferred than runs in D_1 are runs in which goods are returned and payment is refunded.

D_3. Also less preferred than runs in D_1 are runs in which the payment is refunded and the goods are returned.

D_4. Less preferred than all of the above are runs in which goods are returned, but no refund occurs.

D_5. Also preferred less than D_1, D_2, and D_3 are runs in which the goods are shipped, a refund is made, but the the goods are not returned.

Preference Node Denotation. The denotation of a preference node N_D labeled by D, with respect to a protocol P and a graph $L = \langle D_P, R \rangle$, where $D \in D_P$, is the set of runs allowed by the protocol that are also allowed by D, but not by nodes in L that are preferred over N_D. The motivation for the above is that each run can occur in the denotation of at most one node in the graph. Thus if a given run is allowed by two sets of runs, one preferred over the other, the run from the less preferred set would not "get credit" for this run. On each path from the top to the bottom of the graph, a given run can occur at most once. Formally, $[\![N_D]\!] = [\![P]\!] \cap ([\![D]\!] - \bigcup_{(D_i, D) \in R} [\![D_i]\!])$.

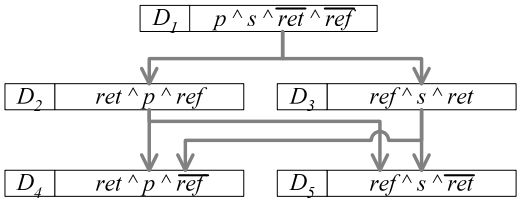


Figure 1: A preference graph for our example, $L = \langle \{D_1, D_2, D_3, D_4, D_5\}, \{(D_1, D_2), (D_1, D_3), (D_2, D_4), (D_3, D_4), (D_2, D_5), (D_3, D_5)\} \rangle$.

Consider the preference node formed from D_4. Since the protocol contains no runs in which the payment has been made, goods have been returned, and the refund does not occur, $[\![D_4]\!] = \{\}$. However, the protocol allows runs in which the payment is refunded, the goods have been shipped, but are not returned. The preference node formed from D_5 captures such runs as the least preferred runs in the protocol.

4. RESOLVING PREFERENCES

How can an agent operationalize a preference graph during protocol enactment?

4.1 Evolution of Graphs

A preference graph evolves as events occur, causing the state of the protocol to change. The evolution changes the denotation of each node of the graph because events residuate dependencies that label graph nodes. Below, we describe, and give an algorithm for, the evolution of a (preference) graph. An *evolution tree* captures the evolution of a graph. Below, $\tau + e$ concatenates event e to run τ.

Evolution Tree. An evolution tree $\mathbb{T}_L$ of a graph L represents all possible states that L can evolve through as a result of events in Γ_L. An evolution tree is a tuple, $\mathbb{T}_L = \langle \mathbb{V}, V_{\tau_0}, \mathbb{E}, L \rangle$, where $\mathbb{V}$ is a set of *evolution* nodes, V_{τ_0} is a distinguished root node, $\mathbb{E} \subseteq \mathbb{V} \times \mathbb{V}$ is a set of *evolution edges*, and L is a preference graph.

Evolution Node. An evolution node captures a state of a preference graph. An evolution node corresponds to a *path* in the evolution tree. This path is a legal run and is the sequence of the events that occurred for the evolution node to be generated, starting from the root of the evolution tree. Evolution nodes are represented as $V_\tau = \langle L, \tau \rangle$, where L is a graph (as evolved from the original preference graph) and τ is the path to V_τ from the root node of the evolution tree. The root node of an evolution tree T_L is $V_{\tau_0} = \langle L, \tau_0 \rangle$, where τ_0 is the zero-length run.

Evolution Edge. An edge labeled e that originates at V_τ ends at $V_{\tau+e}$. Thus edges are written as $\langle V_\tau, e, V_{\tau+e} \rangle$.

Given a graph L, we construct the evolution tree $\mathbb{T}_L$ for it by first creating the root evolution node $V_{\tau_0} = \langle L, \tau_0 \rangle$. From a node V_τ, we create an edge labeled e for each event e in Γ_L that does not occur in τ. For each edge e, we create a new evolution node $V_{\tau+e} = \langle L', \tau + e \rangle$. For this new node, we set $L' = \langle D', R' \rangle$, where D' is the set of nonzero dependencies obtained by residuating each dependency in L by e, and R' relates two dependencies in D' if the dependencies corresponding to them in D were related by R. That is, we preserve the preferences among the residual dependencies in the evolution nodes. Algorithm 1 generates an evolution tree of a specified finite depth.

Figure 2 shows parts of the evolution tree generated using Algorithm 1 on the graph shown in Figure 1. In this figure, end denotes a special event signaling the end of the protocol. The complements of events that have not occurred are asserted when end occurs. In the figure, for example, since ret and ref have not occurred, their complements $\overline{ret}$ and $\overline{ref}$ are asserted when end occurs.

input : A preference graph $L = \langle D_x, R \rangle$, and a limit MAX_DEPTH for the depth of the tree generated
output: An evolution tree $\mathbb{T}_L = \langle \mathbb{V}, V_{\tau_0}, \mathbb{E}, L \rangle$ of L

Create the root node $V_{\tau_0} = \langle L, \tau_0 \rangle$;
depth $\leftarrow 1$;
$V \leftarrow V_{\tau_0}$;
$\mathbb{T}_L.\mathbb{V} \leftarrow \{V\}$;
$\mathbb{T}_L.\mathbb{E} \leftarrow \{\}$;
siblings $\leftarrow \{V\}$;
while *depth* $<$ *MAX_DEPTH* && *siblings* $\neq \{\}$ **do**
  foreach *evolution node* $V \in$ *siblings* **do**
    siblings $\leftarrow$ siblings $-\{V\}$;
    foreach *event* $e \in \Gamma_L$ *such that* $e \notin V.\tau$ **do**
      $V_{V.\tau+e} \leftarrow$ `createChildNode` (V, e) ;
      $\mathbb{T}_L.\mathbb{V} \leftarrow \mathbb{T}_L.\mathbb{V} \cup \{V_{V.\tau+e}\}$;
      $\mathbb{T}_L.\mathbb{E} \leftarrow \mathbb{T}_L.\mathbb{E} \cup E$;
      if $V_{V.\tau+e}.L.D \not= \{\}$ **then**
        siblings $\leftarrow$ siblings $\cup \{V_{V.\tau+e}\}$;
  depth $\leftarrow$ depth $+1$;
return $\mathbb{T}_L$;

Procedure `createChildNode` (V, e): Create a child evolution node for the node V on event e
input : An evolution node $V = \langle L, \tau \rangle$, where $L = \langle D, R \rangle$, and an event e.
output: An evolution node that represents the state of L on event e.
$D' \leftarrow \{\}$;
$R' \leftarrow \{\}$;
foreach *dependency* $d \in V.L.D$ **do**
  if $d/e \not= 0$ **then**
    $D' \leftarrow D' \cup (d/e)$;
    foreach *preference* $(d_i, d_j) \in V.L.R$ **do**
      if $d == d_i$ ***or*** $d == d_j$ **then**
        $R' \leftarrow R' \cup (d_i, d_j)$;
$L' = \langle D', R' \rangle$;
$V_{V.\tau+e} = \langle L, V.\tau + e \rangle$;
return $V_{V.\tau+e}$

Algorithm 1: GenerateEvolutionTree(L, MAX_DEPTH): Compute the evolution tree for a graph $L = \langle D_x, R \rangle$, where MAX_DEPTH is a bound on the depth of the tree to be generated.

4.2 Choosing Between Runs

Consider an agent X operating within the preference graph specified for a protocol in which it is participating. For such an agent to determine, the action to take (there is an event corresponding to every action instance), each evolution node is a decision point. X chooses the event would that take the protocol to the most preferred preference graph node. Since preference nodes are distributed among the evolution nodes, and X cannot change the state of the protocol at a finer granularity than that of evolution nodes, X determines which evolution node is the most preferred. To determine this, a *preference graph of evolution nodes* is constructed as follows: At any evolution node, to decide the next action to take, induce a graph among the child evolution nodes using the original preference structure, i.e., the preference structure specified in the root of the evolution tree. Once this is done, induce a graph on the evolution nodes based on the following criterion. Given two evolution nodes V_1 and V_2, create an edge from V_1 to V_2 (i.e., mark that V_1 is preferred over V_2) if and only if there is a nonzero dependency D_1 in V_1 which is preferred over all nonzero dependencies in V_2, and no nonzero dependency in V_2 is preferred over D_1. Formally,

$$(V_1, V_2) \in \mathbb{O} \quad \Leftrightarrow \quad \exists D_1 \text{ in } V_1 : \forall D_2 \text{ in } V_2, \\ (D_1, D_2) \in \mathbb{R} \text{ and } (D_2, D_1) \notin \mathbb{R}$$

where $\mathbb{O}$ is the preference relation over evolution nodes. $\mathbb{O}$ is irreflexive, antisymmetric, and transitive by the above definition. Algorithm 2 shows how a preference graph can be constructed between evolution nodes. Applying this algorithm to the children of the root evolution node V_{τ_0} in Figure 2, we obtain the evolution node graph shown in Figure 3. Figure 4 shows the same graph in detail. Since V_p contains the dependency D_1/p which is preferred over all dependencies in V_{ret} and no dependency in V_{ret} is preferred over D_1/p, we create an edge from V_p to V_{ret}. Similar reasoning for all four evolution nodes yields the graph of Figure 3.

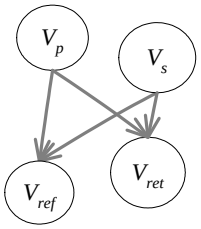


Figure 3: The preference graph for the evolution nodes V_p, V_s, V_{ret}, and V_{ref} shown in Figure 2, based on the preference graph L shown in Figure 1

Selecting the Best Path. Given the graph of preferences among the child (evolution) nodes, the best action to take (event to bring about) is the event that leads to a node in the evolution node graph over which no other node is preferred. However, the evolution node preference graph might have many such nodes. The agent can then choose any one of these. Section 5 discusses some criteria for path selection.

4.3 Types of Events

So far, we have discussed a single agent's perspective. In agent interaction, however, events are brought about by different agents during the enactment of a protocol. Choosing the best path, if one is available, is therefore not up to a single agent. We therefore recognize two types of events, *observed* and *controlled*.

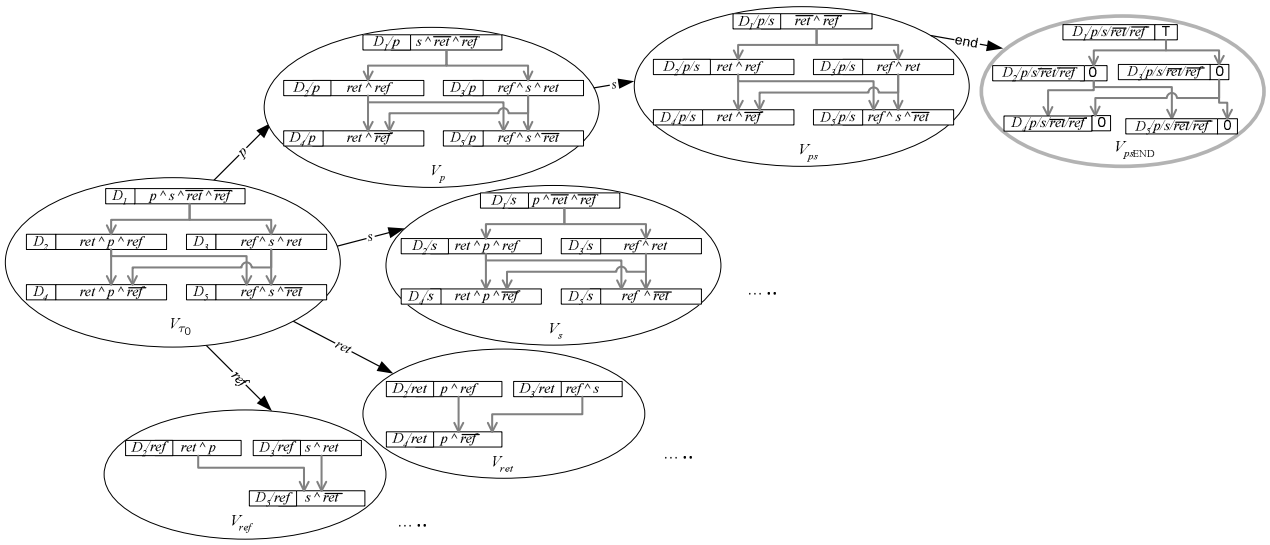


Figure 2: A partial evolution tree for the preference graph shown in Figure 1. Note that the denotations of preference nodes are not taken into consideration in this figure, so that we can illustrate the generation of the evolution nodes clearly. If denotations were taken into account, $[\![D_4]\!] = \{\}$, and this figure would have fewer dependencies. END is a special event signaling the end of the run.

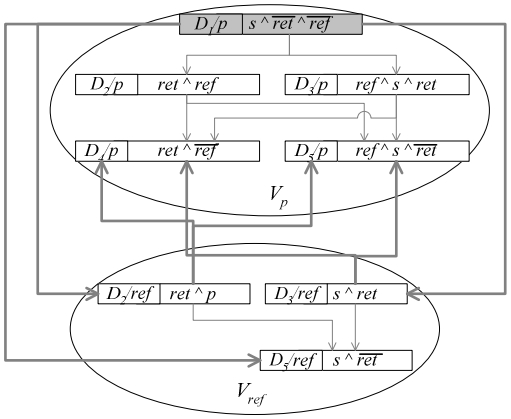


Figure 4: A detailed view of a part of the evolution node preference graph shown in Figure 3. Grey arrows indicate the preference relation. V_p is preferred over V_{ref} because of D_1/p, which is shown in a grey rectangle.

Controlled Events. For a given agent, an event that it can bring about is called a controlled event. We introduce a distinguished controlled event, Noop, which means "do nothing" (in this protocol). An agent can perform a Noop to wait.

Observed Events For a given agent X, an event which X can observe, but not bring about, is called an observed event. We distinguish two unique observed events, start and end, which are observed by agents participating in a protocol when the protocol begins and ends, respectively.

Whether an event is observed or controlled is decided when the events are described for the system being modeled. In our example, payment (p) and return (ret) are controlled events for the customer, and observed events for the merchant, whereas sending goods (s) and refund (ref) are controlled events for the merchant and observed events for the customer.

5. PATTERNS FOR PATH SELECTION

Given that an agent will make a choice at every node of the evolution tree, and that events can be controlled or observed, we identify typical patterns of decision-making that agents can adopt in common situations. Before describing these patterns, we explain rule-based protocols of the type found in the OWL-P framework.

5.1 Rule-Based Protocols

We envision the application of our algorithms in a rule-based protocol enactment framework, such as OWL-P. OWL-P enables agents to specify their *local policies* independent of the protocol specification. The general structure of rules in such protocols is the Event-Condition-Action (ECA) structure,

$$\begin{array}{c}\text{On } e\\ \text{if } \textit{localPolicy}(e, x)\\ \text{then do } a(x)\end{array}$$

Here, e is an event and $a(\cdot)$ is an action (with a corresponding event e_a), which depends on the local policy for its parameters.

As an example, consider a purchase protocol which requires that the receiver of a $\textit{rfq}(c, m, \textit{itemID})$ message (which is a request for a price quote for an item)–the merchant m–respond the the sender–the customer c–with a $\textit{quote}(m, c, \textit{itemID}, \textit{price})$ message (which is a price quote). The rule for this requirement will be

$$\begin{array}{c}\text{On } \textit{rfq}(c, m, \textit{itemID})\\ \text{if } \textit{localPolicy}(\textit{rfq}(c, m, \textit{itemID}), \textit{price})\\ \text{then do } \textit{quote}(m, c, \textit{itemID}, \textit{price})\end{array}$$

Where multiple choices of action are afforded by the protocol on the same triggering event, there will be one rule for each action, with the same event and condition. In such a case, the local policy decides which rule to enable. In case the policy enables multiple rules, the preference specification of the protocol is used to decide which action to take. Based on these principles, we describe patterns of decision-making that agents can use.

5.2 Clear Choice

If an agent has to choose between paths in an evolution tree, and exactly one most-preferred node exists in the evolution node preference graph, the agent takes the path to that node. This situation is shown in Figure 5, and is termed *Clear Choice* .

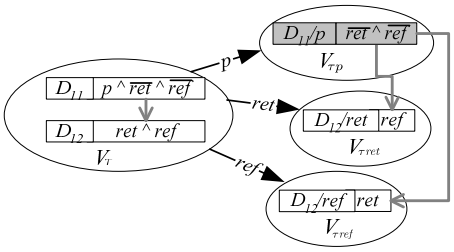


Figure 5: Clear choice: An agent can unambiguously decide which path to choose, since the evolution node preference graph has exactly one node $V_{\tau p}$, over which no other node is preferred.

Using Noop. Sometimes, even when there is a clear choice, an agent might not take any action because of its local policy. Consider the situation shown in Figure 6. This is a variant of our example in which the customer can send an rfq, which is a request for a price quote, to the merchant, to which the merchant can reply with q, which is a quote. At V_τ, the customer c can send either an *rfq* or a payment. According to the preference graph (dictated by the protocol), *rfq* is preferred over payment. However, the customer might not wish to send an *rfq*. Further, instead of performing the only other alternative, payment (p), the customer might want to wait, hoping for a quote (q), or a shipment (s) from the merchant. Since the protocol does not require that c act immediately (most realistic protocols have a notion of timeouts and deadlines), c can perform a Noop, which is do nothing, waiting till its policy allows it to perform an *rfq* or till the protocol forces c to perform p (which will also be a policy decision, if the policy has been designed to not violate the protocol).

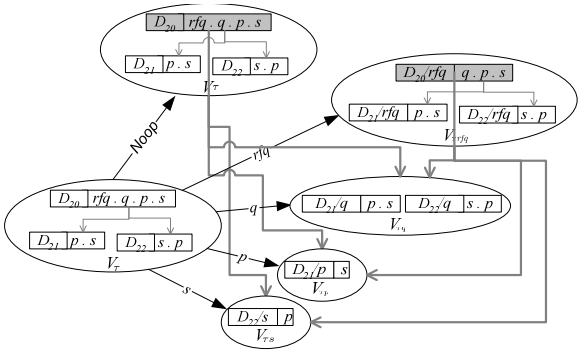


Figure 6: Using a Noop when local policy does not allow the best choice and the protocol does not impose immediate deadlines on the other choice(s). rfq is the best choice for the customer, followed by p, but the the customer chooses to wait using a Noop. This creates two most-preferred nodes, V_τ and V_{rfq}.

5.3 Unclear Choice

The choice of action to take (or event to generate) will be unique if there is exactly one evolution node over which no other node is preferred. However, this is not always the case. Although there will never be a cycle among the preference relations between the evolution nodes (straightforward proof by construction of the evolution node preference graph), it is possible that there are multiple most-preferred nodes, as in Figure 3, where both V_p and V_s are most-preferred, i.e., the customer can either pay or wait for the merchant to send the goods. Since the choice here is between a controlled and an observed event, we call this situation *Unclear Nonlocal Choice*. In such a case, the agent may choose to wait for other agents participating in the protocol to take some action.

There can also be cases in which two or more controlled events are most-preferred (*Unclear Local Choice*). An agent will rely on its local policy to make a choice when this happens. In the Noop example, if there were no preference between D_{20} and D_{21}, the customer would have to decide whether to send an *rfq* or to send a payment. In OWL-P, such local policy-based decisions drive the protocol by enabling agents choose one run from among the many runs that a protocol allows.

6. DISCUSSION

Protocols help us capture agent interactions perspicuously. The very strength of protocols, namely, their flexibility, can pose a challenge. Executions allowed so as to make a protocol flexible under exceptional conditions can begin to be selected over superior executions even when the exceptional conditions do not obtain. This paper proposes that protocols be specified generically, and preference conditions among different sets of runs be stated separately, depending on the context of usage of the protocol. Preferences serve as a rough-and-ready means to capture design goals wherein the normal executions are preferred, yet abnormal executions arising from exceptions or unexpected actions by some of the agents are allowed.

This paper identified a denotational semantics for preference conditions understood as a graph. Next, it showed how such a graph evolves as events and actions take place. It proposed a simple method by which an agent can decide upon its next action, describing an anomalous condition and giving an algorithm for identifying it.

Preferences are often understood solely in decision-theoretic terms. Decision theory is clearly important for modeling preferences and coming up with strategies for acting according to them. However, a denotational understanding of preferences can support the proper application of decision theory.

Related Literature. Yolum and Singh [10] developed one of the first operationalizations of commitments. They specified protocols by listing legal states in terms of the commitments and domain propositions that hold at that state, and using an event calculus planner to generate the set of runs that were allowed. Winikoff *et al.* [9] advance this line of research. Fornara and Colombetti have also proposed a commitment-based interaction protocol framework [3]. However, none of the above approaches specify or operationalize a notion of preferences among the various execution sequences allowed by a protocol. Our work, therefore, is a significant step in this direction. Grosof *et al.* [4] implement rule based agent interaction systems where rules are prioritized. Grosof and colleagues propose *Courteous Logic Program*s, or CLPs. In a CLP, when there is ambiguity regarding which rule to fire, i.e., a conflict arising because multiple rules can be fired at a particular state of the world, the priorities assigned to the rules are used to resolve the conflict. Our work is similar to CLPs in this respect, but different in that we propose a scheme in which preferences among *runs* are specified, independent of a protocol specification. Further, we also present a methodology for translating these preferences into rules that can be

embedded into the (rule-based) protocol specification.

Our work is based on the concept of social interaction among agents, which gives importance only to the observable behavior of agents. We describe how preferences among runs can exist. However, we do not study how agents can reason about the benefits of using one set of protocol runs over another. Pasquier and Chaib-Draa [6] introduce the cognitive dissonance theory into multiagent communication by incorporating the theory and dialogue game protocols into agent interactions. Their theory explores ways in which agents can decide when to start dialogues with other agents and what kind of dialogues to initiate, among other things. This line of research is complementary to and would strengthen the interaction framework we have presented here. Preferences among the available runs of a protocol have also been studied from the game theoretic point of view by Otterloo *et al.* [8]. They describe a logic that can be used for reasoning about a strategy to adopt in a game when the preferences of other agents in the game are known. The work differs from ours because of the use of games instead of commitment protocols. Also, preferences of agents are assumed to be known by other agents, which does not always apply in real-world applications such as the business interaction we have outlined in this paper. We plan to incorporate such reasoning among agents into our framework.

input : An evolution node V_τ, and a root preference graph $L_r = \langle D_r, R_r \rangle$
output: A graph $L_e = \langle V_e, R_e \rangle$ representing the preferences among the child nodes of V_τ.

1 $V_e \leftarrow$ All child nodes of V_τ;
2 $R_e \leftarrow \{\}$;
3 **foreach** *child node* $V_{\tau+x}$ *of* V_τ **do**
4 **foreach** *child node* $V_{\tau+y}$ *of* V_τ **do**
5 **foreach** *dependency* d_x *in* $V_{\tau+x}$ **do**
6 isPrefOverAll $\leftarrow$ true ;
7 **foreach** *dependency* d_y *in* $V_{\tau+y}$ **do**
8 **if** (`getRootDep` (d_x, $V_{\tau+x}$, D_r), `getRootDep` (d_y, $V_{\tau+y}$, D_r)) $\notin R_r$ ***or*** (`getRootDep` (d_y, $V_{\tau+y}$, D_r), `getRootDep` (d_x, $V_{\tau+y}$, D_r)) $\in R_r$ **then**
9 isPrefOverAll $\leftarrow$ false ;
10 **if** *isPrefOverAll* **then**
11 $R_e \leftarrow R_e \cup (V_{\tau+x}, V_{\tau+y})$;
12 break ;
13 **return** $\langle V_e, R_e \rangle$;
14 **Procedure** `getRootDep` ($d, V_{\tau'}, D_0$): Find which dependency among D_0 node was residuated to d in node $V_{\tau'}$
input : A dependency d, its evolution node $V_{\tau'}$,and a set of dependencies D_0.
output: A most-preferred dependency in D_0 that was residuated to d along path τ'; **null** if no such dependency exists.
15 **foreach** $d_0 \in D_0$ **do**
16 $d_t \leftarrow d_0$;
17 **for** $i \leftarrow 0$ ***to*** $|\tau'| - 1$ **do**
18 $d_t \leftarrow d_t / \tau'_{[i]}$;
19 **if** $d_t == d$ **then**
20 $D_{root} \leftarrow D_{root} \cup \{d_t\}$;
21 **foreach** $d_r \in D_{root}$ **do**
22 isPrefOverAll $\leftarrow$ true ;
23 **foreach** $d'_r \in D_{root}$ **do**
24 **if** $(d_r, d'_r) \notin \mathbb{R}$ **then**
25 isPrefOverAll $\leftarrow$ false ;
26 break ;
27 **if** *isPrefOverAll* **then**
28 **return** d_r ;
29 **return null** ;

Algorithm 2: GenerateEvolutionGraph(V_τ, L_r): Compute the preference graph among child nodes of an evolution node V_τ given the preference graph L_r of the root evolution node.

7. REFERENCES

[1] N. Desai, A. U. Mallya, A. K. Chopra, and M. P. Singh. OWL-P: A methodology for business process modeling and enactment. In *AAMAS AOIS Workshop*, July 2005.

[2] N. Desai, A. U. Mallya, A. K. Chopra, and M. P. Singh. Interaction protocols as design abstractions for business processes. *IEEE Trans. Software Engg.*, 31(12):1015–1027, 2006.

[3] N. Fornara and M. Colombetti. Defining interaction protocols using a commitment-based agent communication language. In *Proc. 2nd AAMAS*, pp. 520–527. ACM Press, July 2003.

[4] B. N. Grosof and T. C. Poon. SweetDeal: Representing agent contracts with exceptions using XML rules, ontologies, and process descriptions. In *Proc. WWW 2003*, pp. 340–349.

[5] A. U. Mallya, P. Yolum, and M. P. Singh. Resolving commitments among autonomous agents. In F. Dignum, editor, *Advances in Agent Communication*, vol. 2922 of *LNAI*, pp. 166–182, Berlin, 2003. Springer-Verlag.

[6] P. Pasquier and B. Chaib-Draa. The cognitive coherence approach for agent communication pragmatics. In *Proc. 2nd AAMAS*, pp. 544–551. ACM Press, July 2003.

[7] M. P. Singh. Distributed enactment of multiagent workflows: Temporal logic for web service composition. In *Proc. 2nd AAMAS*, pp. 907–914. ACM Press, July 2003.

[8] S. van Otterloo, W. van der Hoek, and M. Wooldridge. Preferences in game logics. In *Proc. 3rd AAMAS*, pp. 152–159. ACM Press, July 2004.

[9] M. Winikoff, W. Liu, and J. Harland. Enhancing commitment machines. In *Proc. AAMAS DALT Workshop*, 2004.

[10] P. Yolum and M. P. Singh. Flexible protocol specification and execution: Applying event calculus planning using commitments. In *Proc. 1st AAMAS*, pp. 527–534. ACM Press, July 2002.

An Ontology of Social Control Tools

Philippe Pasquier
DAMAS Laboratory
Laval University
G1K 7P4, Québec, Canada
pasquier@iad.ift.ulaval.ca

Roberto A. Flores
Christopher Newport University
Newport News, VA 23606
flores@pcs.cnu.edu

Brahim Chaib-draa
DAMAS Laboratory
Laval University
G1K 7P4, Québec, Canada
chaib@iad.ift.ulaval.ca

ABSTRACT

In multi-agent systems, social commitments are increasingly used to capture roles, social norms, the semantics of agent communication as well as other inter-agent dependencies. Those systems rest on the assumption that agents respect their commitments. In this paper, we present an ontology of sanctions and punishment philosophies which are required ingredients of any social control mechanism susceptible to fosters agents' compliance with the commitments they create.

1. INTRODUCTION AND MOTIVATIONS

A multi-agent system (MAS) is considered an *open* MAS if the following properties hold [5]: (1) agents' behavior and interactions cannot be predicted in advance; (2) agents' internal architecture is not publicly known and (3) agents do not necessarily have common goals, desires or intentions.

The first of those properties implies that the execution of open MAS is *non-deterministic*. Open societies are usually subject to unanticipated outcomes in their interactions. The second property implies that an open MAS can have members with different internal architectures; therefore, they can be *heterogeneous*. The third property implies that the members of an open society may be non-benevolent, non-cooperative or even insincere. In addition, the agents may fail to, or choose not to, conform to some of the normative aspects of the MAS in order to achieve their individual goals. In that context, providing the means and tools for the achievement of a chosen/emergent social order in such system is a challenging issue.

Social commitment has been introduced as a public and social primitive that allows to capture all social and normative dimensions of MAS, including: social norms, roles and the semantics of communication as well as other interagent dependencies. One of the key feature of those public and social primitives is their flexibility, which allow to take into account the dynamic of the environment. Most approaches assume that the agents will respect their social commitments (thus applying regimentation). This assumption is unrealistic since unintended violation is likely to occur and unilateral cancellation as well as commitment modification are desirable.

In a perspective of knowledge transfer, this short paper presents an ontology of social control mechanisms that can be used to support the *enforcement of flexible social commitments* in open systems.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. ONTOLOGY OF SANCTIONS AND SOCIAL CONTROL MECHANISMS

Introduced in sociology as early as the end of the 19th century, the concept of social control originally denoted the capacity of a group or society to regulate itself and to secure coherency and unity in social life [7]. Social control, in this sense, relates to how social action is coordinated toward a chosen or an emergent social order. Often seen as all-encompassing, practically representing any phenomenon leading to conformity or as a broad representation of regulated mechanisms placed upon society's members, social control can be viewed as the glue holding society together [4].

Modern theories of social control focus on the strategies and techniques that help to regulate agent behavior, and lead to conformity and compliance with the rules of society (at both the macro and the micro levels). In the remainder of this section, we detail the main elements used in the enforcement of social commitments: (1) sanctions, which are considered in their general sense of incentives (the next section presents an ontology of sanctions along their different dimensions), and (2) philosophies of punishment (Section 2.2), which result in punishment strategies determining the type of sanction (and its magnitude) to be applied, and explains how sanctions are assigned to social commitments.

2.1 Sanctions

In this paper, we only consider *individual sanctions* and, for simplicity, leave aside other types of sanctions, such as *collective sanctions* [6] (which may be associated to teams, roles or groups of agents). In the next subsections, we go through the three main dimensions of sanctions: direction, type and style.

Sanction Directions

Sanctions have a specific direction. It is usually useful to consider both:

- *positive sanctions*: positive sanctions are rewards that encourage a continuation of desired behavior. For ex-

ample, it is common in open systems that agents accept committing to a task only if the associated reward is worth pursuing.

- *negative sanctions*: on the other hand, negative sanctions are used to discourage norm violating behavior. For example, agents that cannot fulfill their commitments are expected to be punished.

In brief, positive sanctions are incentives to pursue a particular behavior while negative sanctions are incentives against its violation. For the sake of simplicity, in the rest of the paper we use the words sanction to denote negative sanctions, and reward to denote positive rewards

Sanction Types

The first sanction type is *automatic sanctions*, which arises when the violators action carries its own penalty (e.g., because it is not being coordinated with the actions of others). For example, someone who drives on the wrong side of the road has a higher than normal probability of crashing into another car. We will not consider these unintended (since no one decides that they should apply) sanctions in the scope of this paper.

Within the vast literature addressing this topic from various perspectives including economics, criminology, sociology, social psychology, AI and MAS, we encounter three broad types of non-automatic sanctions: (1) material sanctions, (2) social sanctions, and (3) psychological sanctions.

Material sanctions include physical sanctions like physical retaliation or repairing actions, as well as financial sanctions like fees. Material sanctions can be applied immediately at the time of occurrence or be delayed through time.

There are *social sanctions* as well. Trust, credibility and reputation are social values that could be affected by social sanctions. As pointed out in [9], social sanctions are usually the effects of some implicit informational disclosure where the violator's action conveys information about himself that he would rather not have others know. For example, that an agent cancels or modifies a big number of its commitments without any explicit reason might be taken into account by the other agents when evaluating his reputation and the trust they put in him.

Psychological sanction types, which may be more useful in believable agents [1] and which have been used in advanced mono-agent design in mixed communities, can be important as well. Examples of psychological sanctions are guilt (where the violator feels bad about his violation as a result of his knowledge of social norms, quite apart from external consequences), and shame (where the violator feels that his action has lowered himself either in his own eyes or in the eyes of other agents).

The time horizon of sanctions indicates whether the effects of sanctions are long-lasting or short-lived. This concept is important since some sanction types may extend through time (e.g., trust, reputation, credibility) while others may not (e.g., immediate material sanctions). Subtle and complex phenomena, like forgiveness, can require taking into account this time issue.

Sanction Styles

For the specific formal needs of MAS, we distinguish two sanctions styles: implicit and explicit. *Implicit sanctions* are "autonomously" and unilaterally decided by agents. The major difficulty associated with implicit sanctions is that they are not publicly known and agents have to discover whether or not they have been sanctioned (for example, by noticing that others do not communicate with them anymore). On the contrary, *explicit sanctions* are publicly known (at least among the interacting agents).

Another useful distinction can be made between *a priori* decided sanctions and *a posteriori* decided sanctions. In particular, a posteriori decided sanctions do not allow agents to reason about the pros and cons of respecting their commitments. That the punished agent can disagree with the sanction assigned a posteriori may lead to litigation. The complexity associated with litigation makes a posteriori sanctions less desirable than a priori sanctions in open MAS.

In the remainder of the paper, we will consider only a priori defined explicit sanctions. Among a priori known explicit sanctions, we can distinguish *static*, a priori known, explicit, sanction systems provided to all agents at design time, and *dynamically* decided, a priori, explicit sanctions, which are negotiated by the agents through their communications.

2.2 Punishment Policies

Social control mechanisms to enforce social commitments should be designed according to a philosophy of punishment. By punishment, we mean the imposition of sanctions to satisfy open system designers' desire for retribution against wrongdoers. According to social control theorists, there are five different philosophies of punishment from which all *punishment policies* can be derived [10]:

- *Deterrence*: issued from the classical school of criminology, and supported by philosophers like Beccaria [2] and Bentham [3], deterrence is a utilitarian principle stating that the aim of sanctions is to prevent future violation. For deterrence to be effective, punishment must be swift, certain and severe. Applied to the enforcement of social commitment in MAS, it means that commitments should be associated with heavy and explicit sanctions. This extreme position, i.e. using severe sanctions with a high prohibitive effect, tends to transform social commitments into strict obligations, losing part of the flexibility objective desired for commitments.

- *Retribution*: retribution considers that the violation should be repaired by a penalty as severe as the wrongful act.

- *Incapacitation*: incapacitation considers the impairment or restriction of the agent's ability to commit the violation again. In human societies: exclusion, firing, and imprisonment, are the most common methods of incapacitation. Applied to the enforcement of social commitment in MAS, this punishment philosophy may result, for example, in the exclusion of wrongdoers from the system (at least for a certain time). As in human societies, it could have a preventive effect toward others violations, but it is not sure that it could be practical for artificial agents. Although, it is a consideration that is system-dependant, incapacitation usually results in the loss of opportunity during the time of the punishment, a circumstance that could be considered as a material sanction. For example,

in e-business systems, losing activity time usually has material consequences. Since it is reducible to some material explicit sanction, we will consider incapacitation as a special case of the retribution philosophy in the rest of the paper.

- *Rehabilitation*: rehabilitation is the process of correcting from erroneous behavior and returning to a rightful course. This philosophy seems more difficult to implement in MAS since it would require policies to notify the entity responsible for wrongdoers that its decision-making mechanism must be corrected. Moreover, this philosophy would suppose pro-social learning capabilities that cannot be imposed in an heterogeneous open system.

- *Restoration*: restoration attempts to make the victim and community "whole again" through rituals. This punishment mainly occurs in less industrialized countries and will not be treated here since it requires social and cultural dimensions that are not yet considered in MAS.

Since punishment philosophies like incapacitation, rehabilitation and restoration focus on the choice of sanctions types and styles rather than on the choice of sanctions strength, we will restrict our analysis to the two remaining philosophies: deterrence and retribution. While both of them seem adequate for open MAS, we will argue that retribution is a practically better choice for open MAS.

Indeed, the last decades of work in economics and law provide two basic reasons why it is best for sanctions to equal harm[1]. Here, we reformulate these arguments toward retribution punishment policies using MAS terminology.

The first argument concerns the *level of precautions* taken by parties, where the term "precautions" is to be interpreted generally. If sanctions are less than harm, precautions will tend to be inadequate and agents will tend to not respect adopted social commitments when it is to their advantage to do so. Symmetrically, if sanctions exceed harm, precautions will be excessive and may preclude agents committing to wanted commitments (this is the case with the deterrence punishment philosophy). For example, even if sincerely wanting to, an agent will not commit to a course of action if the sanctions attached to violation (which may occur unintentionally) are prohibitive. However, it has been shown that if sanctions equal harm, agents will have socially correct incentives to take precautions [8].

The second reason why it is desirable for sanctions to equal harm involves the agents' *level of activity*, that is, the extent to which agents participate in risky activities. An agent's level of activity affects the magnitude of expected total harm, independently of the precautions taken when engaging in an activity. For example, the more commitments an agent takes (its level of activity), the greater the possible number of accidents (violations) will occur, independently of the safety features of the agent (which affect the expected harm per commitment) [8].

It is worth noticing that concluding that damages should equal harm would require making two assumptions. The first assumption is that agents are *risk neutral*. If injurers are risk averse then the optimal level of sanctions tends to be lower than harm, because it reduces the imposition of risk on injurers and because sanctions do not need to be as high to induce injurers to behave appropriately. The second assumption is that of *strict liability*, which stipulates that injurers are definitely found liable and that injurers cannot escape the corresponding sanctions.

[1]Here, harm is the violation of a particular social commitment and is at least equal to the effort that is needed to fulfill the commitment.

3. CONCLUSION

In this paper, we have raised the problem of the enforcement of flexible social commitment in open systems which has been neglected in the past. We have introduced and discussed tools for treating this problem (Section 2), namely sanctions (Section 2.1) and punishment philosophies (Section 2.2). In doing so, we have shown that supporting the enforcement of social commitments may be made through a priori defined explicit sanctions (statically specified or dynamically negotiated) under a retribution punishment philosophy. In order to gain flexibility, the regimentation usually associated with social commitments has been moved to the sanction system itself. Indeed sanctions must be respected to ensure that this solution does not lead to the meta-problem of the enforcement of sanctions.

Notice that the design of (domain dependent) punishment policies, linking objective (or subjective) actions values to material sanctions, is an open research issue that we wish to inquire in future work.

4. REFERENCES

[1] J. Bates. The role of emotion in believable agents. *Communications of the ACM*, 37(7):122–125, July 1994.

[2] C. Beccaria. *On Crimes and Punishments*. New Jersey: Prentice Hall, 1963.

[3] J. Bentham. *An Introduction to the Principles of Morals and Legislation*. London: The Athlone Press, 1970.

[4] M. Hechter and K. D. Opp. Introduction. In M. Hechter and K. D. Opp, editors, *Social Norms*, pages xi–xx. Russell Sage Foundation, 2001.

[5] C. Hewitt. Open information systems semantics for distributed artificial intelligence. *Artificial Intelligence*, 47:76–106, 1991.

[6] D. J. Levinson. Collective sanctions. Public law research paper no. 57, New York University, School of Law, Center for Law and Business Research, 2003.

[7] D. Martindale. *Social Control for the 1980s: A Handbook for Order in a Democratic Society*, chapter The theory of social control, pages 46–58. Westport, CT: Greenwood Press, 1978.

[8] M.A. Polinsky and S. Shavel. *The New Palgrave Dictionary of Economics and The Law*, volume 3, chapter Punitive Damages, pages 192–198. London: Macmillan Reference Limited, 1998.

[9] R. A. Posner and E. B. Rasmusen. Creating and enforcing norms, with special reference to sanctions. *International Review of Law and Economics*, 19(3):369–382, 1999.

[10] G. B. Vold, T. J. Bernard, and J. B. Snipes. *Theoretical Criminology*. Oxford University Press, 5th edition, 2002.

Modelling the Links Between Social Commitments and Individual Intentions

Philippe Pasquier
DAMAS Laboratory
Laval University
G1K 7P4, Québec, Canada
pasquier@iad.ift.ulaval.ca

Brahim Chaib-draa
DAMAS Laboratory
Laval University
G1K 7P4, Québec, Canada
chaib@iad.ift.ulaval.ca

ABSTRACT

Social commitments have been increasingly used to model inter-agent dependencies and normative aspects of multi-agent systems such as the semantics of agent communication. However, current cognitive agent architecture rest on a formalization of private mental states. In this paper, we propose a modelling of the links between private mental states resulting in individual intentions and social commitments.

1. INTRODUCTION AND MOTIVATIONS

On one hand, social commitment has been introduced as a first class concept to represent socially established (and grounded) interagent dependencies and normative aspects such as roles, authority relations, social norms and the semantics of agent communication [13]. On the other hand, main current cognitive agent architectures rest on a formalization of private mental states [18]. In this paper, we propose a modelling of the links between private mental states (resulting in individual intentions) and social commitments when used in the context of agent communication.

2. SOCIAL COMMITMENTS

Social commitments are useful to model the semantics of agents' interactions. In that context, being able to cancel or modify commitments is a key feature that allows agents to reassess the consequences of past dialogues in the context of dynamic environments. This *semantical flexibility* should not be confused with the commonly considered structural flexibility of dialogues. Since [9] discusses our modelling of flexible social commitments and their enforcement through sanctions, we simply re-introduce the basic of it here.

The notion of commitment is a social one, and should not be confused with the notion of individual commitment used to emphasize individual intention persistance. Conceptually, commitments are oriented responsibilities contracted towards a partner or a group. Following [17], we distinguish *action commitments* from *propositional commitments*.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Commitments are expressed as predicates with an arity of 6. Thus, an *accepted* action commitment takes the form:

$$C(x, y, \alpha, t, s_x, s_y)$$

meaning that agent x is committed towards agent y to α since time t, under the sanctions sets s_x and s_y. An accepted propositional commitment would have propositional content p instead α. *Rejected* commitments, meaning that x is not committed toward y to α, takes the form $\neg C(x, y, \alpha, t, s_x, s_y)$. This notation allows to compose the actions or propositions involved in the commitments: $\alpha_1|\alpha_2$ classically stands for the choice, and $\alpha_1 \Rightarrow \alpha_2$ for the conditional statement that α_2 will occur in case of the occurrence of the event α_1. Finally, agents keep track of each commitment in which they are debtor or creditor in their *agendas*, which constitutes a kind of distributed "Commitment Store".

In previous work, we have proposed a dialogue games based agent communication language (DIAGAL [7]) which is sound an complete according to the creation, cancellation, modification and discharge of social commitments. In that context, social commitments allow to get ride of the main problems associated with the mentalistic semantics previously proposed. Indeed, according to the principle of *information asymmetry*, what is said does not convey anything about what is actually believed. However, what is said socially commits the locutors toward one another. Social commitments raise action expectations and the enforcement of social commitments through various social control mechanisms take place instead of the sincerity and the cooperativeness assumptions. Social commitments, when modelled with their enforcement mechanism (as in [9]), are not necessarily sincere and don't require the agents to be cooperative. From this perspective, communication serves to coordinate the agents whether or not they are cooperative and whether or not they are sincere.

These social commitment based frameworks, enhancing the social aspects of agents' communications, entail a change of paradigm: agents do not necessarily have to reason on others' intentions anymore but rather they must reason on taken and to be taken social commitments. However, social commitments were not taken into account in previous cognitive agents theory. As a first step to fill this gap, we propose to model the links between private cognitions and social commitments.

3. INDIVIDUAL INTENTIONS

According to the classic practical reasoning scheme, private cognitions end up in intentions through deliberation and we make the usual distinction between *intention to* (do something or that someone do something) and *intention that* (a proposition holds) [2]. The *intention to* relates to a particular course of action (eventually of a complex and structured type), while *intention that* refers to a propositional statement that the agent wants to became true. Intentions are either accepted ($I_A(p)$) or rejected ($\neg I_A(p)$).

In order to address the social dimension of communication, we will further distinguish between *internal individual intentions* and *social individual intentions*[1]. Internal individual intentions are intentions that the agent can try to achieve alone while social individual intentions are the intentions that relate to other agents' actions. Social individual intentions are intentions concerning goals which require other agents to be worked on. More generally, any intention that is embedded in a somewhat collective activity would be considered as a social individual intention except if it is part of an already socially accepted collective plan. Those social intentions are intentions about a (even indirectly) collective state of affairs indicating that those intentions will be part of a social activity (a problem requiring action, permission or opinion of the others: commerce, exchange, joint action, delegated actions,...). A classic example is delegation where an agent A has the social intention that an agent B achieves a particular action α, $I_A(\alpha_B)$.

Among internal individual intentions, we will also consider *failed individual intentions* which are the intentions that the agent failed to find an individual plan for or for which the available plans failed. This last type matches the case where the agent faces an individual problem he cannot solve alone or he failed to solve alone.

In our approach, failed individual intentions as well as the social individual intentions will be treated through dialogue. The phase of identifying intentions involving a social dimension appears to be crucial for integrating social commitment based approaches with existing cognitive agent architectures. In our approach, all intentions that are not achievable internal intentions will be selected as such. Filtering those failed and social intentions from the other ones is achieved by selecting the intentions for which the mean-end reasoning failed. For example, in the JACK-BDI framework [6], intentions that don't match any individual plans or for which all available individual plans have failed fall into those categories. Notice that this selection process implies that trying to achieve individual action (through execution of individual plans) is the prioritized behavior of the agent. Figure 1 sums up this intention typology.

4. LINKING INTENTIONS AND SOCIAL COMMITMENTS

In this context, we can return to the general question: what are the links between social commitments and private mental states? As a first answer, we propose linking private and public cognitions as follows. Ideally, an accepted social commitment is the socially accepted counterpart of an

[1]With the "individual" qualifier in both, we mean that we do not refer to notions of we-intention or collective intentions such as those developed by [11] or [15]. Here, intentions are classical private intentions.

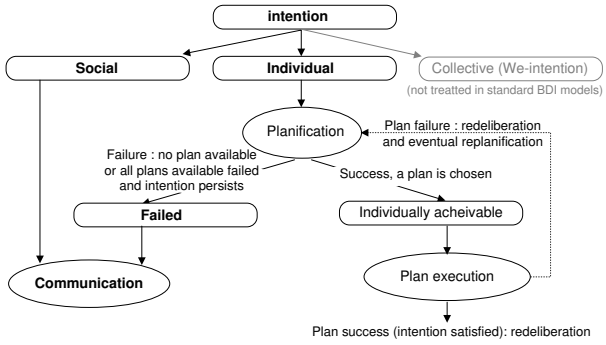


Figure 1: Operational typology of intentions.

accepted intention. Commitments in action are the counterparts of "intentions to" while propositional commitments are the counterparts of "intentions that". In our approach, those links are taken into account by positive and negative binary constraints that link the agents intentions and social commitments. Positive constraints take into account the correspondence relation introduced above while negative constraints model the incompatibility relations that hold between incompatible intentions or/and social commitments.

Let's take an example to illustrate those relations. If an agent A has the accepted individual social "intention to" that another agent B achieves an action α (noted $I_A(\alpha_B)$), our links mean that the corresponding social commitment from B toward A to achieve α_B (noted $C(B, A, \alpha_B, t, s_B, s_A)$ must be socially accepted as part of this intention satisfaction. This ideal link between those two cognitions is captured with a positive constraint. For this constraint to be satisfied, both elements (the intention and the corresponding commitment) must be accepted or rejected. However, all other possibilities are also important to consider. Furthermore, incompatibility relations are modelled with negative constraints.

5. RELATED WORK

Those relations between the private and public cognitions are not completely new since many authors have already considered individual intentions as a special kind of individual commitment [2, 16]. Our links extend this to reach the social level in the appropriate cases by saying that social individual intentions or failed individual intentions should ideally lead to the social acceptance of their social commitments counterparts through dialogue. Those links complement Singh's previous work [12], which introduces the idea of linking individual and social commitments. Comparable links have been introduced for so-called normative-deliberative cognitive agent architecture [5, 1, 3]. In particular, following [4], the following axioms have been introduced [10]:

$$S\text{-}COMM(i, j, \tau_i) \rightarrow I_j(\tau_i),^2 \text{ and}$$
$$S\text{-}COMM(i, j, \tau_i) \rightarrow I_i(\tau_i)$$

From which, one can deduce the following theorem:
$\vdash \neg S\text{-}COMM(i, j, \tau) \vee (I_i(\tau) \wedge I_j(\tau))$, which clearly states

[2] Sometimes formulated : $S\text{-}COMM(i, j, \tau_i) \rightarrow Goal_j(Does_i(\tau))$.

that either the social commitment is rejected or both i and j have the intention that i achieves the action τ. This formalization is not compatible with the semantic flexibility of social commitments described in Section 2. For example, if i decides to violate or cancel the aforementioned commitment, it is probably because he does not have the corresponding intention accepted. In that case, we have the accepted commitment $S\text{-}COMM(i, j, \tau_i)$ and the rejected intention $\neg I_i(\tau_i)$ that holds which invalidates the second of the above axioms. Symmetrically, if the agent j tries to cancel the accepted commitment $S\text{-}COMM(i, j, \tau_i)$, it can be because he does not have the corresponding intention accepted. In that case, we have $S\text{-}COMM(i, j, \tau_i)$ and $\neg I_i(\tau_i)$ which invalidates the first of the above axioms. In other words, those axioms are not flexible enough to provide a good modelling of the links that lie between intentions and flexible social commitments.

Constraints provide bidirectional and symmetric links that go behind the above mentioned axioms.[3] This is why we used constraints in order to model those links. Examples where a commitment is accepted and the corresponding intention is not or the reverse are very common and just mean that the positive constraint linking those two elements is not satisfied. As a consequence, not only those bidirectional links are more correct than the previously criticized axioms but they allow for a new question to be asked. When such a constraint is not satisfied, the agent has to decide which elements' acceptance state he will try to change in order to satisfy this positive constraint: his intention or the corresponding social commitment. This is the basic question of the attitude change process that we modelled in previous works [8].

6. CONCLUSION

While social commitments capture the social interdependencies of the agent, individual intentions make the links between the agent private cognition and its individual or social behavior. This short paper advances the state of the art by motivating and proposing a modelling of the links between individual intentions and social commitments. While this idea is not new, it is shown how the proposed links improve previous proposals.

7. REFERENCES

[1] G. Boella and L. Lesmo. *Social Order in Multi-Agent Systems*, chapter Deliberative normative agents, pages 85–110. Kluwer Academic, 2001.

[2] M. E. Bratman. What is intention? In P. R. Cohen, J. L. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 15–32. The MIT Press: Cambridge, MA, USA, 1990.

[3] J. Broersen, M. Dastani, J. Hulstijn, Z. Huang, and L. Van der Torre. The BOID architecture: Conflicts between beliefs, obligations, intention and desires. In *Proceedings of the Fifth International Conference on Autonomous Agent*, pages 9–16. ACM Press, 2001.

[4] C. Castelfranchi. Commitments: from individual intentions to groups and organizations. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 41–48, San Francisco, CA, USA, 1995.

[5] C. Castelfranchi, F. Dignum, C. Jonker, and J. Treur. Deliberative normative agents: Principles and architecture. In *Intelligent Agents V, Proceedings of the Internationnal Workshop on Agent Theories, Architectures, and Languages (ATAL)*, volume 1757 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 364–378. Springer-Verlag, 1999.

[6] N. Howden, R. Ronnquist, A. Hodgson, and A. Lucas. Jack intelligent agents: summary of an agent infrastructure. In *Proceedings of the Fifth International Conference on Autonomous Agents*. ACM Press, 2001.

[7] P. Pasquier, M. Bergeron, and B. Chaib-draa. DIAGAL: a Generic ACL for Open Systems. In *Proceedings of The Fifth International Workshop Engineering Societies in the Agents World (ESAW'04)*, volume 3451 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 139–152. Springer-Verlag, 2004.

[8] P. Pasquier and B. Chaib-draa. Agent communication pragmatics: The cognitive coherence approach. *Cognitive Systems*, 6(4):364–395, December 2005.

[9] P. Pasquier, R. A. Flores, and B Chaib-draa. Modelling flexible social commitments and their enforcement. In *Proceedings of the Fifth International Workshop Engineering Societies in the Agents World (ESAW'04)*, volume 3451 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 153–165. Springer-Verlag, 2004.

[10] L. Royakkers and F. Dignum. No organisation without obligation: How to formalise collective obligation? In M. Ibrahim, J. Kung, and N. Revell, editors, *Proceedings of the* 11th *Internationnal Conference on Databases and Expert Systems Applications*, volume 1873 of *Lecture Notes in Computer Science (LNCS)*, pages 302–311. Springer-Verlag, 2000.

[11] J. R. Searle. Collective intentions and actions. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 401–416. The MIT Press: Cambridge, MA, USA, 1990.

[12] M. P. Singh. Social and psychological commitments in multiagent systems. In *AAAI Fall Symposium on Knowledge and Action at Social and Organizational Levels*, pages 104–106, 1991.

[13] M. P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 7:97–113, 1999.

[14] P. Thagard. *Coherence in Thought and Action*. The MIT Press: Cambridge, MA, USA, 2000.

[15] R. Tuomela and K. Miller. We-intentions. *Philosophical Studies*, 53:367–389, 1988.

[16] G.H. von Wright. *Freedom and determination*. North Holland Publishing Co., 1980.

[17] D. N. Walton and E. Krabbe. *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. Suny Press, 1995.

[18] M. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, 2001.

[3] We refer the interested reader to [14] for a discussion about bidirectionality in cognitive modelling.

The MAP^a Language of Agent Dialogues

María Adela Grando [*] [†]
Research Group on Mathematical Linguistics,
Rovira i Virgili University,
Tarragona, Spain.

Christopher D. Walton [‡]
Centre for Intelligent Systems and their
Applications, School of Informatics,
University of Edinburgh, UK.

ABSTRACT

In this paper we present the MAP^a language for expressing coordination and communication between agents in multi-agent systems. This is accomplished by defining patterns of dialogues between groups of agents, expressed as protocols. Our protocols are flexible and directly executable. Furthermore, our language allow us to specify the connection between communication and knowledge management in a way that is independent of the specific reasoning techniques used.

1. INTRODUCTION

Multi-Agent Systems (MAS) are a popular technique in Artificial Intelligence for effecting reasoning in a distributed manner. These systems are defined from collections of individual agents which are composed together into systems within an environment. Each individual agent is an autonomous and rational entity, and exhibits intelligent behaviour based on interactions with other agents in the environment. It is this intelligent behaviour that distinguishes a MAS from a conventional distributed software system.

An agent in a MAS is intended to perform in a similar way to a human agent. That is, the agent should be capable of acting on behalf of others, either humans or agents. To accomplish this, it is necessary for the agent to be able to *communicate* and *coordinate* with other agents to achieve certain tasks. This inter-operability between agents is surprisingly difficult to achieve. There are a wide range of issues that have to be addressed, including at least the following: The agent must divide the problem into tasks for which external cooperation is required, and find suitable agents to accomplish these tasks. The tasks must be communicated to the external agents, who must in-turn agree to participate. The agent should interact with these external agents to ensure the task is accomplished, and finally interpret the outcome to obtain the results. These issues may be further complicated as the external agents may use different formalisms or have a different world-view from the initiating agent. Furthermore, it is generally not possible to compel another agent to perform any particular task. Instead, it is typically necessary for agents to negotiate to reach a consensus on a given task. The provision of a solution to many of these issues is the focus of this paper.

To address the issues of communication and coordination that we have highlighted, it is useful to take our inspiration from human *social systems*, where these issues are solved in the real world. Following this approach, it is common to treat a group of agents as a *society* [3]. The agents in the society will typically share a common interest, e.g. solving a particular problem. To become a member of this society, an agent must agree to observe certain rules and conventions. In return, the agent can itself benefit from the society, e.g. the expertise of other agents. In this societal view, it becomes possible to define conventions for the agents to follow, and the incentives for agents to operate together are made clear. This in-turn makes the issues of communication and coordination more manageable.

In any society, the rules and conventions are expressed as *social norms*. These norms may often be implicit, but are necessary nonetheless. For example, at a conference talk there is an implicit norm that the speaker does the talking while the audience listens. If everyone were to talk at the same time then the conference would break down. Thus, it is these social norms that ensure the smooth running of the society. Returning to MAS, we can readily define social norms for a group of agents to follow. These rules should be defined in such a way to facilitate the successful interaction between agents. However, the rules should not be overly restrictive in that they affect the autonomy of the agents.

A popular way to express the social norms between groups of agents is by means of an explicit *protocol*. A protocol is a statement of the social norms. The protocol can be examined in-advance by an agent to inform their decision on whether to join a particular society. The protocol also acts as a guide for the agents to follow once they are operating within the society. This approach has been successfully adopted in the Conversation Policy (CP) [2] and Electronic Institutions (EI) [1, 4] formalisms, among others.

[*]The primary author is a Ph.D. student.

[†]Sponsored by research grant "Programa Nacional para la Formación del Profesorado Universitario" from the Ministry of Education, Culture and Sports of Spain.

[‡]Sponsored by the EPSRC Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (Grant GR/N15764/01).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

In this paper, we define a language which can express protocols. Our protocols define patterns of *dialogue* between groups of agents. If a group of agents follow a particular dialogue then they should reach a particular well-defined outcome. Nonetheless, the protocols contain decision points in which the agents can behave autonomously. The protocols that we define are *executable specifications*. That is, the formalisms that we use to specify our protocols are exactly the same as those used for enactment. This is in contrast to the more traditional means of MAS construction, where an initial specification is refined into an executable model. For example, a MAS specified in a Belief-Desire-Intention (BDI) logic, and implemented in a FIPA-compliant architecture.

Our language defines Multi-Agent Protocols, and is called MAP^a. This language is an extension of the MAP language which we previously presented in [6]. The extensions in this paper make the protocols more flexible and dynamic, but retain the lightweight nature of the previous definitions. In MAP^a, we treat protocols as first-class objects which can be passed between agents as opposed to our previous static definitions. We also permit the organisation of roles in hierarchies, structuring role definitions in a modular and reusable way. Finally, we permit the management of knowledge to be expressed in our protocols, which enables a better connection between the protocol and the agent reasoning processes at the decision points.

2. MAP^A LANGUAGE DEFINITION

MAP^a is a lightweight language-based formalism for the expression of protocols, which express social interactions between groups of agents. The key concepts in MAP^a are *scenes*, and *roles* which we now explain. A scene can be thought of as a bounded space in which a group of agents interact on a single task. We assume that a scene places barrier conditions on the agents, such that a scene cannot begin until all the agents are present, and the agents cannot leave the scene until the dialogue is complete.

The concept of an agent *role* is also central to the definition of our protocols. Agents entering a scene assume a role which they can change during the scene. For example, a hospital scene may involve agents with the roles of *doctor* and *patient*. Roles are defined as a hierarchy, where more specialised roles appear further down in the graph of roles. For example, an agent may initially assume the role *doctor* but may change to the more specialised role *surgeon* during a scene.

A protocol is defined as a sequence of definitions parameterised on the role. Thus, the protocol steps which an agent follows in a scene are directly dependent on the role, or subrole, which they assume. The role also identifies capabilities which the agent must provide. It is important to note that a protocol only contains operations which are specific to the mechanisms of communication and coordination between agents. This makes it straightforward to understand the operation of the protocol without extraneous details, and makes it possible to verify the protocols using automated means, e.g. model checking [5]. All of the other agent facilities, e.g. the reasoning processes, are encapsulated by *decision procedures* which are external to the protocol. In effect, the decision procedures provide an interface between the communicative and the rational process of the agent. We allow a decision procedures to be private to a particular agent, or shared between agents playing the same role.

Interaction between the agents in a scene is performed by the exchange of messages. Messages can be sent or received and this exchange happens in a blocking manner. Every message has an associated identifier which is used to indicate the type of the message and parameters. For convenience, we do not assign any fixed semantics to these messages. However, individual agents can agree on a semantics for a particular scene. In this way, we can readily represent FIPA-style agent communication, e.g. the contract-net protocol.

The final concept in MAP^a is the representation of knowledge management during agent interaction. The language allows us to define *knowledge*, represented as axioms, at a scene level, a role level, or the level of an individual agent. In this way, we can clearly express the exchange of knowledge between one agent and another, and the transition from private to public knowledge of certain facts during protocol execution.

S	::=	$\langle R^{(i)},\ P^{(i)},\ M^{(k)},\ K^{(b)} \rangle$	(Scene)
R	::=	$\langle id,\ Proc^{(l)},\ K^{(m)},\ r^{(n)} \rangle$	(Role)
P	::=	$\mathbf{agent}(id, r, Proc^{(l)}, K^{(m)}, \phi^{(f)}) = op.$	
K	::=	$axiom$	(Knowledge)
$Proc$	::=	$type :: id((\phi, type)^{(g)})$	(Procedure)
M	::=	$id((\phi, type)^{(h)})$	(Performative)
op	::=	v	(Variable)
	|	α	(Action)
	|	op_1 **then** op_2	(Sequence)
	|	op_1 **or** op_2	(Choice)
	|	op_1 **par** op_2	(Parallel)
	|	(op)	(Precedence)
α	::=	**null**	(No Action)
	|	$v = p(\phi^{(g)})$	(Decision)
	|	$id(\phi^{(x)}) \Longleftarrow \mathbf{agent}(id,\ r)$	(Receive)
	|	$id(\phi^{(y)}) \Longrightarrow \mathbf{agent}(id,\ r)$	(Send)
	|	$\mathbf{agent}(id,\ r,\ Proc^{(w)},\ K^{(v)},\ \phi^{(d)})$	(Invocation)
ϕ	::=	$c \mid \_ \mid v$	(Term)

Figure 1: MAP^a Language Syntax.

A BNF-style abstract syntax for MAP^a is shown in Figure 1. Superscripts are used to indicate a set, e.g $P^{(i)}$ is a set with elements P of size i. To illustrate the MAP^a language syntax we now define in Figure 2 a general negotiation protocol. We have used the following abbreviations: IS is an Information Seeker role, IP is an Information Provider role, t and newt are a negotiation topic and new negotiation topic, f and newf are a fact and new fact, a is an answer, and prot is a protocol variable.

There are four parts to the negotiation protocol, parameterised on the roles of Patient, GPDoctor, IS, and IP. The core of the dialogue is between an information seeker (IS) and an information provider (IP). The roles Patient and GPDoctor begin the dialogue. The Patient starts searching for a suitable information provider for a given topic and fact (t, f) and will then initiate a dialogue with this provider. $K^{(b)}$ and $Proc^{(a)}$ stand for Patient knowledge and Patient procedures. $K^{(b)}$ can be defined as the basic knowledge that patients have about illnesses, and $Proc^{(a)}$ can be the set of possible actions to perform in presence of some illness. GPDoctors adopt the role of providers with advanced knowledge of medicine, $K^{(d)}$, and treatments for the known diseases, $Proc^{(c)}$.

The information provider (IP) is defined in four cases separated by the choice operator. The evaluation model of the protocol is such that the protocol will backtrack and retry

agent(id$_1$, Patient, Proc$^{(a)}$, K$^{(b)}$, t, f) =
id$_2$=search_doctor(t, f) then (agent(id$_1$, IS,
Proc$^{(a)}$,K$^{(b)}$∪{answer(t,f)=null}, t, f, id$_2$)).

agent(id$_1$,GPDoctor, Proc$^{(c)}$, K$^{(d)}$)=
agent(id$_1$, IP, Proc$^{(c)}$, K$^{(d)}$).

agent(id$_1$, IP, Proc$^{(t)}$, K$^{(z)}$) =
(request(t, f) ⟸ agent(id$_2$, IS)
then agent(id$_1$, IP, ∅, ∅)) or
(a=get_answer(t, f) then
inform(t, f, a) ⟹ agent(id$_2$, IS)) or
(prot = get_protocol(t, f) then
what_to_do(t, f, prot)⟹agent(id$_2$, IS)) or
((newt, newf) = next_to_know(t, f) then
agent(id$_1$, IS, ∅,
{answer(t,f)=null,answer(newt,newf)=null},
newt, newf, id$_2$) then agent(id$_1$, IP, ∅, ∅, id$_2$)).

agent(id$_1$, IS, Proc$^{(m)}$, K$^{(s)}$, t, f, id$_2$) =
(request(t, f) ⟹ agent(id$_2$, IP) then
agent(id$_1$, IS, ∅, ∅, t, f, id$_2$)) or
(inform(t, f, a) ⟸ agent(id$_2$, IP) then
agent(id$_1$, IS, ∅, {answer(t,f)=a} , t, f, id$_2$) or
result=Newknowledge({answer(t,f)=a})) or
(what_to_do(t, f, prot) ⟸ agent(id$_2$, IP) then
prot
then agent(id$_1$, IS, ∅, {answer(t,f)=prot} ,t,f,id$_2$)
or result=Newknowledge({answer(t,f)=prot}))
or (request(newt, newf) ⟸ agent(id$_2$, IS)
then agent(id$_1$, IP, ∅, ∅)
then agent(id$_1$, IS, ∅, ∅, t, f, id$_2$)).

Figure 2: MAP^a Negotiation Protocol.

each of these cases until one of them can be satisfied. In the first case, the information provider receives a request for information (on some topic and fact) from an information seeker. The information provider will then update its state and attempt to answer the request. The appropriate answer is determined in the remaining cases. The second case is triggered when the information provider has a direct answer for the request, in which case this answer is returned to the seeker. In the third case, it is necessary for the seeker to perform additional actions before the request can be satisfied, so the provider sends a protocol to the seeker containing the appropriate actions. In the final case, more information is required, so the provider becomes a seeker for this new information. What he learned as IP is taken as private knowledge for the new role of IS (answer(t,f)=null, answer(newt,newf)=null). Upon obtaining the extra information, the agent reverts back from a seeker to a provider again.

The information seeker (IS) protocol is essentially the complement of the information provider protocol, and there are four corresponding cases. In the first case, the seeker sends a request for information to the provider. The remaining cases deal with the responses from the provider. In the second case, the requested information is received from the provider and the protocol either restarts with the new private knowledge (answer(t,f)=a) or terminates. In the third case, a protocol is received from the provider which is evaluated. Lastly, in the fourth case, a request for more information is received, and the agent assumes the role of a provider until this request is satisfied.

3. CONCLUSIONS

In this paper we have presented an overview of our MAP^a language. The purpose of this language is the specification of the communication and coordination processes between groups of agents in a multi-agent system. This is accomplished by the definition of protocols which express social behaviours between the agents. In particular, these protocols define social norms which agents must observe if they are to participate in, and benefit from, the society. The use of protocols ensures that the interactions between agents in the society happen in a controlled and predictable manner. Nonetheless, these protocols do not overly restrict the individual autonomy of the agents.

Our approach differs from previous work on multi-agent protocols, in a number of important ways. Our protocols are executable specifications, and we have constructed a meta-interpreter for our protocols in Java. This direct execution eliminates a potentially error-prone refinement task between the specification and implementation. Our protocols are defined separately from the reasoning processes of the agents. This enables our protocols to be independently examined and verified before an agent subscribes to the society. Furthermore, it permits a range of reasoning strategies to be adopted, e.g. reactive or planning systems, rather than restricting us to a single model.

With regard to future work, we plan to use MAP^a in conjunction with Web Services, to perform service composition on the Semantic Web. We also plan to use the language with eco-grammars to perform automatic agent-based grammar generation. Our future plans for extensions to the language include facilities for constraint satisfaction and the inclusion of deontic relationships between agents.

4. REFERENCES

[1] M. Esteva, J. A. Rodríguez, C. Sierra, P. Garcia, and J. L. Arcos. On the Formal Specification of Electronic Institutions. In *Agent-mediated Electronic Commerce (The European AgentLink Perspective)*, number 1991 in Lecture Notes in Artificial Intelligence, pages 126–147, 2001.

[2] M. Greaves, H. Holmback, and J. Bradshaw. What is a Conversation Policy? In *Proceedings of the Workshop on Specifying and Implementing Conversation Policies, Autonomous Agents '99*, Seattle, Washington, May 1999.

[3] Y. Shoham and M. Tennenholtz. On the Synthesis of Useful Social Laws for Artificial Agent Societies. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, San Diego, USA, 1992.

[4] W. Vaconcelos. Norm Verification and Analysis of Electronic Institutions. In *2004 Workshop on Declarative Agent Languages and Technologies (DALT-04)*, pages 141–155, New York, USA, July 2004.

[5] C. Walton. Model Checking Multi-Agent Web Services. In *Proceedings of the 2004 AAAI Spring Symposium on Semantic Web Services*, Stanford, California, March 2004. AAAI.

[6] C. Walton. Multi-Agent Dialogue Protocols. In *Proceedings of the Eighth International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, Florida, January 2004.

Convergence Analysis for Collective Vocabulary Development *

Jun Wang
Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign
junwang4@uiuc.edu

Les Gasser
Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign
gasser@uiuc.edu

Jim Houk
Department of Physiology
Northwestern University Medical School
Chicago, IL
j-houk@northwestern.edu

ABSTRACT

We study how decentralized agents can develop shared vocabularies without global coordination. Answering this question can help us understand the emergence of many communication systems, from bacterial communication to human languages, as well as helping to design algorithms for supporting self-organizing information systems such as social tagging or ad-word systems for the web. We introduce a formal communication model in which senders and receivers can adapt their communicative behaviors through a type of *win-stay lose-shift* adaptation strategy. We find by simulations and analysis that for a given number of meanings, there exists a threshold for the number of words below which the agents can't converge to a shared vocabulary. Our finding implies that for a communication system to emerge, agents must have the capability of inventing a minimum number of words or sentences. This result also rationalizes the necessity for syntax, as a tool for generating unlimited sentences.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems, Languages and structures, Coherence and coordination*

Keywords

self-organizing vocabularies, win-stay lose-shift

1. INTRODUCTION

We study the question of how decentralized agents can develop a shared vocabulary without global coordination. This is a fundamental question that underpins how we understand the emergence of many communication systems arising in nature and in human society, e.g., how do different organisms develop the communication needed to coordinate their activities, how do encoding/decoding networks arise in the brain, or how did human languages emerge? The study of this question can also help us design algorithms that support self-organizing information systems such as social tagging, web ad-words, or peer-to-peer retrieval systems.

Two general frameworks, *evolutionary* and *self-organizing*, have been proposed to study the emergence of communication systems. Evolutionary frameworks assume that vocabularies of agents are inherited genetically and/or learned culturally from parents or teachers. The driving force for the emergence of shared vocabularies comes from mechanisms in which good communicators produce more offspring with similar vocabularies[2, 6, 7]. In contrast, in self-organizing frameworks decentralized agents actively develop shared vocabularies without global knowledge in a short span of time by changing their own representations via distributed learning and positive feedback loops[1, 3, 4, 8]. The work reported here can be seen as an effort in the self-organizing camp.

Most existing work on the development of shared vocabularies has been done using computer simulations and only intuitive interpretations. While this is often suggestive, to achieve a thorough understanding we need to augment simulation studies with mathematical characterizations of specific conditions and limitations under which the emergent behaviors are stable. Recent work by Baronchelli et al.[1] provides one example; they explain how a sharp transition to a shared vocabulary observed in simulations can occur as the density of certain inter-agent vocabulary hypotheses grows beyond a "tipping point."

In this paper, we focus on a formal communication model in which senders and receivers can adapt their communicative behaviors through a type of win-stay lose-shift learning strategy[5]. To simplify the characterization of the dynamics, we study a model that consists of two agents. Through simulations and analysis, we give the conditions under which the agents can develop a stable shared vocabulary.

2. WIN-STAY LOSE-SHIFT ADAPTATION MODEL

Our communication model consists of two agents, called a *sender* and a *receiver*. In our model, there are m meanings $\{x_1, \cdots, x_m\}$, and n words $\{y_1, \cdots, y_n\}$. The sender and

*Work supported by NSF Grant IIS-0340996 and UIUC Beckman Summer Fellowship.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

The sender's dictionary **The receiver's dictionary**

Round 0

meaning	word	wins
x_1		0
x_2		0
x_3		0

word	meaning	wins
y_1		0
y_2		0
y_3		0

Round 1 (play game on meaning x_1, successful—a luck.)

meaning	word	wins
x_1	y_1	1
x_2		0
x_3		0

word	meaning	wins
y_1	x_1	1
y_2		0
y_3		0

Round 2 (play game on meaning x_1, successful.)

meaning	word	wins
x_1	y_1	2
x_2		0
x_3		0

word	meaning	wins
y_1	x_1	2
y_2		0
y_3		0

Round 3 (play game on meaning x_2 via word y_1, failed.)

meaning	word	wins
x_1	y_1	2
x_2		0
x_3		0

word	meaning	wins
y_1	x_1	1
y_2		0
y_3		0

Round 4 (play game on meaning x_3 via word y_1, failed.)

meaning	word	wins
x_1	y_1	2
x_2		0
x_3		0

word	meaning	wins
y_1		0
y_2		0
y_3		0

Figure 1: An illustration of repeated play of communication games.

receiver each have a dictionary, consisting of a list of entries. The sender's entry is of the format $<meaning, word>$ or $<x, y>$. Each entry is indexed by a meaning; it indicates that for each meaning there is one and only one entry. For each entry $<x, y>$, there is an associated number that indicates how many more times the sender has successfully communicated the meaning x via the word y than he has failed to do. Such a number is called the *wins* of the entry, and denoted by $wins_s(x)$, where the subscript $_s$ indicates the sender. We require $wins \geq 0$. When there is no win (i.e., $wins = 0$), the word y will be empty. Initially, all entries have no win.

Similarly, the format $<word, meaning>$ or $<y, x>$represents the receiver's entry. Each entry is indexed by a word; it indicates that for each word there is one and only one entry. For each entry $<y, x>$, there is an associated number that indicates how many more times the receiver has successfully interpreted the word y to the meaning x than he has failed to do. Such a number is called the *wins* of the entry, and denoted by $wins_r(y)$, where the subscript $_r$ indicates the receiver. When there is no win (i.e., $wins = 0$), the meaning x will be empty. Initially, all entries have no win.

A communication event is initiated by the sender who tries to communicate a meaning to the receiver. The meaning that the sender wants to communicate is drawn according to some probability distribution. (In this short paper, we will assume it is uniform distribution.) To communicate a meaning, say, x, the sender needs to represent it as a word. The sender looks up the meaning x in his dictionary. Let the found entry be $<x, y>$. If $wins_s(x) > 0$, then the sender will use word y to represent the meaning x. If $wins_s(x) = 0$, which means the word y is empty, then the sender will randomly choose a word to represent the meaning. (This is the idea of *win-stay lose-shift*.)

When receiving a word, say, y, the receiver looks up the word in her dictionary. Let the found entry be $<y, x'>$. If $wins_r(y) > 0$, then the receiver will use the meaning x' to interpret the word y. If $wins_r(y) = 0$, which means the meaning x' is empty, then the receiver will randomly choose a meaning as the interpretation to the word.

If the interpreted meaning x' is correct (we suppose *successful task performance* can provide feedback on communicative success), i.e., $x' = x$, we say the communication is successful; otherwise, the communication is failed. When the communication succeeds, the sender will update the *wins* of the entry indexed by x according to the reward rule: $wins_s(x) \leftarrow wins_s(x) + 1$. If the word field is empty he will replace it with the word y. Similarly, the receiver will update the *wins* of the entry indexed by y according to the rule: $wins_r(y) \leftarrow wins_r(y) + 1$. If the meaning field is empty she will replace it with the meaning x.

When the communication fails, the sender will update the *wins* of the entry indexed by x according to the rule: $wins_s(x) \leftarrow wins_s(x) - 1$ if $wins_s(x) > 0$. If $wins_s(x) = 0$ he will replace the word field with empty. Similarly, the receiver will update the *wins* of the entry indexed by y according to the rule $wins_r(y) \leftarrow wins_r(y) - 1$ if $wins_r(y) > 0$. If $wins_r(y) = 0$ she will replace the meaning field with empty.

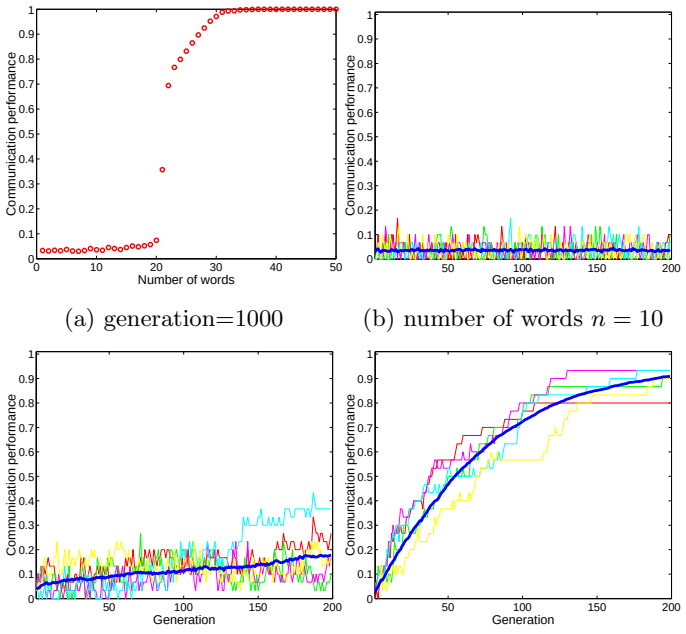
We call one communication event between the sender and receiver a *communication game*, and thus we model the vocabulary development process as repeated play of communication games (Fig. 1).

3. SIMULATIONS

In this section, we show by simulations that for a given number of meanings, there exists a threshold of the number of words below which the agents can't develop a shared vocabulary using the win-stay lose-shift adaptive communication model. In Fig. 2(a), we can see that there exists a dramatic *phase transition* when the number of words n is around a *threshold* $n^* = 21$. When n is above the threshold, the two agents can develop a communication system with converged communicative performance given by $\min\{\frac{n}{m}, 1\}$; that is, the sender and receiver can develop $\min\{n, m\}$ meaning-word agreements. However, when the number of words is below the threshold, it rapidly approaches random guess—the sender and receiver can't develop any effective communication system, or a shared vocabulary. The three graphs (c,d,e) in Fig. 2 illustrate the dynamics of communication over time for three specific settings of the number of words: $n = 10$, $n = 21$, and $n = 40$.

4. A SIMPLE ANALYSIS

In this section, we give a simple analysis aiming to understand why the above phase transition phenomenon happens. The basic idea is that soon or later the sender and receiver will form a temporary meaning-word agreement. Then, in the following rounds, this temporary agreement may be either reinforced or weakened. If the temporary agreement has a better chance to be reinforced than to be weakened, random walk theory tells us that the temporary agreement will have a positive probability of becoming a permanent agreement. We will show that the probability for a temporary agreement to become permanent depends on the relation between the number of words and the number of meanings. When the number of meanings is fixed, the more the words,



(a) generation=1000 (b) number of words $n = 10$

(c) number of words $n = 21$ (d) number of words $n = 40$

Figure 2: (a) Communication performance (success ratio) as a function of the number of words n when the number of meanings is held constant at $m = 30$. A dramatic phase transition occurs when the number of words n is around a threshold $n^* = 21$, above which the agents can develop a communication system with eventual communication performance given by $\min\{\frac{n}{m}, 1\}$. The performance numbers, marked by red circles, are obtained by averaging the results from 100 runs (after 1000 generations, where each generation contains $m = 30$ rounds of play). (b,c,d) Dynamics of adaptive communication over time (only 200 generations are shown). The thick lines in the graphs are obtained by averaging the results of 100 runs, 5 of which are shown as thin lines. All meanings are uniformly distributed.

the better the chance for a temporary agreement to become permanent. (See Fig. 1 for a better understanding of the following analysis.)

For simplicity, we will assume that all meanings are uniformly distributed. Since we have m meanings, so given a meaning the probability that it is chosen for play is $\frac{1}{m}$.

Without loss of generality, let a temporary agreement be on the meaning-word pair (x_1, y_1). With probability $\frac{1}{m}$, at the next round the sender will play game on the meaning x_1. Clearly the communication on meaning x_1 will succeed, so the *wins* value of the sender and receiver, $wins_s(x_1)$ and $wins_r(y_1)$, will increase by 1. Therefore we have a *reinforcement probability* of $\frac{1}{m}$ that the temporary agreement will be reinforced.

With probability $1 - \frac{1}{m}$, at the next round the sender will play game on a meaning other than x_1, say, x_2. If the *wins* value of x_2 in the sender's dictionary is zero, i.e., $wins_s(x_2) = 0$, then the sender will randomly choose a word from the n words to represent the meaning x_2, and so there is a probability of $\frac{1}{n}$ that word y_1 will be used to represent the meaning x_2. If this happens, the communication will fail, because the receiver will interpret word y_1 as meaning x_1. And then, the *wins* of the word y_1 in the receiver's dictionary, $wins_r(y_1)$, will decrease by 1. In summary, we have a *weakening probability* of $(1-\frac{1}{m})\frac{1}{n}$ that the temporary agreement will be weakened.

If the reinforcement probability, $\frac{1}{m}$, is larger than the weakening probability, $(1 - \frac{1}{m})\frac{1}{n}$, the temporary agreement will become stable. Therefore, by solving the following inequality

$$\frac{1}{m} > (1 - \frac{1}{m})\frac{1}{n}$$

we can see that when $n > m-1$, the sender and receiver can guarantee to establish a permanent agreement. Note that if in the above analysis we consider the case of $wins_s(x_2) > 0$, then the weakening probability will be even lower, and thus the condition $n > m-1$ can be relaxed. This partly explains why in the above simulations the threshold value is $n^* = 21$ rather than $n^* = m - 1 = 29$.

Clearly, the two agents can develop as many as $\min\{m, n\}$ stable agreements, since there are at most $\min\{m, n\}$ distinct pairs of meaning-word. And thus, the maximum converged communicative performance should be $\min\{m, n\}\frac{1}{m}$ or equivalently $\min\{\frac{n}{m}, 1\}$.

5. CONCLUSION

We have presented an adaptive communication model in which a sender and a receiver can adapt their communicative behaviors through a simple learning strategy—win-stay lose-shift. By computer simulations and mathematical analysis, we find that for a given number of meanings, there exists a threshold for the number of words below which the agents can't develop a shared vocabulary. Our finding shows that for a communication system to emerge, agents must be able to invent a minimum number of words or sentences. This result also rationalizes the necessity for syntax, as a tool for generating unlimited sentences.

Though this analysis treats just two communicating agents, we believe it can be extended to the case of multiple agents—one sender and many receivers, many senders and one receiver, or many senders and many receivers.

6. REFERENCES

[1] A. Baronchelli, M. Felici, E. Caglioti, V. Loreto, and L. Steels. Sharp transition towards shared vocabularies in multi-agent systems. http://arxiv.org/abs/physics/0509075, 2005.

[2] J. Hurford. Biological evolution of the saussurean sign as a component of the language acquisition device. *Lingua*, 77(2):187–222, 1989.

[3] F. Kaplan. Semiotic schemata: Selection units for linguistic cultural evolution. In M. Bedau and et al., editors, *Artificial Life VII*. The MIT Press, 2000.

[4] T. Lenaerts, B. Jansen, K. Tuyls, and B. de Vylder. The evolutionary language game: An orthogonal approach. *Journal of Theoretical Biology*, 235(4):566–582, August 2005.

[5] F. A. Matsen and M. A. Nowak. Win-stay, lose-shift in language learning from peers. *PNAS*, 101(52):18053–18057, December 2004.

[6] M. A. Nowak, J. B. Plotkin, and D. C. Krakauer. The evolutionary language game. *Journal of Theoretical Biology*, 200:147–162, 1999.

[7] K. Smith. The evolution of vocabulary. *Journal of Theoretical Biology*, 228(1):127–142, 2004.

[8] L. Steels. Self-organizing vocabularies. In C. Langton and T. Shimohara, editors, *Artificial Life V*, Nara, Japan, 1996.

Learning a Common Language Through an Emergent Interaction Topology

Samarth Swarup
Dept. of Computer Science,
University of Illinois at
Urbana-Champaign,
Urbana, IL, USA
swarup@uiuc.edu

Kiran Lakkaraju
Dept. of Computer Science,
University of Illinois at
Urbana-Champaign,
Urbana, IL, USA
klakkara@uiuc.edu

Les Gasser
Graduate School of Library
and Information Science, and
Dept. of Computer Science,
University of Illinois at
Urbana-Champaign,
Urbana, IL, USA
gasser@uiuc.edu

ABSTRACT

We study the effects of various *emergent* topologies of interaction on the rate of language convergence in a population of communicating agents. The agents generate, parse, and learn sentences from each other using recurrent neural networks. An agent chooses another agent to learn from, based on that agent's *fitness*. Fitness is defined to include a frequency-dependent term capturing the approximate number of interactions an agent has had with others—its "popularity" as a teacher. This method of frequency-dependent selection is based on our earlier *Noisy Preferential Attachment* algorithm, which has been shown to produce various network topologies, including scale-free and small-world networks. We show that convergence occurs much more quickly with this strategy than it does for uniformly random interactions. In addition, this strategy more closely represents choice preference dynamics in large natural populations, and so may be more realistic as a model for adaptive language.

Categories and Subject Descriptors

E.1 [**Data**]: Data Structures—*Graphs and networks*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms, Design, Experimentation, Languages

Keywords

Scale free networks, language evolution, preferential attachment

1. INTRODUCTION

One of the main problems in language evolution is to get a population of agents to converge to a common language, without globally imposing some kind of hierarchy on the population. The general problem can be stated as "Given a landscape of states and possible moves between them, and a search strategy under which agents learn from encounters with others, what constraints on the agent interaction topology will ensure the most rapid convergence to some highest-fitness state?" In effect, how should agents make choices of learning partners to achieve the fastest collective convergence to the best objective state? This is the problem we study, in the context of developing a common and high-quality language.

The "fitness" of a state in this formulation is defined by both an intrinsic measure (the objective quality of the state), and by a frequency-dependent measure (the number of agents that are in that state).

A simple strategy for rapid convergence would be to designate a special agent from which each agent learns its own language. This corresponds essentially to a pre-imposed or designed language, which may or may not be of the highest objective quality. Such a centralized system is brittle in practice because a) the teacher agent has to be responsible for adapting the language to keep up with changing tasks, environments and needs of all the agents, b) communicative load on the teacher increases at least linearly with population size, reducing scalability, and c) the centralized teacher is a single failure point. Multi-agent systems are generally distributed and open, which means that there is no central control point, and agents may enter and leave the population at any time. This means that although desirable for its speed, uniformity, and certainty, a pre-imposed interaction topology with a centralized teacher is both undesirable and unrealistic for a general multi-agent system.

2. RELATED WORK

Although there has been much work on how the topology of interaction graphs affect the convergence of a population of agents, none has focused on emergent topologies.

Delgado[2], extending the model of Shoham and Tennenholtz [5], examined the emergence of social conventions in a population of agents. His key result showed that social conventions could emerge much more quickly amongst a population of agents with small-world or scale-free interac-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

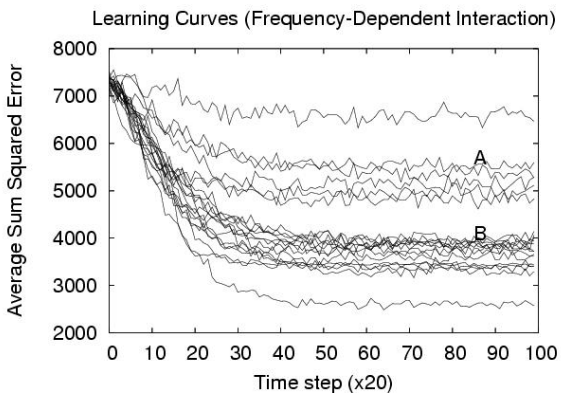


Figure 1: The learning curves for frequency-dependent selection.

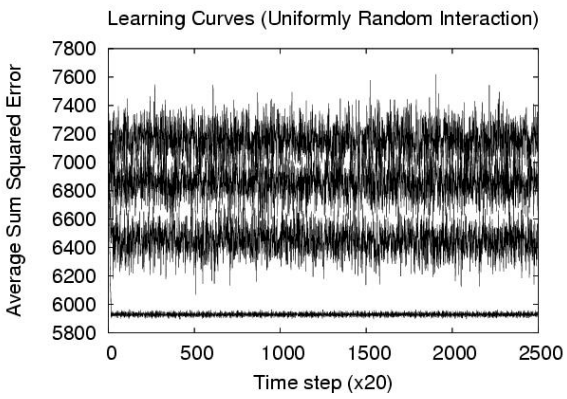


Figure 2: The learning curves for uniformly random interaction.

tion graph topologies rather than regular interaction graphs. The interaction graph topology was prespecified though, and there was no learning in the model.

Lee et. al [4] empirically studied the effect of interaction topology on the critical learning fidelity in the Komarova et. al. model [3]. Although several different interaction topologies were used, the topologies were not emergent and were prespecified by the creator of the experiments.

Our agents use simple recurrent neural networks to generate, parse, and learn languages from each other. Further, the interaction topology emerges in a distributed manner as a consequence of a specific rational agent strategy, based on our own Noisy Preferential Attachment (NPA) algorithm [6]. These two mechanisms, taken together, capture the desirable features of *agent autonomy* (learning and choice), *realism* (*emergent* global interaction patterns), and *flexibility* (continuous adaptation of the collective language space).

3. EXPERIMENTAL FRAMEWORK

Fitness: In our model, Each agent has a fitness, defined as follows:

$$f_i(t) = \alpha L_i(t) + (1 - \alpha)K_i(t). \quad (1)$$

$L_i(t)$ is the intrinsic fitness of the language used by agent i at time t, which may correspond to its expressivity or how good it is at describing a task. $K_i(t)$ is the number of times agent i has communicated a set of messages to another agent up to time-step t. $K_i(t)$ is normalized over the population. α is a constant representing a tradeoff between these two components of fitness.

Interaction topology: At each time-step, an agent chooses another agent with whom to interact, with probability proportional to the fitness of that agent. With a small probability p, however, it might switch to another agent chosen uniformly randomly. This "noise" is helpful in keeping the topology from getting too centralized.

The intuition is that if a large number of agents are listening to a particular agent, then it makes sense to pay more attention to that particular agent because its language is probably shared by many agents. The key thing here is that the central teacher agent *emerges* from the interactions between agents, rather than being prespecified.

The parameter p also models incomplete knowledge of the interaction history of the entire population. The assumption is that, since there is no central controller telling each agent the fitness of all the other agents, each agent must maintain an estimate of the fitness of the other agents. Since the agents may not be able to observe all the other agents all the time, this estimate will have some error. It turns out that we can take advantage of this error to produce scale-free topologies. In the absence of error, i.e. if $p = 0$, frequency-dependent selection would result in a pin-wheel topology. This is because initially none of the nodes in the graph have any links, and so the first node to acquire a link acquires all the links. However, when $p \neq 0$, but is small, it can be shown that the resulting topologies have power-law like or exponential degree distributions. This model for determining interactions is derived from a version of the preferential attachment algorithm [1], which we call *Noisy Preferential Attachment* (NPA), that is based on the replicator-mutator equation of biological evolution.

4. EXPERIMENTS AND RESULTS

We did an experiment with a population of 200 agents. The intrinsic fitness term for each agent, L_i, was set to zero, because in the current experiments the agents are not situated, i.e. they are not performing a particular task, nor are they receiving any external inputs except communication from other agents.

The parameter p, which is the probability of choosing a partner uniformly randomly, was set to 0.05. The maximum sentence length was set to 10. The number of agents sampled by the learner, m, was set to 1.

The set of terminals for all the languages was the same, and consisted of five symbols: `a, b, c, d, e`. Each agent's SRN had five input nodes, six hidden layer nodes, and five output nodes. The symbols were encoded using a 1-of-n encoding, and all weights were initialized uniformly randomly in the range $[-0.5, 0.5]$.

The simulation was run for 2000 time steps, and coherence of the population was evaluated every 20 time steps. Coherence is evaluated as the average error of an agent on a set of sentences sampled from the entire population. An agent only took the last 300 time steps into account when selecting an agent to interact with. The simulation was run twenty times with identical parameters, and the results are shown in figures 1 and 2.

	a	b	c	d	e	none	mean frequency	variance
Freq-dep Interaction	5	0	13	1	0	1	.7715	.008
– Cluster A	1	0	5	0	0	0	.644	.005
– Cluster B	4	0	8	1	0	0	.811	.001
Uniform Interaction	0	0	0	0	0	20	nan	nan

Table 1: The dominating language in the 20 frequency-dependent interaction runs and the 20 uniformly random interaction runs. The letter dominance threshold was set to 0.75 and the language dominance threshold was set to 0.5. Mean frequency and variance is the average and variance of the proportions of the dominating language.

A key difference between the runs is that the networks in clusters A and B are dominated by a single node, whereas the outliers both have multiple nodes with high in-degree. This difference can be objectively measured by calculating the *betweenness centrality* for the networks. Betweenness centrality is a value that measures how close a network is to a pin-wheel structure. It is a number between 0 and 1, and a low value means that the network is closer in topology to a ring or a fully-connected network. As shown in table 2, the average betweenness centrality for clusters A and B is very close to 1, whereas the value for the outliers is much smaller.

Network	Average betweenness centrality
High outlier	0.5451
Cluster A	0.9348
Cluster B	0.9016
Low outlier	0.7881

Table 2: The betweenness centrality for a network measures to what extent it is dominated by a single node.

The surprising fact here is that betweenness centrality is smaller for both the high and the low outlier. This can be explained by examining the actual languages to which the populations converge.

An agents language is dominated by the letter `a` if the frequency of the letter `a` is higher than some *letter dominance threshold*. A run is dominated by letter `a` if the proportion of agents using `a` dominated languages is higher than the *language domination threshold*. For each run of the simulation, we found the dominant language, or whether the run was not dominated.

Table 1 shows the distribution of dominating languages. In nearly all the frequency-dependent interaction runs, there was a dominating language. The average of the proportion of the dominating language is .7715. The one run that did not converge to a dominating run is the high outlier in Figure 1. In this case, the language distribution was split between an `a` language and a `b` language, which is why there is a high error for this run. This run resulted in a network with low betweenness centrality, just like the low outlier, but the difference is that in this run, the two most connected nodes in the network correspond to agents with different languages. In the low outlier, the most connected nodes all correspond to agents with the same language.

5. DISCUSSION AND FUTURE WORK

We showed that by incorporating a frequency-dependent measure in the definition of fitness for an agent, we can generate scale-free topologies, which lead to quick convergence. This definition of fitness makes intuitive sense: if many agents are talking to a particular agent, then it pays to talk to that agent more, since it probably speaks a widely-spoken and useful language.

However, the tradeoff in this case is convergence vs robustness. Networks with high betweenness centrality, as are generally observed in our simulations, are very susceptible to attack, because removing the dominant node will cause the network to fall apart. This suggests one direction for future work: how do we converge to a stable language if agents have finite lifetimes?

The other observation is that the population converges to a "simple" language, i.e. a language consisting entirely of strings of a single symbol. It may be the case that in the absence of some external pressure towards a complex language, populations always tend towards a maximally simple common language as it is the most easily learnable. This suggests another natural followup: how do we converge to a "good" language? The "goodness" of a language can be incorporated into our framework using the frequency-independent part of fitness. Thus, agents that are particularly good at solving some task and talking about it may have high frequency-independent fitness. Therefore we intend to study convergence in a situated population of agents, where the environment would provide the pressure towards a more complex language.

6. ACKNOWLEDGEMENTS

We thank the UIUC Language Evolution Group for very helpful and stimulating discussions. This work was supported in part by NSF grant IIS-0340996.

7. REFERENCES

[1] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, October 1999.

[2] J. Delgado. Emergence of social conventions in complex networks. *Artificial Intelligence*, 141:171–185, 2002.

[3] N. Komarova, P. Niyogi, and M. Nowak. Evolutionary dynamics of grammar acquisition. *Journal of Theoretical Biology*, 209(1):43–59, 2002.

[4] Y. Lee, T. C. Collier, C. E. Taylor, and E. E. Stabler. The role of population structure in language evolution. In *Proceedings of the 10th International Symposium on Artificial Life and Robotics*, 2005.

[5] Y. Shoham and M. Tennenholtz. On the emergence of social conventions: modeling, analysis, and simulations. *Artificial Intelligence*, 94(1-2):139–166, jul 1997.

[6] S. Swarup and L. Gasser. Unifying network and evolutionary dynamics. Technical Report UIUCLIS–2006/1+AMAG.

Organizational Petri Nets for Protocol Design and Enactment

Stéphanie Combettes
University Toulouse 3
115, Route de Narbonne
31077 Toulouse, France
stephanie.combettes@iut-tlse3.fr

Chihab Hanachi
IRIT - University Toulouse 1
Place Anatole France
31042 Toulouse, France
hanachi@univ-tlse1.fr

Christophe Sibertin-Blanc
IRIT - University Toulouse 1
Place Anatole France
31042 Toulouse, France
sibertin@univ-tlse1.fr

ABSTRACT
To integrate protocols in real organizations, we need to take into account three complementary dimensions: the protocols behavior, the structure and deontic aspects of the organization being considered and its informational model. Basic Petri Nets (PN) are known to deal suitably with the behavioral aspect while some high level extensions are intended to take into account one of the additional dimensions. This paper proposes a formalism called *Organizational Petri nets* to capture these three dimensions in a coherent framework. We give the definition of this new formalism, precise its semantics and illustrate its use through an example.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *Petri Nets, Object-Oriented design Methods*
D.3.1 [**Programming languages**]: Formal Definitions and Theory – *Semantics, syntax.*

General Terms
Design, Theory.

Keywords
Petri Net, deontic aspect, protocol, agent.

1. INTRODUCTION
Protocols are one of the basic building blocks [7] for structuring the design of Multi-Agent Systems (MAS) and coordinating their execution. In real-world applications such as Agent-Oriented Information Systems (AOIS), protocols involve three inter-dependant dimensions: the behavioral, organizational and informational ones. These three dimensions may capture by the three inter-related models described below. *The behavioral model* describes the protocol control structure, i.e. the actions (performatives sending or receiving, elementary actions) constituting the steps of a protocol, and their coordination (a pre-order relation between these interventions). *The organizational model* should refer to the logical actors able to carry out the activity of the organization, in order to define the specific contribution of each role with regard to the others and the rule applied to grant these roles to the concrete actors. This determines "what agents have permission to do within a protocol". Moreover, AOIS applications – such as e-alliance, inter-organizational workflow or e-institutions – often deal with documents conform to regulations like contracts or norms. In this case two other deontic aspects should be represented: obligation and prohibition. Finally, the *informational model* represents documents, forms and data having attributes that are handled and exchanged by the actors of the organization, possibly through the contents of communicative acts.

Although High Level Petri Nets are suitable for specifying, validating, simulating and even implementing protocols (see [1],[2],[3]), when referring to the state of the art, no Petri net dialect meets all the previous requirements. However, two works merit to be examined. The first one is due to Raskin and al. [9] who extend Petri nets by integrating deontic aspects. Unfortunately, this work is based on basic Petri nets and does not allow to represent the actors and information involved in Protocols. In other words, protocols cannot be described within their execution context. The second work, by Sibertin [5], proposes the Petri Nets with Objects (PNO) formalism which combines coherently the Petri net theory and the Object-Oriented (OO) approach. While PN are very suitable to express the dynamic behavior of protocols, the OO approach permits the modeling and structuring of its active (actors) and passive (information resources) entities. In a basic PN, tokens are atomic undistinguishable items, whereas in a PNO they are objects provided with attributes and methods. Unfortunately, PNO do not take into account the deontic aspect of Protocols. Notably, they do not give means to represent the obliged and prohibited actions.

Consequently, the aim of this paper is to provide a High Level Petri net language integrating the above three dimensions (behavioral, organizational and informational) to represent AOIS protocols. The proposed formalism is a combination of the two above-mentioned languages: Deontic Petri Nets [4] and PNO [5]. After presenting the formalism, we develop an example for the modeling of the book borrowing procedure of a library.

2. ORGANIZATIONAL PETRI NETS

2.1 Syntactic Definition of OgPN
Definition 1 (OgPN): An Organizational Petri Net (OgPN) is a n-uple (C,P,T,V, PreCd, Act, EmR, Pre, Post, Resp, Cost, CTD):

- **C** is the set of classes of the object processed by the net, **C** = **A** ∪ **D** where **A** contains the classes of active objects (the organizational actors) and **D** contains the classes of passive data or resources; any object class in *A* is assumed to have a predefined *penalty* attribute;
- **P** is the set of *places*, typed by a function ***P→ C****;

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

- **T** is the set of *transitions*, **T = Id ∪Vi ∪ Re** where these sub-sets are respectively the sets of *ideal*, *violation* and *repair* transitions. Transitions in **Vi** and **Re** are the sub-ideal ones;
- **V** is the set of *variables* that label the arcs;
- **PreCd** is the set of *preconditions* that guard transitions;
- **Act** is the set of *actions* performed on transition occurrences;
- **EmR** is the set of *emission rules* that control the direction of tokens at the completion of transitions;
- **Pre** is the *forward incidence function*: $\boldsymbol{P}$ x $\boldsymbol{T} \rightarrow \boldsymbol{V}$ *;
- **Post** is the *backward incidence function*: $\boldsymbol{T}$ x $\boldsymbol{EmR}$ x $\boldsymbol{P} \rightarrow \boldsymbol{V}$*;
- **Resp** is the function that associates to each sub-ideal transition a variable corresponding to the *responsible actor*: **Vi ∪ Re → V** where the type of **Resp***(t)* belongs to *A*;
- **Cost** is the function that states the punishment or reward resulting from the occurrence of a transition *t*; if t ∈ **Id** then **Cost**(t) = 0; if t∈**Vi** then **Cost**(t)> 0; if t∈**Re** then **Cost**(t) < 0;
- **CTD ⊂ Vi x Re**, is a relation which associates each repair transition to violation transitions and thus defines the *Contrary To Duty* obligations: (t, t') ∈ **CTD** ⇔ t' repairs t.

Each sub-ideal transition must have an actor responsible for that transition, which will assume the cost of performing this transition. Accordingly, a violation or repair transition has an input variable whose type belongs to *A*, and at each occurrence of the transition, the actor linked to that variable is charged with the value of the *Cost* function. The *penalty* attribute of the object classes in *A* is intended to store the amount of each actor penalties.

2.2 Semantics of Organizational Petri Nets

The *Organizational Petri Nets* formalism is a combination of the *Deontic Petri Net* and *Petri Net with Object* formalisms, which both have a well-funded semantics extending the *basic Petri nets* semantics. In this line, the OgPN semantics includes the PNO semantics (as defined in [6]) and introduces into actor objects the DPN's computation of penalties.

A transition may occur, by a *substitution principle* [5]**,** only if it can pick tokens in its input places in such a way that each of its input variables is bound to an element of its domain of value.

Definition 2 (Transition firing): The occurrence of a transition *t* that may occur with a substitution S from a marking M yields the marking M' (noted $M \xrightarrow{t,s} M'$) by following this 5-steps process:

1. Withdraw from the input places of *t* the tokens bound to its input variables: M'(p) := M(p) \ S(***Pre***(p, t));
2. Generate new objects, if needed to instantiate the output variables of *t* that are not already input variables;
3. Perform the action $\boldsymbol{Act_t}$ of *t,* that most often consists in calling methods or updating values of attributes of the objects bound to the transition variables;
4. If the transition *t* is a violation or repair one, update the penalty of the actor responsible for the occurrence of *t*: a.penalty := a.penalty + ***Cost***(t), where a = S(***Resp***(t));
5. Evaluate emission rules $\boldsymbol{EmR_t}$ of *t* to decide the one r that is satisfied, and put tokens bound to variables into the appropriate output places: M'(p) := M(p) + S(***Post***(t, r, p)).

Steps 1 and 5 define the location of tokens into places, while steps 2 to 4 compute the value of these tokens.

The **cost** associated to the executions of a transition is a penalty of the actor responsible for that transition occurrence. Provided that its initial value is null, the penalty attribute of an actor is the sum of the costs of the transition occurrences he is responsible for. Most often, the execution of a sequence of transitions is more significant than the occurrence of a single transition in isolation. Let s = t1**.** … **.**tn ∈ T* be a sequence of transitions and M a marking. *s* is enabled from M and its occurrence yields the marking M' if substitutions S1, …, Sn and marking M1,… Mn=M' such that $M \xrightarrow{t1,s1} M1 \xrightarrow{t2,s2} \dots \xrightarrow{tn,sn} Mn$ exist.

According to the semantics of DPN, the **cost of a transition sequence** $s = t_1 \dots t_n$ is the sum of the costs associated to the transitions in *s*: $\boldsymbol{Cost}(s) = \sum_{i=1..n} \boldsymbol{Cost}(t_i)$. If an actor *a* is responsible for all the transitions of a sequence *s*, the performance of this sequence will update his penalty in the following way: a.penalty := a.penalty + ***Cost*** (s).

3. EXAMPLE

Let us first describe the protocol for borrowing books from a library. Someone, who has borrowed a book, must return it in time and without damages. If the borrower delays the book delivery he is penalized. Likewise, if he damages the book, he is first penalized but he may repair the damage. In case of repairing, his penalty is reduced.

The design of an OgPN consists in three inter-dependant stages that correspond to the production of three models: informational, organizational and behavioral models.

The *informational model* describes the structure of the information together with some integrity constraints. In an OgPN, information units are represented as object classes, using the powerful mechanisms (such as classification, inheritance, polymorphism…), that allow to describe complex data structure (forms, rules…) usually used in normative systems. These objects will flow in the OgPN and are transformed by actions performed by the actors. In our example, we only consider the attributes of the Book class (see table 1).

Table 1. The Book Class

Class Book
Attributes:
Title: string
Author: string
Reference: string
ReturnDate: date
Aspect: {Good, Damaged, Repaired}
State: {Available, Returned, Borrowed}
End

The *organizational model* structures the actors in object classes sharing the same features. A class represents a role and can comprise actors having the same capabilities or belonging to the same organizational unit. It also attributes to each actor, permission to perform actions on objects of the informational model and the responsibility of some actions that violate the rules of the organization. In our context, roles are described as objects or agent classes; the permissions are implicitly given by the declaration of method's parameters, which refer the objects classes of the informational model. The inheritance mechanism propagates permissions between roles. An actor is responsible for

all actions implemented by methods, it can perform, and its "*penalty*" attribute records its amount of penalties.

The organizational model is limited in our example to a single role described by the `Borrower` class (see table 2). This role contains attributes and methods, and has the permission to handle books since this type of object appears as a parameter of the methods. The `penalty` attribute is updated each time an agent playing the role transgresses a deontic rule expressed in the Petri net.

Table 2. The Borrower Class

Class Borrower
Attributes:
Name: string
Penalty: integer
Methods:
Borrow (B : Book) % triggered when the borrower borrows the book
Read (B : Book) % triggered when he reads the book
ReturnLate (B : Book) % triggered when he returns the book late
Return (B : Book) % triggered when he returns the book
Repair (B : Book) %triggered when he repairs the damaged book
Damage (B : Book) %triggered when he damages the book
End

Finally, the *behavioral model* describes how the protocols behave by defining the action sequences and the information and roles involved in each action. Thanks to the high expressive power of Petri nets, the designer may express a lot of control structures (parallelism, sequence, alternative…), which represent deontic constraints between two actions. The permissions granted to a role are expressed by the nature of the arc between an information place and a transition. The ideal, violation and repair actions are represented by transitions with their respective cost. Violation and repair transitions charge the actor responsible for that transition.

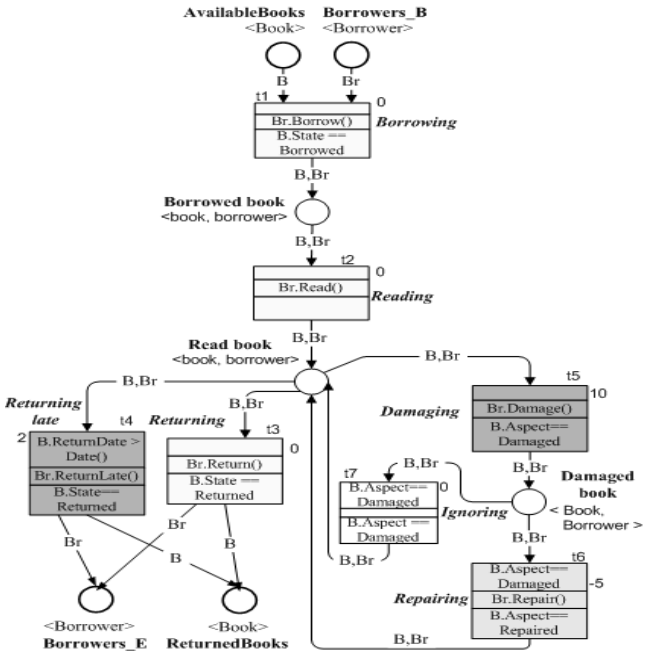


Figure 1. OgPN modeling borrower behavior

Regarding the behavioral model (see figure 1), the organizational Petri net includes seven transitions and six places typed with the two previously defined classes. The two initial places (`AvailableBooks` and `Borrowers_B`) represent a starting situation where a Borrower arrives in order to borrow an available Book. The two final places (`ReturnedBooks` and `Borrowers_E`) correspond to a situation where the Borrower has returned the Book. The transitions are grayed to distinguish more easily between ideal transitions (white: t1, t2, t3, t7), transgression transitions (dark grey: t4, t5) and repair transitions (bright grey: t6). The number associated to each transition represents its cost. For example, the transition `t6` called `Repairing` is a repair transition and its cost is "-5". Here the responsible actor is implicit because there is only one actor. However in case of ambiguity, the cost is prefixed with the responsible class name: <name>: cost. Let us explain a scenario represented by sequence s2 = t1.t2.t5.t6.t4 corresponding to the case where the Borrower borrows the Book (t1), reads it (t2), damages it (t5), repairs it (t6) and finally returns it late (t4). The penalty gained by the Borrower agent linked to the `Br` variable during an execution of this sequence is the sum of the costs of the transitions, that is 10 – 5 + 2 = 7.

Let us notice that this OgPN is an executable specification that can be used for simulation purpose. To completely control the operational functioning of a library, the code of the methods of object classes must be given.

4. CONCLUSIONS

We gave the formal semantics of the formalism called *Organizational Petri Nets* and some guidelines to produce models through an example. The advantages of this formalism are: 1) an easy integration in real world organization 2) the possibility of simulating protocols before their deployment 3) behavioral analysis capabilities 4) an operational semantics. In future work, we will investigate how this formalism can help to support the notion of protocol in existing agent-oriented methodologies like GAIA.

5. REFERENCES

[1] Cost R.S., Chen Y., Finin T., Labrou Y., Peng Y., *Modeling Agent Conversations with Colored Petri Nets*. Workshop on Specifying and Implementing Conversation Policies, Seattle,1999, pp. 59-66.

[2] Hanachi C., Sibertin-Blanc C., *Protocol Moderators as active Middle-Agents in Multi-Agent Systems*. Journal of AAMAS, 8, 131-164, 2004.

[3] Mazouzi H., EL Fallah-Seghrouchni A., Haddad S., *Open Protocol Design for complex Interactions in Multi-agent Systems*. First International Joint Conference on AAMAS, Italy, 2002.

[4] Raskin J.F., Tan Y.H. and van der Torre L.W.N. *Modeling Deontic States in Petri Nets.* Seventh Dutch AI Conference (NAIC-95), the Netherlands, 1995.

[5] Sibertin-Blanc C. *High Level Petri Nets with Data Structure*. 6th European Workshop on Application and Theory of Petri Nets, p141 - 170, 1985.

[6] Sibertin-Blanc C. *A functional semantics of Petri Nets with Objects*. IRIT Report, University of Toulouse 1, 1992.

[7] Sturm A., Shehory O. *A Framework for Evaluating Agent-Oriented Methodologies*. Agent Oriented Information System Workshop, 2003: 94-109.

Optimization and constraint processing

Nogood-based Asynchronous Distributed Optimization (ADOPT-ng)

Marius C. Silaghi
Florida Institute of Technology
msilaghi@fit.edu

Makoto Yokoo
Kyushu University
yokoo@is.kyushu-u.ac.jp

ABSTRACT

This work proposes an asynchronous algorithm for solving Distributed Constraint Optimization problems (DCOPs) using a new kind of nogoods, namely valued nogoods. The proposed technique is an extension of the asynchronous distributed optimization (ADOPT) where valued nogoods enable more flexible reasoning, leading to important speed-up. Valued nogoods are an extension of classic nogoods that associates each nogood with a threshold and optionally with a set of references to culprit constraints.

DCOPs have been shown to have very elegant distributed solutions, such as ADOPT, distributed asynchronous overlay (DisAO), or DPOP. These algorithms are typically tuned to minimize the longest causal chain of messages, as a measure of how the algorithms will scale for systems with remote agents (with large latency in communication). Among the mentioned techniques, DPOP performs very well for the chosen metric (requiring a number of such sequential messages linear in the number of agents), but in general has exponential space requirements. DisAO and ADOPT have the advantage of needing only polynomial space. ADOPT has the property of maintaining the initial distribution of the problem. ADOPT needs a preprocessing step consisting of computing a depth first search (DFS) tree on the agent graph. We show that valued nogoods reduce the practical importance/need of this preprocessing since independent subproblems are now dynamically detected and exploited.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multi-agent systems*

General Terms

Algorithm

Keywords

Distributed Constraint Optimization

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. INTRODUCTION

Distributed Constraint Optimization (DCOP) is a formalism that can model naturally distributed problems. These are problems where agents try to find assignments to a set of variables subjects to constraints. The natural distribution comes from the assumption that only a subset of the agents has knowledge of each given constraint. Nevertheless, in DCOPs it is assumed that agents try to maximize their cumulated satisfaction by the chosen solution. This is different from other related formalisms where agents try to maximize the satisfaction of the least satisfied among them [22].

Several synchronous and asynchronous distributed algorithms have been proposed for distributedly solving DCOPs. Since a DCOP can be viewed as a distributed version of the common centralized Valued Constraint Satisfaction Problems (VCSPs), it is normal that successful techniques for VCSPs were ported to DCOPs. However the effectiveness of such techniques has to be evaluated from a different perspective (and different measures) as imposed by the new requirements. Typically research has focused on techniques in which reluctance is manifested towards modifications to the distribution of the problem (modification accepted only when some reasoning infers it is unavoidable for guaranteeing to reach solutions). This criteria is largely believed valuable and adaptable for large, open, and/or dynamic distributed problems. It is also perceived as an alternative approach to privacy requirements [15, 21, 26, 17].

A synchronous algorithm, synchronous branch and bound, was the first known distributed algorithm for solving DCOPs [7]. Stochastic versions have also been proposed [27]. From the point of view of efficiency, a distributed algorithm for solving DCOPs is typically evaluated with regard to applications to agents on the Internet, namely where latency in communication is significantly higher than local computations. A measure representing this assumption well is given by the number of cycles of a simulator that lets each agent in turn process all the messages that it receives [23]. Within the mentioned assumption, this measure is equivalent for real solvers to the longest causal chain of sequential messages, as used in [18].

From the point of view of this measure, a very efficient currently existing DCOP solver is DPOP [12, 11], which is linear in the number of variables. However, in general that algorithm has message sizes and local computation costs that are exponential in the induced width of a chosen depth-first search tree of the constraint graph of the problem, clearly

[0]We thank Benjamin Pflanz, Boi Faltings, Rina Dechter, Adrian Petcu and reviewers for useful comments.

invalidating the assumptions that lead to the acceptance of the number of cycles as efficiency measure.

Two other algorithms competing as efficient solvers of DCOPs, are the asynchronous distributed optimization (ADOPT) and the distributed asynchronous overlay (DisAO). DisAO works by incrementally joining the subproblems owned by agents found in conflict. ADOPT implements a parallel version of A* [16]. While DisAO is typically criticized for its extensive abandon of the maintenance of the natural distributedness of the problem at the first conflict (and expensive local computations invalidating the above assumptions like DPOP [4, 8, 1]), ADOPT can be criticized for its strict message pattern that only provides reduced reasoning opportunities. ADOPT also works only on special orderings on agents, namely dictated by some Depth First Search tree on the constraint graph.

It is easy to construct huge problems whose constraint graphs are forests and that are easily solved by DPOP (in linear time), but unsolvable with the other known algorithms. It is also easy to construct relatively small problems whose constraint graph is full and therefore require unacceptable (exponential) space with DPOP, while being easily solvable with algorithms like ADOPT, e.g. for the trivial case where all tuples are optimal with cost zero.

In this work we address the aforementioned critiques of ADOPT showing that it is possible to define a message scheme based on a new type of nogoods, called *valued nogoods*, that not only virtually eliminates the need of precomputing a DFS tree of the constraint graph, but also leads to significant improvement in efficiency. Nogoods are at the basis of much flexibility in asynchronous algorithms. A nogood specifies a set of assignments that conflict constraints [20]. A basic version of the valued nogoods consists in associating each nogood to a threshold, namely a cost limit violated due to the assignments of the nogood. It is significant to note that the valued nogoods lead to efficiency improvements even if used in conjunction with a DFS tree, instead of the less semantically explicit cost messages of ADOPT. We find that a version of valued nogoods that is associated with a list of culprit constraints [13] produces additional important improvements. Each of these incremental concepts and improvements is described in the following sections.

2. DFS-TREES

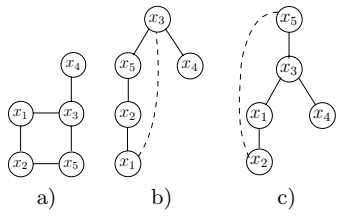


Figure 1: For a DCOP with primal graph depicted in (a), two possible DFS trees (pseudotrees) are (b) and (c). Interrupted lines show constraint graph neighboring relations not in the DFS tree.

The primal graph of a DCOP is the graph having the variables as nodes and having an arc for each pair of variables linked by a constraint [5]. A Depth First Search (DFS) tree associated to a DCOP is a spanning tree generated by the arcs used for visiting once each node during some depth first traversal of its primal graph. DFS trees were first successfully used for distributed constraint problems in [3]. The property exploited there is that separate branches of the DFS-tree are completely independent once the assignments of common ancestors are decided. Two examples of DFS trees for a DCOP primal graph are shown in Figure 1.

Nodes directly connected to a node in a primal graph are said to be its *neighbors*. In Figure 1.a, the neighbors of x_3 are $\{x_1, x_5, x_4\}$. The *ancestors* of a node are the nodes on the path between it and the root of the DFS tree, inclusively. In Figure 1.b, $\{x_5, x_3\}$ are ancestors of x_2. x_3 has no ancestors.

3. ADOPT

ADOPT [9] is an asynchronous complete DCOP solver, which is guaranteed to find an optimal solution. Here, we only show a brief description of ADOPT. Please consult [9] for the detail. First, ADOPT organizes agents into a Depth-First Search (DFS) tree, in which constraints are allowed between a variable and any of its ancestors or descendants, but not between variables in separate sub-trees.

ADOPT uses three kinds of messages: VALUE, COST, and THRESHOLD. A VALUE message communicates the assignment of a variable from ancestors to descendants who share constraints with the sender. When the algorithm starts, each agent takes a random value for its variable and sends appropriate VALUE messages. A COST message is sent from a child to its parent, which indicates the estimated lower-bound of the cost of the subtree rooted at the child. Since communication is asynchronous, a cost message contains a context, i.e., a list of the value assignments of the ancestors. The THRESHOLD message is introduced to improve the search efficiency. An agent tries to assign its value so that the estimated cost is lower than the given threshold communicated by the THRESHOLD message from its parent. Initially, the threshold is 0. When the estimated cost is higher than the given threshold, the agent opportunistically switches its value assignment to another value that has the smallest estimated cost. Initially, the estimated cost is 0. Therefore, an unexplored assignment has an estimated cost 0. A cost message also contains the information of the upper-bound of the cost of the subtree, i.e., the actual cost of the subtree. When the upper-bound and the lower-bound meet at the root agent, then a globally optimal solution has been found and the algorithm is terminated.

4. DISTRIBUTED VALUED CSPS

Constraint Satisfaction Problems (CSPs) are described by a set X of variables and a set of constraints on the possible combinations of assignments to these variables with values from their domains.

DEFINITION 1 (DCOP). *A distributed constraint optimization problem (DCOP), aka distributed valued CSP, is defined by a set of agents $A_1, A_2, ..., A_n$, a set X of variables, $x_1, x_2, ..., x_n$, and a set of functions $f_1, f_2, ...f_i, ..., f_n$, $f_i : X_i \rightarrow \mathbb{R}$, $X_i \subseteq X$, where only A_i knows f_i.*

The problem is to find $\operatorname{argmin}_x \sum_{i=1}^{n} f_i(x_{|X_i})$. *We assume that x_i can only take values from a domain $D_i = \{1, ..., d\}$.*

For simplification and without loss of generality one typically assumes that $X_i \subseteq \{x_1, ..., x_i\}$

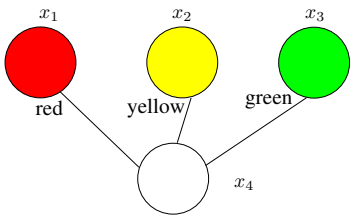


Figure 2: MIN resolution on valued global nogoods

Our idea can be easily applied to general valued CSPs.

5. COST OF NOGOODS

Previous flexible algorithms for solving distributed constraint satisfaction problems exploit the inference power of nogoods (e.g., ABT, AWC, ABTR [23, 24, 19])[1]. A nogood $\neg N$ stands for a set N of assignments that was proven impossible, by inference using constraints. If $N = (\langle x_1, v_1\rangle, ..., \langle x_t, v_t\rangle)$ where $v_i \in D_i$, then we denote by $\overline{N}$ the set of variables assigned in N, $\overline{N} = \{x_1, ...x_t\}$.

In order to apply nogood-based algorithms to DCOP, we redefine the notion of nogoods as follows. First we attach a value to each nogood obtaining a *valued global nogood*.

DEFINITION 2 (VALUED GLOBAL NOGOOD). *A valued global nogood has the form $[c, N]$, and specifies that the (global) problem has cost at least c, given the set of assignments, N, for distinct variables.*

Given a valued global nogood $[c, (\langle x_1, v_1\rangle, ..., \langle x_t, v_t\rangle)]$, one can infer a *global cost assessment (GCA)* for the value v_t from the domain of x_t given the assignments $S = \langle x_1, v_1\rangle, ..., \langle x_{t-1}, v_{t-1}\rangle$. This GCA is denoted (v_t, c, S), and is semantically equivalent to an applied global value nogood, (i.e. the inference):

$$(\langle x_1, v_1\rangle, ..., \langle x_{t-1}, v_{t-1}\rangle) \rightarrow (\langle x_t, v_t\rangle \text{ has cost } c).$$

REMARK 1. *Given a valued global nogood $[c, N]$, one can infer the GCA (v, c, N) for any value v from the domain of any variable x, where x is not assigned in N, i.e., $x \notin \overline{N}$.*

E.g., if A_3 knows the valued global nogood $[10, \{(x_1, r), (x_2, y)\}]$, then it can infer for the value r of x_3 the GCA $(r, 10, \{(x_1, r), (x_2, y)\})$.

PROPOSITION 1 (MIN-RESOLUTION). *From a set $\{(v, c_v, S_v)\}$ containing exactly one GCA for each value v in the domain of variable x_i and $\forall k, j$, assignments for $\overline{S_k} \cap \overline{S_j}$ are identical in both S_k and S_j of a minimization VCSP, one can resolve a new valued global nogood: $[\min_v c_v, \cup_v S_v]$.*

Example 5.1 *For the graph coloring problem in Figure 2 (costs are not shown), x_1 is colored red (r), x_2 yellow (y) and x_3 green (g). Assume that the following valued global nogoods are known for each of the values $\{r, y, g\}$ of x_4:*

(r): $[10, \{(x_1, r), (x_4, r)\}]$, obtaining for x_4 the GCA $(r, 10, \{(x_1, r)\})$

(y): $[8, \{(x_2, y), (x_4, y)\}]$, obtaining for x_4 the GCA $(y, 8, \{(x_2, y)\})$

(g): $[7, \{(x_3, g), (x_4, g)\}]$, obtaining for x_4 the GCA $(g, 7, \{(x_3, g)\})$

By min-resolution on these GCAs one obtains the valued global nogood $[7, \{(x_1, r), (x_2, y), (x_3, g)\}]$, meaning that given the coloring of the first 3 nodes there is no solution with (global) cost lower than 7.

Min-resolution can be applied on valued global nogoods:

COROLLARY 1.1. *Assume $\mathcal{S}$ is a set of no-goods associated to the variable x_i, such that for each $[c_v, S_v]$ in $\mathcal{S}$, $\exists\langle x_i, v\rangle \in S_v$. If $\mathcal{S}$ contains exactly one global valued nogood $[c_v, S_v]$ for each value v in the domain of variable x_i of a minimization VCSP, i.e. $\cup_v\{v\} = D_i$, one can resolve a new valued global nogood: $[\min_v c_v, \cup_v(S_v \setminus \langle x_i, v\rangle)]$.*

REMARK 2 (DFS SUBTREES). *Given two GCAs (v, c'_v, S'_v) and (v, c''_v, S''_v) for a value v in the domain of variable x_i of a minimization VCSP, if one knows that the two GCAs are inferred from different constraints then one can infer a new GCA: $\langle c'_v + c''_v, S'_v \cup S''_v\rangle$. This is similar to what ADOPT does to combine cost messages coming from disjoint problem subtrees [10, 3].*

This powerful reasoning can be applied when combining a nogood obtained from the local constraints with a valued nogood received from other agents (and obtained solely by inference on other agents' constraints). When a DFS tree of the constraint graph is used for constraining the message pattern as in ADOPT, this powerful inference applies, too.

The question is how to determine that the two GCAs are inferred from different constraints, in a more general setting. We introduce tagging of cost assessments with the identifiers of the constraints used to infer them.

DEFINITION 3 (SRC). *A set of references to constraints is a set of identifiers, each of them for a distinct constraint.*

Note that several constraints of a given problem description can be composed in one constraint (in a different description of the same problem).

REMARK 3. *For privacy, a constraint can be represented by several constraint references and several constraints of an agent can be represented by a single constraint reference.*

Now we generalize the concept of *valued global nogood* into the concept of *valued nogood*.

DEFINITION 4 (VALUED NOGOOD). *A valued nogood has the form $[SRC, c, N]$ where SRC is a set of references to constraints having cost at least c, given a set of assignments, N, for distinct variables.*

Valued nogoods are generalizations of valued global nogoods. Valued global nogoods are valued nogoods whose SRCs contain the references of all the constraints.

Once we decide that a nogood $[SRC, c, (\langle x_1, v_1\rangle, ..., \langle x_i, v_i\rangle)]$ will be applied to a certain variable x_i, we obtain a cost assessment tagged with the set of references to constraints SRC[2], denoted $(SRC, v_i, c, (\langle x_1, v_1\rangle, ..., \langle x_{i-1}, v_{i-1}\rangle))$.

[1]Other algorithms, like AAS, exploit generalized nogoods (i.e., extensions of nogoods to sets of values for a variable), and the extension of the work here for that case is pointed to in [13]

[2]This is called a *valued conflict list* in [13]

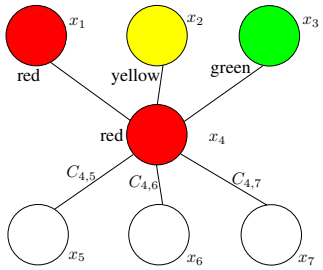


Figure 3: SUM-inference resolution on CAs

DEFINITION 5 (COST ASSESSMENT (CA)). *A cost assessment of variable x_i has the form (SRC, v, c, N) where SRC is a set of references to constraints having cost with lower bound c, given a set of assignments, N, for distinct variables, and the assignment of x_i to some value v.*

As for valued nogoods and valued global nogoods, cost assessments are generalizations of global cost assesments.

REMARK 4. *Given a valued nogood $[SRC, c, N]$, one can infer the CA $(SRC, v, c, N))$ for any value v from the domain of any variable x, where x is not assigned in N, i.e., $x \notin \overline{N}$. E.g., if A_6 knows the valued nogood $[\{C_{4,7}\}, 10, \{(x_2, y), (x_4, r)\}]$, then it can infer for the value b of x_6 the CA $(\{C_{4,7}\}, b, 10, \{(x_2, y), (x_4, r)\})$.*

We can now detect and perform the desired powerful reasoning on valued nogoods and/or CAs coming from disjoint subtrees, mentioned in Remark 2.

PROPOSITION 2 (SUM-INFERENCE). *A set of cost assessments for the value v of some variable, (SRC_i, v, c_i, N_i) where $\forall i, j : i \neq j \Rightarrow SRC_i \cap SRC_j = \emptyset$, and the assignment of any variable x_k is identical in any N_i where x_k is present, can be combined into a new cost assessment. The obtained cost assessment is (SRC, v, c, N) such that $SRC{=}\cup_i SRC_i$, $c{=}\sum_i (c_i)$, and $N{=}\cup_i N_i$.*

Example 5.2*[!th] For the graph coloring problem in Figure 3, x_1 is colored red, x_2 yellow, x_3 green, and x_4 red. Assume that the following valued nogoods are known for (x_4, r):*

- $[\{C_{4,5}\}, 5, \{(x_2, y), (x_4, r)\}]$ *obtaining CA* $(\{C_{4,5}\}, r, 5, \{(x_2, y)\})$
- $[\{C_{4,6}\}, 7, \{(x_1, r), (x_4, r)\}]$ *obtaining CA* $(\{C_{4,6}\}, r, 7, \{(x_1, r)\})$
- $[\{C_{4,7}\}, 9, \{(x_2, y), (x_4, r)\}]$ *obtaining CA* $(\{C_{4,7}\}, r, 9, \{(x_2, y)\})$

Also assume that based on x_4's constraint with x_1, one has obtained for $\langle x_4, r\rangle$ the following valued nogood:

- $[\{C_{1,4}\}, 10, \{(x_1, r), (x_4, r)\}]$ *obtaining CA* $(\{C_{1,4}\}, r, 10, \{(x_1, r)\})$

Then, by sum-inference on these CAs one obtains for x_4 the CA $[\{C_{1,4}, C_{4,5}, C_{4,6}, C_{4,7}\}, r, 31, \{(x_1, r), (x_2, y)\}]$, meaning that given the coloring of the first 2 nodes, coloring x_4 in red leads to a cost of at least 31 for the constraints $\{C_{1,4}, C_{4,5}, C_{4,6}, C_{4,7}\}$.

REMARK 5 (SUM-INFERENCE FOR VALUED NOGOODS). *Sum inference can be similarly applied to any set of valued nogoods with disjoint SRCs and compatible assignments. The result of combining nogoods $[SRC_i, c_i, S_i]$ is $[\cup_i SRC_i, \sum_i c_i, \cup_i S_i]$. This can also be extended to the case where assignments are generalized to sets [13].*

The min-resolution proposed for GCAs translates straightforwardly for CAs as follows.

PROPOSITION 3 (MIN-RESOLUTION). *A set of cost assessments for x_i, (SRC_i, v_i, c_i, N_i) where $\cup_i \{v_i\}$ covers the whole domain of x_i and $\forall k, j$, assignments for $\overline{N_k} \cap \overline{N_j}$ are identical in both N_k and N_j, can be combined into a new valued nogood. The obtained valued nogood is $[SRC, c, N]$ such that $SRC{=}\cup_i SRC_i$, $c{=}\min_i(c_i)$ and $N{=}\cup_i N_i$.*

Example 5.3 *For the graph coloring problem in Figure 2, x_1 is colored red, x_2 yellow and x_3 green. Assume that the following valued nogoods are known for the values of x_4:*

(r): $[\{C_{1,4}\}, 10, \{(x_1, r), (x_4, r)\}]$ *obtaining CA* $(\{C_{1,4}\}, r, 10, \{(x_1, r)\})$

(y): $[\{C_{2,4}\}, 8, \{(x_2, y), (x_4, y)\}]$ *obtaining CA* $(\{C_{2,4}\}, y, 8, \{(x_2, y)\})$

(g): $[\{C_{3,4}\}, 7, \{(x_3, g), (x_4, g)\}]$ *obtaining CA* $(\{C_{3,4}\}, g, 7, \{(x_3, g)\})$

By min-resolution on these CAs one obtains the valued global nogood $[\{C_{1,4}, C_{2,4}, C_{3,4}\}, 7, \{(x_1, r), (x_2, y), (x_3, g)\}]$, meaning that given the coloring of the first 3 nodes there is no solution with cost lower than 7 for the constraints $\{C_{1,4}, C_{2,4}, C_{3,4}\}$.

As with valued global nogoods, the min-resolution could be applied directly on valued nogoods:

COROLLARY 3.1 (MIN-RESOLUTION ON NOGOODS). *From a set $\{[SRC_v, c_v, S_v)] | \exists v, \langle x_i, v\rangle \in S_v\}$ containing exactly one valued nogood for each value v in the domain of variable x_i of a minimization VCSP, one can resolve a new valued nogood: $[\cup_v SRC_v, \min_v c_v, \cup_v (S_v \setminus \langle x_i, v\rangle)]$.*

6. ADOPT WITH NOGOODS

We will now present a distributed optimization algorithm using valued nogoods, to maximize the efficiency of reasoning by exploiting increased flexibility. The algorithm can be seen as an extension of both ADOPT and ABT, and will be denoted Asynchronous Distributed OPTimization with valued nogoods (ADOPT-ng).

Like in ABT, agents communicate with **ok?** messages proposing new assignments of sender's variable, **nogood** messages announcing a nogood, and **add-link** messages announcing interest in a variable. Like in ADOPT, agents can also use **threshold** messages, but their content can be included in **ok?** messages.

For simplicity we assume in this algorithm that the communication channels are FIFO. Addition of counters to proposed assignments and nogoods can help to remove this requirement with minimal additional changes (i.e., older assignments and older nogoods for the currently proposed value are discarded).

6.1 Data Structures

Each agent A_i stores its agent view (received assignments), and its outgoing links (agents of lower priority than A_i and having constraints on x_i). Instantiations may be tagged with counters. To manage nogoods and CAs, A_i uses matrices l[1..d], h[1..d], ca[1..d][i+1..n], lvn[1..i][i..n], lr[i+1..n] and lastSent[1..i-1] where d is the domain size for x_i. *crt_val* is the current value A_i proposes for x_i.

- l[k] stores a CA for $x_i = k$, that is inferred solely from the constraints between x_i and prior variables.
- ca[k][j] stores a CA for $x_i = k$, that is obtained from valued nogoods received from A_j.
- lvn[k][j] stores the last valued nogood for variables with higher and equal priority than k and that is received from A_j, j>i. lvn[k][i] stores nogoods coming via **threshold/ok?** messages.
- lr[k] stores the last valued nogood received from A_k.
- h[k] stores a CA for x_i =k that is inferred from ca[k][j], lr[t], lvn[t][j], and l[k], for all j and t. Before inference, the valued nogoods in lvn[t][j] need to be first translated into a CAs as described in Remark 4.
- lastSent[k] stores the last valued nogood sent to A_k.

6.2 ADOPT with valued nogoods

The pseudocode for ADOPT-ng is given in Algorithm 1. To extract the cost of a CA we introduce the function *cost*(), $cost((SRC, T, c, v))$ returns c. The $min\_resolution(j)$ function applies the min-resolution over all values of the current agent, but using only CAs having no assignment from agents with lower priority than A_j (e.g., not using lvn[t][k] for $t > j$). The $sum\_inference()$ function used in Algorithm 1 applies the sum-inference on its parameters whenever this is possible (detects disjoint SRCs), otherwise selects the nogood with highest threshold or whose lowest priority assignment has the highest priority (this has been previously used in [2, 19]). Function $vn2ca(vn, i)$ transforms a valued nogood vn in a cost assesment for x_i. Its inverse is function $ca2vn$. Function $target(N)$ gives the index of the lowest priority variable present in the nogood N. Like with file expansion, "*" in an index of a matrix means the set obtained for all possible values of that index (e.g., lvn[*][t] stands for $\{lvn[k][t] \mid \forall k\}$).

An agent A_i stores several nogoods (CAs) coming from the same source A_t and applicable to the same value v (namely the one at ca[v][t], all those at lvn[k][t] for any k, and lr[t] when v is *crt_val*).

REMARK 6. *The order of combining CAs matters. Notation $lr[t]_{|v}$ stands for vn2ca(lr[t]) when lr[t]'s value for x_i is v, and $\emptyset$ otherwise. To compute h[v]:*

1. *a) When maintaining DFS trees, for a value v, CAs are combined for each DFS subtree s: tmp[v][s]=sum-inference$_{t \in s}$(ca[v][t], $lr[t]_{|v}$, lvn[*][t]); b) Else, CAs coming from each agent A_t are combined: tmp[v][t]=sum-inference(ca[v][t], $lr[t]_{|v}$, lvn[*][t]);*
2. *CAs from step 1 (a or b) are combined: h[v]=sum-inference(tmp[v][*]);*
3. *Add l[v]: h[v]=sum-inference(h[v], l[v]);*
4. *Add threshold: h[v]=sum-inference(h[v], lvn[*][i])*

In ADOPT-ng agents are totally ordered, A_1 having the highest priority and A_n the lowest priority. Each agent A_i starts by calling the init() procedure, which initializes its l with valued nogoods infered from local (unary) constraints. It assigns x_i to a value with minimal local cost, *crt_val*, announcing the assignment to lower priority agents in outgoing-links. The agents answer any received message with the correponding procedure: "**when** receive **ok?**", "**when** receive **nogood**", and "**when** receive **add-link**".

When a new assignment is learned from **ok?** or **nogood** messages, valued nogoods based on older assignments for the same variables are discarded and the l vector is updated. Received nogoods are stored in matrices ca, lr and lvn. The vector h is updated on any modification of l, lvn, lr or ca. A_i always sets its *crt_val* to the index with the lowest CA threshold in vector h (preferring the previous assignment). On each change to the vector h, its values are combined by min-resolution to generate new nogoods for each higher priority agent (or ancestor, in versions using DFS trees). The generation and use of multiple nogoods at once is already shown useful for the constraint satisfaction case in [25].

The threshold valued nogood tvn delivered with **ok?** messages sets a common cost on all values of the receiver (see Remark 4), effectively setting a threshold on costs below which the receiver does not change its value. This achieves the effect of THRESHOLD messages in ADOPT.

Intuitively, the convergence of ADOPT-ng can be noticed from the fact that valued nogoods can only monotonically increase valuation for each subset of the search space, and this has to terminate since such valuations can be covered by a finite number of values. If agents A_j, j<i no longer change their assignments, valued nogoods can only monotonically increase at A_i for each value in D_i: nogoods' thresholds only increase as they only change by sum-inference.

LEMMA 1. *ADOPT-ng terminates in finite time.*

PROOF. Given the list of agents $A_1, ..., A_n$, define the suffix of length m of this list as the last m agents. Then the result follows immediately by induction for an increasingly growing suffix (increasing m), assuming the other agents reach quiescence.

The first step of the induction (for the last agent) follows from the fact that the last agent terminates in one step if the previous agents do not change their assignments.

Assuming that the previous induction assertion is true for any suffix of k agents. Let us prove that it is also true for a suffix of k+1 agents: For each assignment of the agent A_{n-k}, the remaining k agents will reach quiescence, according to the assumption of the induction step, or the assignment's CA threshold increases. After values are proposed in turn and the smallest threshold reaches its highest estimate, agent A_{n-k} selects the best value and reaches quiescence. The other agents reach quiescence according to the induction step. □

LEMMA 2. *The last valued nogood sent by each agent additively integrates the non-zero costs of the constraints of all of its successors.*

PROOF. At quiescence, each agent A_k has received the valued nogoods describing the costs of each of its successors, in the list given by the used ordering on agents (or descendants in the DFS tree when a DFS tree is maintained).

```
procedure init do
    h[v] := l[v]:=initialize CAs from unary constraints;
    crt_val=argmin_v(cost(h[v]));
    send ok?(⟨x_i, crt_val⟩,∅) to all agents in outgoing-links;
when receive ok?(⟨x_j, v_j⟩, tvn) do
    integrate(⟨x_j, v_j⟩);
    if (tvn no-null and has no old assignment) then
        k:=target(tvn); // threshold tvn as common cost;
        lvn[k][i]:=sum-inference(tvn,lvn[k][i]);
    check-agent-view();
when receive add-link(⟨x_j, v_j⟩),A_j do
    add A_j to outgoing-links;
    if (⟨x_j, v_j⟩) is old, send new assignment to A_j;
when receive nogood(rvn, t) from A_t do
    foreach new assignment a of a linked variable x_j in
    rvn do
        integrate(a); // counters show newer assignment;
    if (an assignment in rvn is outdated) then
        if (some new assignment was integrated now) then
            check-agent-view();
        return;
    foreach assignment a of a non-linked variable x_j in
    rvn do
        send add-link(a) to A_j;
    if ((j := target(rvn)) == i) then
        vn2ca(rvn, i) → CA rca for a value v of x_i;
        ca[v][t]:=sum-inference(rca,ca[v][t]);
        lr[t]:=rca;
    else
        lvn[j][t]:=rvn;
    check-agent-view();
procedure check-agent-view() do
    for every(v ∈ D_i) update l[v] and recompute h[v];
    for every A_j with higher priority than A_i (respectively
    ancestor in the DFS tree, when one is maintained) do
        if (h has non-null cost CA for all values of D_i)
        then
            vn:=min_resolution(j);
            if (vn ≠ lastSent[j]) then
                send nogood(vn,i) to A_j;
    crt_val=argmin_v(cost(h[v]));
    if (crt_val changed) then
        send ok?(⟨x_i, crt_val⟩,
                ca2vn(sum-inference(lvn[*][k],ca[crt_val][k]))
            to each A_k in outgoing_links;
procedure integrate(⟨x_j, v_j⟩) do
    discard CAs and nogoods in ca, lvn, and lr that are
    based on other values for x_j;
    store ⟨x_j, v_j⟩ in agent view;
```

Algorithm 1: Procedures of A_i in ADOPT-ng

The lemma results by induction for an increasingly growing suffix of the list of agents (in the order used by the algorithm): It is trivial for the last agent.

Assuming that it is true for the agent A_k, it follows that it is also true for agent A_{k-1} since adding A_{k-1}'s local cost to the cost received from A_k will be higher (or equal when removing zero costs) to the result of adding A_{k-1}'s local cost to one from any successor of A_k. Respecting the order in Remark 6 guarantees this value is obtained. Therefore the sum between the local cost and the last valued nogood coming from A_k defines the last valued nogood sent by A_{k-1}. □

THEOREM 4. *ADOPT-ng returns an optimal solution.*

PROOF. We prove by induction on an increasingly growing suffix of the list of agents that this suffix converges to a solution that is optimal for their subproblem.

The induction step is immediate for the suffix composed of the agent A_n alone. Assume now that it is true for the suffix starting with A_k. Following the previous two lemmas, one can conclude that at quiescence A_{k-1} knows exactly the cumulated cost of the problems of its successors for its chosen assignment, and therefore knows that this cumulated cost cannot be better for any of its other values.

Since A_{k-1} has selected the value leading to the best sum of costs (between his own local cost and the costs of all subsequent agents), it follows that the suffix of agents starting with A_{k-1} converged to an optimal solution for their subproblem. □

The space complexity is basically the same as for ADOPT. The SRCs do not change the space complexity of the valued nogood.

6.3 Optimizing valued nogoods

Both for the version of ADOPT-ng using DFS trees, as well as for the version that does not use such trees preprocessing, if valued nogoods are used for managing cost inferences, then a lot of effort can be saved at context switching by not discarding nogoods that remain valid [6]. The amount of saved effort is higher if the nogoods are carefully selected (to minimize their dependence on changes in often switched low priority variables). Computing valued nogoods by minimizing the index of the least priority variable involved in the context is shown by our experiments to perform well in this case. This is done by computing the valued nogoods using incrementally lower priority variables, and keeping the valued nogoods with lower priority agents only if they have better thresholds. Nogoods optimized in similar manner were used in several previous DisCSP techniques [2, 19].

6.4 Exploiting DFS trees

Note that while our versions of ADOPT work better than the original DFS-tree based version, they can also create hybrids by using an existing DFS tree. We have identified two ways of exploiting such an existing structure. The first way is by having each agent send its valued nogood only to its parent in the tree (less efficient in length of longest causal chain of messages but more efficient in number of total messages), roughly equivalent to the original ADOPT. The other way is by sending valued nogood to all the ancestors. This later hybrid can be seen as a certain fulfillment of a direction of research suggested in [9], namely communication of costs to higher priority parents.

An extension proposed to this work consists in integrating consistency maintenance in nogood-based optimization. This can be done with the introduction of *valued consistency nogoods*, as described in [13, 14].

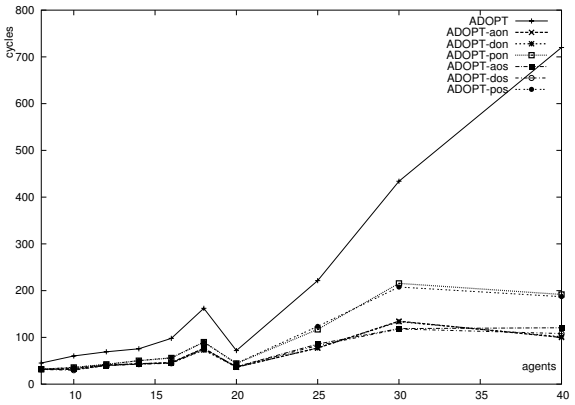


Figure 4: Longest causal chain of messages (cycles) used to solve versions of ADOPT using CAs, on problems with density .2.

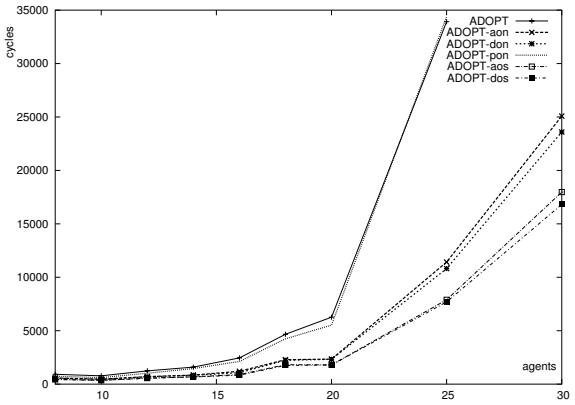


Figure 5: Longest causal chain of messages (cycles) used to solve versions of ADOPT using CAs, on problems with density .3.

7. EXPERIMENTS

We implemented several versions of ADOPT-ng, differing by how the agents picks the targets of his nogoods. In the implementation ADOPT-pon, valued global nogoods are sent only to the parent of the current agent in the DFS tree. In ADOPT-don, the valued global nogoods are sent to all the ancestors of the current agent in the DFS tree. ADOPT-aon is a version where the DFS tree is reduced to the linear list of agents (each having the predecessor as parent). ADOPT-pos, ADOPT-dos, and ADOPT-aos are the corresponding versions with valued nogoods rather than valued global nogoods. These are early versions that do not yet exploit threshold nogoods in **ok?** messages. Improvements from those techniques will be reported elsewhere.

The algorithms were compared on the set of problems posted together with ADOPT, which are the same problems that are used to report ADOPT's performance in [9]. To correctly compare our techniques with the original ADOPT we have used the same order (or DFS trees) on agents for each problem. The set of problems distributed with ADOPT and used here contain 25 problems for each problem size. It contains problems with 8, 10, 12, 14, 16, 18, 20, 25, 30, and 40 agents, and for each of these numbers of agents it contains test sets with density .2 and with density .3. The density of a (binary) constraint problem's graph with n variables is defined by the ratio between the number of binary constraints and $\frac{n(n-1)}{2}$. Results are averaged on the 25 problems with the same parameters.

The number of cycles, i.e., longest causal (sequential) chain of messages, for problems with density .2 is given in Figure 4. Results for problems with density .3 are given in Figure 5. The original ADOPT for 20 and 25 agents and density .3 required more than 2 weeks for solving one of the problems, and it was therefore evaluated using only the remaining 24 problems at those problem sizes.

We can note that the use of valued nogoods brought up to 10 times improvement on problems of density 0.2, and up to 5 times improvement on the problems of density .3.

Another interesting remark is that sending nogoods only to the parent node is significantly worse (in number of cycles), than sending nogoods to all ancestors. Versions using DFS trees require less parallel/total messages, being more network friendly, as seen in Figure 6.

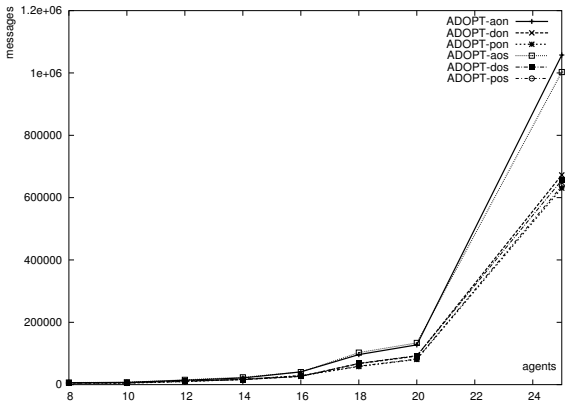


Figure 6: Total number of messages used by versions of ADOPT-ng using CAs to solve problems with density .3.

Figure 5 shows that, with respect to the number of cycles, the use of SRCs practically replaces the need of precomputing a DFS tree since ADOPT-aos is one of the best solvers, only slightly worse than ADOPT-dos. SRCs bring improvements over versions with valued global nogoods, since SRCs allow detection of dynamically obtained independences.

We do not perform any runtime comparison since our versions of ADOPT are implemented in C++, while the original ADOPT is in Java (which obviously leads to all our versions being an irrelevant order of magnitude faster).

It is visible from Figure 5 that the highest improvement in number of cycles is brought by sending valued nogoods to other ancestors besides the parent. The next factor for improvement with difficult problems (density .3) was the use of SRCs. The use of the structures of the DFS tree

bring slight improvements in number of cycles (when nogoods reach all ancestors) and large improvements in total message exchange.

Experimental comparison with DPOP is redundant since its performance can be easily predicted. DPOP is a good choice if the induced width γ of the graph of the problem is smaller than $\log_d T/n$ and smaller than $\log_d S$ where T is the available time, n the number of variables, d the domain size, and S the available memory of a computer.

8. CONCLUSIONS

The ADOPT distributed constraint optimization algorithm can be used efficiently (in number of cycles) without knowing a DFS tree, typically computed in a preprocessing step, by using valued nogoods tagged with sets of references to culprit constraints. The generalized algorithm is denoted ADOPT-ng. SRCs allow detection and exploitation of dynamically created independences between subproblems (that are due to some assignments). Experimentation shows that it is important to send nogoods to several higher priority agents. DFS trees can still be used in conjunction with the value nogood paradigm for optimization improving the total number of messages. ADOPT-ng versions exploiting DFS trees that we tested so far are also slightly better (in number of cycles) than the ones without DFS trees.

Besides that elegance brought by valued nogoods to the description and implementation of ADOPT in ADOPT-ng, use of SRCs to dynamically detect and exploit independences, as well as generalized communication of nogoods to several ancestors, brings experimental improvements of an order of magnitude.

9. REFERENCES

[1] S. Ali, S. Koenig, and M. Tambe. Preprocessing techniques for accelerating the DCOP algorithm ADOPT. In *AAMAS*, 2005.

[2] C. Bessiere, I. Brito, A. Maestre, and P. Meseguer. Asynchronous backtracking without adding links: A new member in the abt family. *Artificial Intelligence*, 161:7–24, 2005.

[3] Z. Collin, R. Dechter, and S. Katz. Self-stabilizing distributed constraint satisfaction. *Chicago Journal of Theoretical Computer Science*, 2000.

[4] J. Davin and P. J. Modi. Impact of problem centralization in distributed cops. In *DCR*, 2005.

[5] R. Dechter. *Constraint Processing*. Morgan Kaufman, 2003.

[6] M. L. Ginsberg. Dynamic backtracking. *Journal of AI Research*, 1, 1993.

[7] K. Hirayama and M. Yokoo. Distributed partial constraint satisfaction problem. In *Proceedings of the Conference on Constraint Processing (CP-97),LNCS 1330*, pages 222–236, 1997.

[8] R. Maheswaran, M. Tambe, E. Bowring, J. Pearce, and P. Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed event scheduling. In *AAMAS*, 2004.

[9] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *AIJ*, 161, 2005.

[10] P. J. Modi, M. Tambe, W.-M. Shen, and M. Yokoo. A general-purpose asynchronous algorithm for distributed constraint optimization. In *Distributed Constraint Reasoning, Proc. of the AAMAS'02 Workshop*, Bologna, July 2002. AAMAS.

[11] A. Petcu and B. Faltings. Approximations in distributed optimization. In *Principles and Practice of Constraint Programming CP 2005*, 2005.

[12] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, 2005.

[13] M.-C. Silaghi. *Asynchronously Solving Distributed Problems with Privacy Requirements*. PhD Thesis 2601, (EPFL), June 27, 2002. `http://www.cs.fit.edu/~msilaghi/teza`.

[14] M.-C. Silaghi. Howto: Asynchronous PFC-MRDAC –optimization in distributed constraint problems +/-ADOPT–. In *IAT*, Halifax, 2003.

[15] M.-C. Silaghi and B. Faltings. A comparison of DisCSP algorithms with respect to privacy. In *AAMAS-DCR*, 2002.

[16] M.-C. Silaghi, J. Landwehr, and J. B. Larrosa. volume 112 of *Frontiers in Artificial Intelligence and Applications*, chapter Asynchronous Branch & Bound and A* for DisWCSPs with heuristic function based on Consistency-Maintenance. IOS Press, 2004.

[17] M.-C. Silaghi and D. Mitra. Distributed constraint satisfaction and optimization with privacy enforcement. In *3rd IC on Intelligent Agent Technology*, pages 531–535, 2004.

[18] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Consistency maintenance for ABT. In *Proc. of CP'2001*, pages 271–285, Paphos,Cyprus, 2001.

[19] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Hybridizing ABT and AWC into a polynomial space, complete protocol with reordering. Technical Report #01/364, EPFL, May 2001.

[20] R. M. Stallman and G. J. Sussman. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9:135–193, 1977.

[21] R. Wallace and M.-C. Silaghi. Using privacy loss to guide decisions in distributed CSP search. In *FLAIRS'04*, 2004.

[22] M. Yokoo. Constraint relaxation in distributed constraint satisfaction problem. In *ICDCS'93*, pages 56–63, June 1993.

[23] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. Distributed constraint satisfaction for formalizing distributed problem solving. In *ICDCS*, pages 614–621, June 1992.

[24] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE TKDE*, 10(5):673–685, 1998.

[25] M. Yokoo and K. Hirayama. Distributed constraint satisfaction algorithm for complex local problems. In *Proceedings of 3rd ICMAS'98*, pages 372–379, 1998.

[26] M. Yokoo, K. Suzuki, and K. Hirayama. Secure distributed constraint satisfaction: Reaching agreement without revealing private information. In *CP*, 2002.

[27] W. Zhang and L. Wittenburg. Distributed breakout revisited. In *Proc. of AAAI*, Edmonton, July 2002.

MDPOP: Faithful Distributed Implementation of Efficient Social Choice Problems

Adrian Petcu [*]
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
adrian.petcu@epfl.ch

Boi Faltings
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
boi.faltings@epfl.ch

David C. Parkes [†]
DEAS, Harvard University
Cambridge, MA 02138 USA
parkes@eecs.harvard.edu

ABSTRACT

We model social choice problems in which self interested agents with private utility functions have to agree on values for a set of variables subject to side constraints. The goal is to implement the efficient solution, maximizing the total utility across all agents. Existing techniques for this problem fall into two groups. Distributed constraint optimization algorithms can find the solution without any central authority but are vulnerable to manipulation. Incentive compatible mechanisms can ensure that agents report truthful information about their utilities and prevent manipulation of the outcome but require centralized computation.

Following the agenda of *distributed implementation* [16], we integrate these methods and introduce *MDPOP*, the first distributed optimization protocol that *faithfully* implements the VCG mechanism for this problem of efficient social choice. No agent can benefit by unilaterally deviating from any aspect of the protocol, neither information-revelation, computation, nor communication. The only central authority required is a bank that can extract payments from agents. In addition, we exploit structure in the problem and develop a faithful method to redistribute some of the VCG payments back to agents. Agents need only communicate with other agents that have an interest in the same variable, and provided that the distributed optimization itself scales the entire method scales to problems of unbounded size.

1. INTRODUCTION

Distributed optimization problems can model environments where a set of agents must agree on a set of decisions subject to side constraints. We consider settings in which each agent has its own preferences on subsets of these decisions, expressed as relations that define its utility. The agents are self interested, and each one would like to obtain the decision that maximizes its own utility. However, the system as whole agrees (or some social designer determines) that a solution should be selected to maximize the total utility across all agents. Thus, this is a problem of *efficient social choice*. As motivation, we have in mind problems such as meeting scheduling, where the decisions are about when and where to hold each meeting, or scheduling contractors in construction projects.

Agents can of course solve such problems using a central authority that computes the optimal solution. In combination with a mechanism such as the Vickrey-Clarke-Groves (VCG) mechanism, we can also prevent manipulation by agents. However, in many practical settings it is hard to bound the problem so that such a central authority is feasible. Consider meeting scheduling: while each agent only participates in a few meetings, it is in general not possible to find a set of meetings that has no further constraints with any other meetings and thus can be optimized separately. Similarly, contractors in a construction project simultaneously work on other projects, again creating an unbounded web of dependencies that cannot be optimized in a centralized fashion.

Algorithms for distributed constraint reasoning such as ABT and AWC ([21]), AAS [20], DPOP [17] and ADOPT [14] can deal with problems of unbounded size as long as the influence of each agent on the solution is limited to a bounded number of variables. However, the current techniques do not address the problem of making agents truthfully declare their preferences and execute the protocol correctly.

In this paper, we advance the agenda of *distributed implementation* [16], which integrates methods from mechanism design with methods from distributed constraint optimization. In distributing the centralized computation of mechanism design across a system of self-interested agents the key challenge is to ensure that agents cannot gain from deviating from the distributed protocol. In addition to information revelation, agents will now be asked to participate in computation and message passing, both of which can provide new opportunities for manipulation. We describe the first faithful distributed constraint optimization algorithm, implementing the VCG outcome without any trusted third party besides a bank, used to enforce payments. The protocol forms an *ex post* Nash equilibrium [12], so that no agent can benefit by unilaterally deviating, whatever the utility functions of other agents and whatever the constraints. While noting that our protocol *never* runs at a deficit, we also demonstrate how to exploit problem structure in facilitating payment distribution *back* to agents from the bank. To do this we identify components of the problem that define payments that cannot be influenced by some subset of agents, that are then eligible to receive a share of the payments.

After preliminaries, in Section 3 we describe the DPOP [17] algorithm for distributed constraint optimization, which is the focus of our study. Section 4 extends DPOP to compute the VCG outcome and proves that the extended protocol, called MDPOP, is

[*]The first author is a PhD student at EPFL/Lausanne, supported by the Swiss National Science Foundation grant 200020-103421/1.

[†]Parkes is supported by National Science Foundation grants IIS-0238147, IIS-0534620 and an Alfred P. Sloan Foundation award.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

faithful. We also provide an accelerated version of MDPOP that simultaneously computes the solution to the marginal and main economies, and again establish faithfulness. Section 5 discusses the issue of budget balance and defines our payment redistribution method.

2. PRELIMINARIES

We assume that the social choice problem consists of a finite but possibly unbounded number of decisions that all have to be made at the same time. Each decision is modeled as a variable that can take values in a well-defined domain. There can be side constraints between the variables, and each agent can also have private *relations* that define its utility for decisions.

Modeled as a distributed constraint optimization problem, DCOP($\mathcal{A}$) on agents $\mathcal{A}$ we have:

DEFINITION 1. *An efficient social choice problem is modeled as a distributed constraint optimization problem (DCOP) as a tuple* $< \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{R} >$ *such that:*

$\mathcal{A} = \{A_1, ..., A_n\}$ *is a set of* **self-interested** *agents interested in the optimization problem;*

$\mathcal{X} = \{X_1, ..., X_m\}$ *is the set of* **public** *decision variables;* $P(A_i) \subseteq \mathcal{X}$ *is the sub-domain of variables on which agent* A_i ***could*** *have relations;* $X(A_i) \subseteq P(A_i)$ *are the variables in which agent* A_i *is interested and does have relations;*

$\mathcal{D} = \{d_1, ..., d_m\}$ *is the set of finite* **public** *domains of the variables* $\mathcal{X}$*; each domain is known to all interested agents;*

$\mathcal{C} = \{c_1, ..., c_q\}$ *is a set of* **public** *constraints, where a constraint* c_i *is a function* $c_i : d_{i_1} \times .. \times d_{i_k} \rightarrow \{-\infty, 0\}$ *that returns 0 for all allowed combinations of values of the involved variables, and* $-\infty$ *for disallowed ones; these constraints are known and agreed upon by all agents involved in the respective communities;*

$\mathcal{R} = \{R_1, ..., R_n\}$ *is a set of* **private** *relations, where* R_i *is the set of relations specified by agent* A_i *and relation* $r_i^j \in R_i$ *is a function* $d_{j_1} \times .. \times d_{j_k} \rightarrow \mathbb{R}$ *specified by agent* A_i*, which denotes the utility* A_i *receives for all possible values on the involved variables* $\{j_1, \ldots, j_k\}$ *(negative values can be thought of as costs).*

The optimal solution is a complete instantiation X^* *of all variables in* $\mathcal{X}$*, s.t.* $X^* = argmax_{X \in \mathcal{D}}(\sum_{R_i \in \mathcal{R}} R_i(X) + \sum_{c_i \in \mathcal{C}} c_i(X))$[1]*, where* $R_i(X) = \sum_{r_i^j \in R_i} r_i^j(X)$ *is* A_i*'s utility for this solution.*

Refer to the agents A_i for which $X_j \in X(A_i)$ for some variable X_j as forming the *community* for variable X_j. We will use DCOP($-A_i$) to denote the constraint optimization problem without agent A_i, and refer to this as the "marginal problem without agent A_i."

An agent can also have *private variables*, and relations/constraints imposed on subsets of private variables and public variables. Decisions about private variables, as well as explicit information about these relations and constraints will remain private to an agent.

In addition to defining values for variables, our faithful protocols will also define payments, to be collected (or made) to agents. The only central authority that we require is a *bank* that can enforce these payments. Agents are modeled with quasilinear utility functions, so that agent i's total utility for decision X and payment p made to a bank is $R_i(X) - p$.

The main assumptions made for this paper are as follows:

- The set of variables $\mathcal{X}$, i.e. the number of decisions, is fixed and independent of the participating agents. Moreover, each agent knows the variables that it is interested in.
- Domains $\mathcal{D}$ are known to all interested agents.
- Each constraint $c_i \in \mathcal{C}$ is known to all agents interested in any variable involved in c_i.
- The agents with possible and actual interest in a variable X_i are known to all agents in the community of X_i.
- An agent can communicate with all agents in all communities in which it is a member.
- Agents are modeled as *rational but helpful*, meaning that although self-interested, they will follow a protocol whenever there is no deviation that will make them *strictly* better off.
- No collusion between agents.
- The problem has a feasible solution.
- Catastrophic failure if all agents in a community do not eventually agree on the same value for the variable.
- Every agent has a trusted communication channel with the bank.

To motivate the assumption that all members of a community are known to each other, consider meeting scheduling in which the decision variables are the times and locations of each meeting. Here, we would require that for each meeting there will be a list of participants that have to agree on the time and place. Realize that the only communication that we assume (other than with the bank) is among agents in the same community.

The assumption of catastrophic failure given disagreement is only used to ensure that once the multi-agent system has come to a decision it will be finally *executed.* It is to prevent "hold-out" by an unhappy agent at this final stage. Given that the other agents set their local values to be the agreed upon solution, no agent can benefit by adopting an alternative view of the decision. To motivate this, realize that a scheduled meeting where some participants assumed a different time than others would not be valid, benefitting no one. [2]

A simple "centralized" model of the DCOP($\mathcal{A}$), which we write COP($\mathcal{A}$), can be represented as a *multigraph* (for example Figure 1(a)), with the decision variables as nodes, and (possibly) multiple relations belonging to different agents that involve the same variables. Our complexity results are stated in terms of the induced width of this graph ([3]).

In order to allow multiple agents to express preferences on the same set of variables and *in a distributed fashion*, we adopt a distributed model where each agent has a local replica of the variables that it is interested in (e.g. Figure 1(b).) For each public variable, $X_j \in X(A_i)$, agent A_i has a local copy of X_j, denoted X_j^i. Agent A_i then models its interests as a local problem $COP(X(A_i), R_i)$, by specifying its relations $r_i^j \in R_i$ on the locally replicated variables $X(A_i)$. All copies of the same variable are synchronized between agents through equality constraints. In solving the problem, agents interact with others only through the equality constraints between local replicas of the public variables. [3]

[1]Notice that the second sum is either $-\infty$ if X is an infeasible assignment, or 0 if it is feasible. Thus, optimal solution X^* will always satisfy all hard constraints when that is possible.

[2]On the other hand, this is not an appropriate assumption in market domains where the decision is a trade of goods: it may in fact *not* be catastrophic for a seller to finally renege on an agreed trade. Here, we would need additional techniques such as monitoring to extend out methods. See Shneidman and Parkes [19] for an extended discussion of the problem of final execution of an agreement.

[3]Local, private variables do not show up in inter-agent communication. Agents typically need not solve the internal problem for all

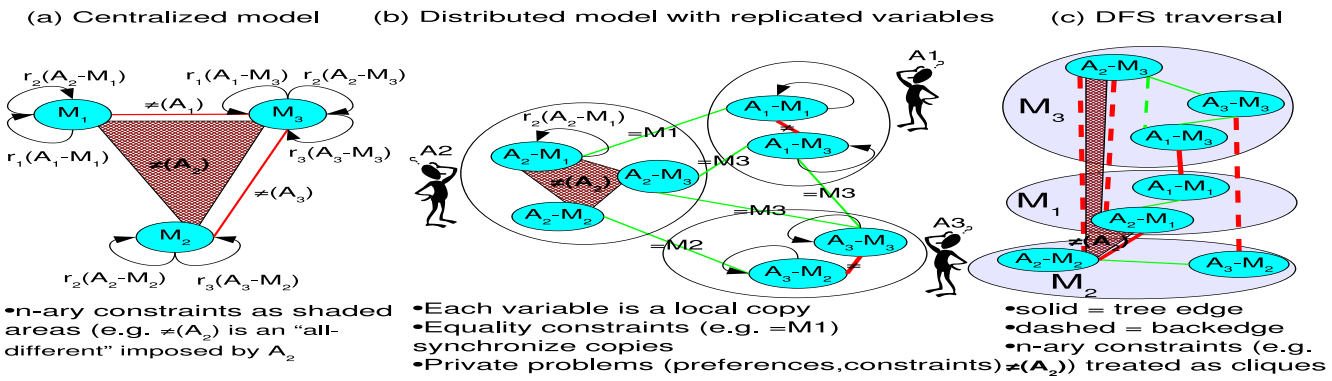


Figure 1: *A meeting scheduling problem, its modeling as a DCOP with replicated variables, and a DFS arrangement*

Example: Meeting Scheduling. This model can be instantiated for distributed meeting scheduling, yielding the PEAV model [13]. Figure 1 shows an example where 3 agents want to find the optimal schedule for 3 meetings. Each agent has as variables the starting times of the meetings it participates in (e.g. A_2-M_1 represents the local copy of the variable representing meeting M_1 for agent A_2). Local *all-different* constraints between an agent's variables ensure that it does not participate in several meetings at the same time. Inter-agent equality constraints between local copies corresponding to the same meeting model the requirement of global agreement. Unary relations on the starting times of the meetings (e.g. $r_2(A_2\text{-}M_1)$) model the preferences of the agents.

2.1 The Centralized VCG Mechanism

The Vickrey-Clarke-Groves (VCG) mechanism (see Jackson [11]) provides a centralized and incentive-compatible (IC) solution to efficient social-choice problems. Indeed, the Groves family of mechanisms (of which VCG is an instance) are the only efficient, IC social choice mechanisms [9]. There is a long tradition of leveraging the VCG mechanism within DAI, going back to Ephrati and Rosenschein [4] who considered the use of VCG mechanisms to achieve consensus.

To use the centralized VCG, each agent would report to a center its relations (and also the domains of variables, and constraints if they were not already known by the center). The center would assign values to variables and determine payments to be made by each agent. The VCG mechanism enjoys a strong form of IC: it is *truthful*, meaning that each agent can always maximize its own utility by reporting true information about its relations, *whatever* the reports of other agents. Truthful reporting is a *dominant-strategy equilibrium* (DSE), which is useful because it frees an agent from modeling the behavior of other agents in computing its equilibrium strategy. Each agent makes a payment equal to the marginal impact of its presence on the rest of the system. In determining this, the center in the VCG mechanism would also solve the marginal problems DCOP($-A_i$) without each agent A_i. Let X^*_{-i} denote the solution to DCOP($-A_i$) and X^* denote the solution to DCOP($\mathcal{A}$). The payment by agent i to the center is:

$$Tax(A_i) = \sum_{j \neq i} \left(R_j(X^*_{-i}) - R_j(X^*) \right), \qquad (1)$$

where $R_j \in \mathcal{R}$ are the relations specified by agent A_j. Thus, agent A_i makes a payment equal to the total marginal negative effect of its presence on the utility of other agents.

combinations of values of the public variables [21].

Realize that in all problem instances we have $Tax(A_i) \geq 0$, for all A_i, with $\sum_{j \neq i} R_j(X^*_{-i}) \geq \sum_{j \neq i} R_j(X^*)$ because agent A_i's presence can only have the effect of changing the values of variables away from the best possible settings just for agents $\neq A_i$. Thus, we always have *weak* budget-balance, with the center running a surplus in all instances of our social choice problem.

The payment by agent A_i can be disaggregated, with $Tax_j(A_i) = R_j(X^*_{-i}) - R_j(X^*)$ denoting the payment made by agent A_i based on its marginal effect on agent A_j. Indeed, it can be further disaggregated to the individual relations of agent A_j. This simple observation will be very powerful in our setting.[4] It will permit a distributed computation of tax payments, where agent A_j computes the payment that should be made by other agents to the bank for their marginal effect on itself; i.e. agent A_j computes $Tax_j(A_i)$ for all $i \neq j$, which will be possible because A_j will know the values on variables of interest in X^* and X^*_{-i}.

2.2 Distributed Implementation

Parkes and Shneidman [16, 19] introduce the notion of *distributed implementation* for social choice problems. A distributed implementation (DI), $d_M = < g, \Sigma, s^m >$, defines an outcome rule $g : \Sigma^n \rightarrow \mathcal{D} \times \mathbb{R}^n$, where $g_1 \in \mathcal{D}$ defines values on variables and $g_2 \in \mathbb{R}^n$ defines the payment by each agent, a feasible strategy space Σ, and a *suggested* (multi-agent) protocol s^m. A protocol d_M is ex post *faithful* if suggested protocol s^m is an *ex post* Nash equilibrium (NE), meaning that no agent can benefit by deviating from the protocol in equilibrium (i.e. given that other agents follow the protocol) and whatever the particular instance of DCOP.

In a DI, the suggested protocol, s^m, combines the information revelation actions of mechanism design with the computational and communication actions of distributed algorithms. Thus, in following s^m an agent is both revealing information about its private relations and assisting in solving the DCOP and computing payments. The outcome rule g defines the assignment of values and payments for all possible termination states, including those that could arise from unilateral deviations. The feasible strategy space Σ, restricts the space of actions available to an agent in all possible states of the protocol, i.e. the messages that an agent can send that are interpretable by the other agents given that they are following the suggested protocol.

Parkes and Shneidman [16] introduced the **partition principle** for the distributed implementation of VCG mechanisms. Briefly, this principle states that a distributed mechanism is an *ex post*

[4] See Feigenbaum et al. [6] for a corresponding disaggregated VCG payment in the domain of shortest-path Internet routing.

faithful distributed implementation of efficient social choice if: (1) optimal solutions are always obtained for $DCOP(\mathcal{A})$ and $DCOP(-A_i)$ given s^m; (2) agent A_i cannot influence either the solution to $DCOP(-A_i)$ or its tax; (3) the optimal solution of $DCOP(\mathcal{A})$ is correctly executed and the corresponding taxes collected.

Loosely, the partition principle holds because no agent can affect its payment for any outcome. Thus, it is in the best interest of every agent to follow the suggested protocol so that the efficient outcome (i.e. the outcome selected in the VCG mechanism) is selected. The only effect of a deviation by an agent is to change either the outcome or some other agent's payment. Truthfulness of VCG then gives faithfulness. The suggested strategy forms an *ex post* NE but not a DSE because it relies on other agents following the strategy; if another agent deviates, e.g. from its role in the computation to solve $DCOP(\mathcal{A})$, then the correct outcome of the VCG mechanism will not be selected.

In related work, Feigenbaum and colleagues [6, 7] introduced the notion of *distributed algorithmic mechanism design*, but emphasized complexity questions rather than the faithfulness that is central to DI; see [19] for a faithful extension. Monderer and Tennenholtz [15] consider a distributed single item auction problem, but focus on communication of messages by self-interested agents rather than distributed computation. Finally, Izmalkov et al. [10] leverage cryptographic methods to convert centralized mechanisms into DIs on fully connected graphs.

3. DISTRIBUTED OPTIMIZATION VIA DPOP

In this section, we instantiate the DPOP algorithm [17] for efficient social choice. DPOP is an instance of Dechter's general bucket elimination scheme [3], adapted for the distributed case. This instantiation runs in three phases, which are very similar to the ones from the standard DPOP protocol. Phase one (section 3.1) constructs $DFS(\mathcal{A})$, which defines the control ordering of the inference algorithm. Phase two is a bottom-up utility propagation, and phase three is a top-down value assignment propagation (see section 3.2). There are some slight differences in phases one and two because we seek to exploit the structure of this DCOP model with replicated variables, for computational efficiency reasons.

Notice that DPOP can be applied to disconnected problems as well: the DFS arrangement is then a DFS forest, and agents in each connected component simply execute DPOP on a tree of that forest. The solution to the (disconnected) problem is then simply the union of optimal solutions for each independent, connected subproblem.

Section 4 will modify DPOP to make it faithful.

3.1 Phase One: DFS Tree Generation

See Algorithm 1. This phase has as a goal to generate a depth-first traversal (DFS) of the problem graph in a distributed manner. A DFS arrangement of a graph G is a rooted tree with the same nodes and edges as G and the property that adjacent nodes from the original graph fall in the same branch of the tree (thus, there are no edges between different branches of the tree). Common definitions of parent, child, pseudoparent, pseudoparent apply. For example, in Figure 1(c), A_2-M_1 and A_2-M_2 are parent/child to each other, and A_2-M_3 and A_2-M_2 are pseudoparent/pseudochild. *Tree-edges* connect parents/children (e.g. A_2-$M_1 - A_2$-M_2), and *back-edges* connect pseudoparents/pseudochildren (e.g. A_2-$M_3 - A_2$-M_2).

First, each agent A_i formulates internally its interests on the variables $X(A_i)$ as $COP(X(A_i), R_i)$, with a replicated variable X_j^i for each $X_j \in X(A_i)$. All agents subscribe to the communities they are interested in, and learn which other agents belong to these communities. [5] In doing this the *problem graph* is constructed. Next, one of the variables, X_0 is chosen as the DFS root. [6] The agents involved in the community for X_0 then randomly choose one of them, A_r as the *leader*. The local copy X_r^0 of variable X_0 forms the root of the DFS. In the case that the problem is initially disconnected then a modification is required to choose multiple root communities, one for each connected component.

Algorithm 1: *Phase One of DPOP.*

DPOP($\mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{R}$):

1 Each agent A_i models its interests as $COP_i(X(A_i), R_i)$: a set of relations R_i imposed on a set $X(A_i)$ of variables X_j^i that each replicate each public variable $X_j \in X(A_i)$

2 Each agent A_i subscribes to the communities of $X_i \in X(A_i)$

DFS Generation:

3 The agents $\mathcal{A}$ choose one of the variables, X_0, as the root.

4 Agents in X_0's community elect a "leader", A_r

5 A_r initiates the token passing to construct the DFS

6 At completion, each A_i knows $P(X_j^i)$, $PP(X_j^i)$, $C(X_j^i)$, $PC(X_j^i)$, for all local copies X_j^i.

Second, the agents participate in a distributed depth-first traversal of the problem graph to construct the DFS for problem DCOP($\mathcal{A}$), which we denote $DFS(\mathcal{A})$. (Multiple DFS trees are generated for disconnected problems.) For convenience, we describe the DFS process as a token-passing algorithm in which all members within a community can observe the release or pick up of the token by the other agents. But, this can also be implemented via (private) message passing.

Let us refer to the example from Figure 1, and assume that M_3 was chosen as the start community, and A_2 was chosen within the community as the start node. A_2 creates an empty token, adds A_2-M_3's ID to the token, and then releases it back to the community. Another agent from M_3's community (e.g. A_3) picks up the token, adds its copy of M_3 to the token (A_3-M_3's ID) and releases it again. A_1 picks it up and automatically adds equality constraints between its variable A_1-M_3 and all its corresponding replica variables that precede it in the context of the token (A_2-M_3 and A_3-M_3) (i.e. one tree edge and one back edge.) Notice that the result is that all replicas of a variable are arranged in a chain, and have equality constraints (back-edges) with all the predecessors that are replicas of the same variable.

Agent A_1 also adds its copy of M_3 to the token (A_1-M_3) and as the last agent in community M_3 to receive the token looks to see if it is a member of another community that has yet to receive the token (choosing one at random if such a community exists). Here, agent A_1 is linked to community M_1 and adds its copy of M_1 to the token (i.e. A_1-M_1), and then releases the token in M_1's community, where A_2 picks it up. When a dead end is reached, the last agent backtracks by sending the token back to its parent. In our example, this happens when A_3 receives the token from A_2 in the M_2 community. Then, A_3 sends back the token to A_2, etc. Eventually the token returns on the same path all the way to the root, and then the process is complete. [7]

[5] A community can be e.g. a bulletin board, a mailing list, etc

[6] This can be done e.g. randomly, using any distributed algorithm for random number generation, or by simply picking the variable with the highest ID, etc.

[7] K-ary constraints (involving k variables) are treated like a cliques during the DFS construction. Concretely, in Figure 1, there is a ternary *all-diff* constraint $\neq A_2(M_1, M_2, M_3)$. A_2 then considers

In constructing $DFS(\mathcal{A})$, the DFS traversal is made according to the structure defined by the relations of the agents. Most hard constraints appear thus as backedges in such a DFS tree. By convention, any hard constraint $c_i \in \mathcal{C}$ is assigned to the highest agent in the community of the variable involved in c_i that is lowest in the DFS ordering. For example, in Figure 1(c), assume that there is a constraint between M_2 and M_3 that specifies that M_2 should occur after M_3. With our convention, this constraint becomes a backedge between the 2 communities, and is assigned to A_2 for handling, because $A_2\_M_2$ is the highest variable in M_2's community, which is lower than M_3's community in the DFS. A_2 then handles this constraint in parallel with its own relation $A_2\_M_2$-$A_2\_M_3$.

Realize that the choice of DFS does not change the solution, so the choice of root node, leaders, etc does not affect the incentive properties.

3.2 Phases Two and Three: Inference

Phase 2 is a bottom-to-top pass that propagates aggregated information about the relations towards the root. The *UTIL* propagation starts bottom-up from the leaves and propagates upwards only through tree edges, from children to parents. A *UTIL* message sent by X_i to its parent X_j informs X_j how much utility $u^*_{X_i}(v^k_j)$ each one of its values v^k_j gives to the whole subtree rooted at X_i in the optimal solution.

To compute the *UTIL* message for its parent, X_j has to join the messages it received from all its children, and the relations it has with its parent and pseudoparents. [8] Afterwards, it projects itself out of the join and sends the result to its parent. The result of the projection is in fact the set of optimal utilities that can be obtained by the subtree rooted at this node, plus the relations it has with its parent/pseudoparents, for each combination of values of the parent/pseudoparents (see [17] for details and examples). This projection provides for an efficient algorithm.

A useful optimization for social choice problems can be introduced to handle replica variables. In the example of Figure 1, $A_2\_M_2$, $A_3\_M_2$ and $A_2\_M_1$ all have back-edges: $A_2\_M_2 - A_2\_M_3$, $A_3\_M_2 - A_3\_M_3$ and $A_2\_M_1 - A_2\_M_3$ respectively. These represent the inequality constraints for agents. Normal DPOP would condition $UTIL$ messages on both $A_2\_M_3$ and $A_3\_M_3$ separately. For social choice these will adopt the same values due to the equality constraints, and thus the conditioning can be collapsed into a single dimension, the value of M_3. This is possible because all 3 agents involved, i.e. A_1, A_2 and A_3 know that $A_1\_M_3$, $A_2\_M_3$ and $A_3\_M_3$ represent the same variable.

Phase 3 is a top-to-bottom pass that makes decisions about the value of variables, with decisions made recursively from the root down to the leaves. This "$VALUE$ propagation" phase is initiated by the agent A_r representing the root variable X_0, once it has received *UTIL* messages from all of its children. Based on these *UTIL* messages, the root assigns itself the value v^* that maximizes the sum of its own utility and that communicated by all its subtrees. It then sends a *VALUE*$(X^0_r \leftarrow v^*)$ message to every child. The process continues recursively to the leaves, with agents X_i assigning values to local copies of variables.

3.3 Complexity Analysis of DPOP

It has been proved in Petcu and Faltings [17] that *DPOP* produces a linear number of messages for general distributed optimization problems. Its complexity lies in the size of the *UTIL* messages (the *VALUE* messages have linear size). This is also true for its instantiation to social choice problems.

Let us denote by w the width of the problem graph for the centralized model of DCOP($\mathcal{A}$) (e.g. Figure 1(a)). The induced width of a graph is a topological parameter that captures the density and clustering of the graph [3]. It is roughly defined as the *maximal number of overlapping tree paths between any pair of different vertices*. In the example from Figure 1, $w = 2$. Let $D = \max_m |d_m|$ denote the maximal domain of any variable.

THEOREM 1. *The number of messages passed is $2 \times m$, $(n-1)$ and $(n-1)$ for phases one, two and three respectively, where n and m are the number of nodes and edges in the distributed model.*

The maximal amount of computation on any node in DPOP is $O(D^{w+1})$, and the largest UTIL message has $O(D^{w+1})$ entries, where w is the width of the centralized problem graph.

Sketch of Proof. Follows from the analysis of DPOP in Petcu and Faltings [17], and the fact that equivalent variables use up only one dimension in the *UTIL* messages (see Section 3.2), and that a dimension is not projected out immediately as it reaches the first target variable, but only when it reaches its top most copy. □

The complexity of DPOP for social choice problems is exponential in the tree width of the centralized graphical model, but not the decentralized graphical model which includes the replicated variables. This is due to the special handling of replica variables described in Section 3.2.

4. MDPOP: A FAITHFUL PROTOCOL FOR DISTRIBUTED OPTIMIZATION

In this section we extend the DPOP algorithm to define *MDPOP*, and prove that MDPOP is a faithful implementation of distributed constraint optimization, terminating with the outcome of the VCG mechanism. We first provide a simple extension, that we call *simple*-MDPOP, before describing our preferred extension, that we call MDPOP.

Algorithm 2: *Simple-MDPOP.*

1 Run DPOP for $DCOP(\mathcal{A})$ on $DFS(\mathcal{A})$; find X^*
2 **forall** $A_i \in \mathcal{A}$ **do**
3 Run DPOP for $DCOP(-A_i)$ on $DFS(-A_i)$; find X^*_{-i}
4 $\forall A_j \neq A_i$, compute $Tax_j(A_i) = R_j(X^*_{-i}) - R_j(X^*)$
5 $\forall A_j \neq A_i$, report $Tax_j(A_i)$ to the bank
6 Bank deducts $\sum_{j \neq i} Tax_j(A_i)$ from A_i's account
7 A_i implements X^* as solution to its local $COP(A_i)$

Algorithm 2 describes simple-MDPOP. The algorithm is presented for a setting in which the main problem and the subproblems are connected but extends immediately to disconnected problems, as discussed in the previous section and without new incentive considerations. Notice that the protocol sets up, and then solves, $n+1$ DPOP protocols, one for the main problem and one for the n marginal problems, with each agent A_i removed in turn. Once these $n+1$ stages are complete every agent A_j has sufficient local knowledge of the solutions $\{X^*, X^*_{-1}, \ldots, X^*_{-n}\}$ to compute the tax payment that every other agent A_i, for $i \neq j$, should make to the bank because of its marginal effect on agent j. Each agent will finally respect decision X^*, to avoid catastrophic failure.

THEOREM 2. *The simple-MDPOP algorithm is a faithful distributed implementation of the optimal solution to a DCOP, and terminates with the outcome of the VCG mechanism.*

the variables in the scope of this constraint to be a fully connected component, which produces the result from Figure 1(c).

[8] A k-ary relation is introduced in this join only once, by the lowest node in the DFS tree, which is part of its scope. E.g. in Figure 1(c), the constraint $\neq A_2(M_1, M_2, M_3)$ is introduced by $A_2\_M_2$.

PROOF. Follows from the partition principle [16]. First, DPOP computes optimal solutions to DCOP($\mathcal{A}$) and $DCOP(-A_i)$ for all $A_i \in \mathcal{A}$ when every agent follows the protocol. Second, agent A_i cannot influence the solution to $DCOP(-A_i)$ because it is not involved in that computation. The DFS is constructed and then inference performed by the other agents, who completely ignore A_i's variables and constraints, and any messages that agent A_i might send. Moreover, agent A_i is not required to perform any message passing in solving for $DCOP(-A_i)$. Note that any hard constraints that A_i may have handled in $DCOP(\mathcal{A})$ are reassigned automatically to some other agent in $DCOP(-A_i)$.

Notice that $DCOP(-A_i)$ could become disconnected without the presence of agent A_i. However, as noted in the beginning of Section 3, DPOP would still solve $DCOP(-A_i)$ correctly. Finally, agent A_i cannot prevent the correct calculation and reporting of the tax it should pay because this is done by agents $A_j \neq A_i$. The bank collects payments and all agents finally set local copies of variables as in X^* to prevent catastrophic failure. (Notice that agent A_i will not deviate as long as other agents do not deviate. Moreover, if agent A_i is the only agent that is interested in a variable then its value is already optimal for agent A_i anyway.) □

In particular, notice that we get from the partition principle that no agent has an interest in obstructing the choice of root community or leader agent in Phase one of DPOP, or in the information-revelation, computation and message passing in Phases two and three of DPOP. Also, no agent A_i can usefully influence its payment by misreporting the local utility of another agent A_j, as $UTIL$ messages are exchanged. While this could change the select of X^* or X^*_{-k} for some $k \neq \{i, j\}$, it would *not* change the utility information used in finally determining agent A_i's payments because only the utility information local to A_j and known to A_j is used in computing the component of A_i's payment due to its effect on A_j.

Note on antisocial behavior: While it is true that an agent A_j has no immediate self-interest in reporting the payment another agent should make, it does have a long-term self-interest if it wants other agents to be truthful (e.g. imagine a system where over time agents realize that the correct payments are not being collected from others). Reporting exaggerated taxes hurts other agents, but does not increase one's own utility, so this is excluded by our assumption that the agents are self-interested but helpful.

4.1 Full-MDPOP

In *simple-MDPOP*, the computation to solve the main problem is completely isolated from the computation to solve each of the marginal problems. The full *MDPOP* algorithm leverages the computation already performed in solving the main problem in solving the marginal problems whenever this is possible and without breaking incentive properties.

This enables the algorithm to scale well to problems where each agent's influence is limited to a small part of the entire problem. [9]

The first stage of MDPOP solves the main problem just as in Simple-MDPOP, running DPOP($\mathcal{A}$). Once this is complete, each marginal problem is solved in parallel. To solve $DCOP(-A_i)$, a DFS-tree is constructed as a modification to DFS($\mathcal{A}$), retaining as much of the structure as possible. This maximizes the reuse of $UTIL$ messages. The new tree, $DFS(-A_i)$ must be constructed in a way that is non-manipulable, i.e. without allowing agent i to interfere with its construction, and also to ensure correctness. This requires that communities of variables that remain connected in $DCOP(-A_i)$ remain connected in the $DFS(-A_i)$ tree when edges that link to nodes owned by A_i are disabled in solving $DCOP(-A_i)$. For instance, in Figure 2, A_1-M_1−A_1-M_3 is a tree edge in $DFS(\mathcal{A})$, and its removal disconnects $DCOP(-A_1)$).

[9]For example, in a meeting scheduling problem with thousands of agents, any one agent only participates in a few meetings, in a rather restricted circle of acquaintances.

Algorithm 3: *MDPOP.*

1 Run DPOP for $DCOP(\mathcal{A})$ on $DFS(\mathcal{A})$; find X^*
2 **forall** $A_i \in \mathcal{A}$ **do**
3 **Create** $DFS(-A_i)$ **by adjusting** $DFS(\mathcal{A})$**:**
exclude all variables X^i_j and relations that belong to A_i;
the highest descendant of each excluded X^i_j that has a back edge with an ancestor of X^i_j turns it into a tree-edge;
4 **Run DPOP for** $DCOP(-A_i)$ **on** $DFS(-A_i)$:
children/parents of each excluded X^i_j recompute their *UTIL* messages and restart propagations;
reuse *UTIL* msgs from $DPOP(\mathcal{A})$ not influenced by A_i;
5 Compute and levy taxes as in simple-MDPOP;
6 A_i implements X^* as solution to its local $COP(A_i)$;

Phase One of MDPOP for a marginal problem. Consider DPOP($-A_i$). In building $DFS(-A_i)$ from $DFS(\mathcal{A})$, existing links that were back-edges in $DFS(\mathcal{A})$ can be turned into tree-edges in $DFS(-A_i)$ as necessary to keep it connected. This preserves as much as possible of the tree structure. Figure 2 shows an example of a common $DFS(\mathcal{A})$, adjusted for each marginal problem using this idea. For example, A_2-M_3−A_2-M_1 is a back-edge in $DFS(\mathcal{A})$, but becomes a tree-edge in $DFS(-A_1)$ to compensate for the loss of edge A_1-M_1 − A_1-M_3. The algorithm works by considering each of the nodes X_i belonging to A_i in turn. For each X_i that will be excluded from DPOP($-A_i$), all nodes below X_i check the path from the root to themselves, and the list of nodes reachable from their children (both these pieces of information are available after $DFS(\mathcal{A})$ is constructed). The highest node below X_i that has a back edge pointing to a node above X_i converts this edge into a tree edge and converts its pseudo parent into a parent.

Thus, no additional links are created, as we use only existing ones, previously designated as back-edges. Realize that this conversion, in converting back edges to tree edges, cannot increase the induced width of $DFS(-A_i)$ above the one of $DFS(\mathcal{A})$, therefore *UTIL* messages can only decrease in size.

Phase Two of MDPOP for a marginal problem. Each marginal problem is then solved on $DFS(-A_i)$. Notice that the parent and children of excluded nodes will have to recompute their messages from $DPOP(\mathcal{A})$ to account for the new structure and initiate the corresponding propagation for $DPOP(-A_i)$.

Subsequently, any message can be reused iff it comes from a subtree that does not contain any of A_i's variables, because A_i could not have influenced it. E.g. in $DPOP(-A_1)$, A_2-M_1 is a child of $A_2-M_1 \in A_1$. It has to recompute a *UTIL* message and send it to A_2-M_3. To do this, it can reuse the message sent by A_2-M_2 in $DPOP(\mathcal{A})$, because the sending subtree does not contain A_1. By doing so, it reuses the effort spent in $DPOP(\mathcal{A})$ to compute the messages $A_3-M_2 \rightarrow A_2-M_2$ and $A_2-M_2 \rightarrow A_2-M_1$.

THEOREM 3. *The MDPOP algorithm is a faithful distributed implementation of the optimal solution to a DCOP, and terminates with the outcome of the VCG mechanism.*

Sketch of Proof. From the partition principle [16]. First, agent i cannot prevent the construction of a valid DFS for $DCOP(-A_i)$ because in the construction of $DFS(-A_i)$ from the main DFS, all transformations are initiated by neighbors of A_i, and all links with A_i are simply dropped. Second, agent i cannot influence the execution of DPOP on $DCOP(-A_i)$ because all messages that A_i

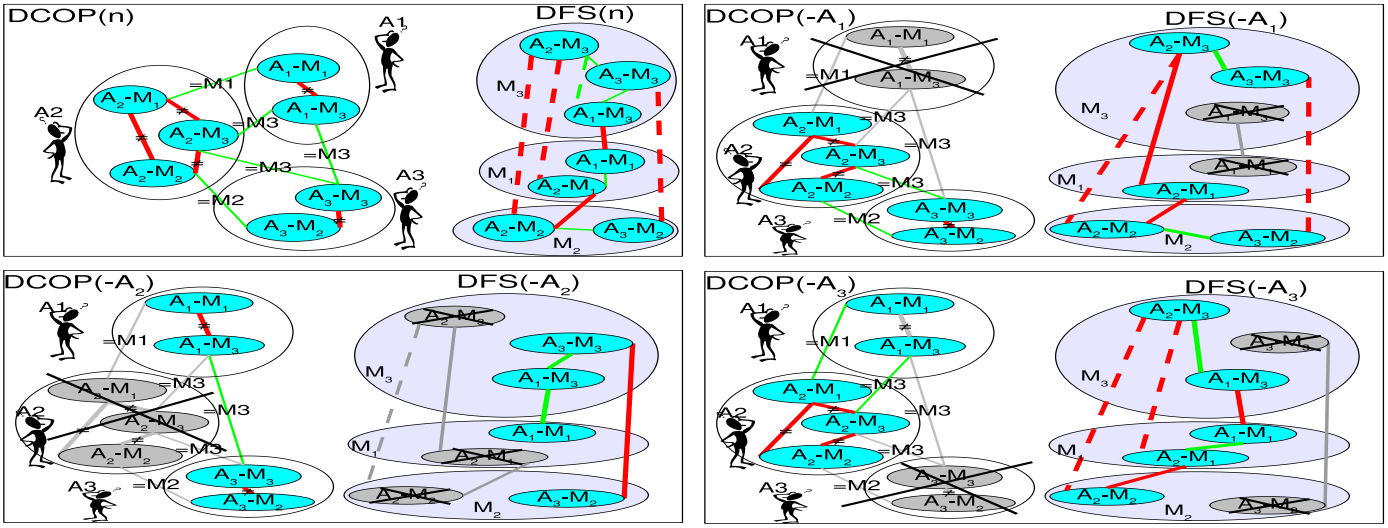


Figure 2: *Each agent A_i is excluded in turn from the optimization $DCOP(-A_i)$. $DFS(-A_i)$ is adapted from $DFS(\mathcal{A})$.*

influenced in the main problem $DCOP(\mathcal{A})$ are recomputed in the new structure. This follows from the fact that every link where agent A_i was responsible for computing a message is eliminated by its neighbors and that all these propagations are restarted. □

5. INCENTIVE COMPATIBLE VCG PAYMENT REDISTRIBUTION

No social choice mechanism can be efficient, incentive-compatible, and individually rational while at the same time guaranteeing exact budget-balance [8]. [10] In our setting, where there are no positive externalities, the VCG mechanism runs at a surplus with the bank receiving a net payment from agents. While these taxes cannot be simply redistributed among the agents, a tax payment *can* be refunded to an agent A_l as long as that agent has no influence on the computation of the payments it receives. This general idea was suggested by Ephrati and Rosenschein ([4]), and recently explored by Faltings [5] and Cavallo [2].

The most straightforward way to implement this idea is to consider any agent that does not influence any of the optimizations that are used in computing a certain part of the VCG tax as eligible to receive this tax. However, this approach would not maintain incentive-compatibility, as an agent with only a small influence on some aspect of the problem could gain an advantage by misstating its preferences to become non-pivotal and thus receive a possibly much larger tax payment.

Faltings [5] suggests to deal with this problem by forcing an agent A_l to be non-pivotal independently of its declarations by simply ignoring it in the optimization. In this way, it is guaranteed that the agent does not have an influence on the tax computation and thus can receive it without creating unwanted incentives. While the mechanism may be forced to choose a suboptimal solution, [5] shows through experiments on randomly generated problems that the expected utility loss from suboptimality is much smaller than what would result from wasting the taxes.

However, a drawback of this approach is that in a large optimization problem, some agent would not be considered at all in the entire problem. It would be more advantageous if various agents could receive *some portion* of the tax in return for *some* reduction of their influence on the solution.

Consider the VCG payment portion:

$$tax_j(A_i) = r_j(X^*_{-i}) - r_j(X^*)$$

that is paid in Algorithm 2 by agent A_i with respect to relation r_j. Let $r_j \in R_k$, i.e. r_j was posted by agent A_k.

We designate an agent A_l, $l \notin \{k, i\}$ to receive this payment as a refund. A straightforward way to choose A_l would be to take an agent that did not influence the values of any of the variables in r_j in the solution. However, this would destroy incentive-compatibility, since an agent may now have an incentive to hide its interest in order to be eligible to receive the refund. Similarly, an agent could have an interest to make other agents look pivotal to increase its own chance of receiving a refund.

To avoid such influence, A_l needs to be chosen independently of the agents' declarations. Our algorithm does this as follows:

1. For each agent A_l, we use the set $P(A_l)$ of the variables on which the agent could possibility express interest and ignore its declarations when they involve other variables.

2. For each payment portion $tax_j(A_i)$, choose an agent A_l that will be eligible to receive it, using any criterion that is not related to the agents' own declarations. This can be done by random selection among agents that cannot possibly be part of the community of the variables.

3. Using the declarations of the agents, for each payment portion verify that the agent A_l chosen to receive it indeed cannot have any influence on the values of the variables involved. If there is no possible influence, the agent receives the payment as a refund, if not, it has to be wasted.

We now give a brief description of the third step of the algorithm. This step is important because even an agent not in the community of a variable may still be able to influence relations via the propagation of its effect over the problem graph.

We use the *omnidirectional* utility propagation from the DPOP extension presented in [18]. In this version, messages circulate in

[10] For budget balance in general problems, one can settle for *ex ante* individual-rationality [1] and Bayes-Nash incentive-compatibility, but this requires the mechanism and agents to have common knowledge about a distribution on agent types.

all directions along the DFS tree (parent to children, too). A message from a parent to its child summarizes the utility information from the entire problem except the subtree of that child. Joining messages from the parent and the children gives each node the same global view of the system as the root in the simple DPOP has.

We can characterize the influence that an agent A_l has on a variable X_k by a *label* that contains a value for each member of X_k's domain d_k. The value is "1" if A_l can make X_k take the corresponding value by its declarations, and "0" if it cannot. If A_l posts a relation on X_k, it can make any value the most preferred one, so each position has a "1".

As an example, consider a variable X_k that can take three values `a, b, c`. Let A_l have no influence on any sibling or ancestor of X_k in the DFS ordering, but complete control of the value of X_k. Thus, A_l's label for X_k is `(1,1,1)`. Let X_a be the ancestor of X_k in the DFS ordering with possible values `d, e, f`, and assume that some other agent has imposed the following relation between X_k and X_a:

$X_a =$	d	e	f
$X_k = a$	3	2	1
$X_k = b$	2	3	1
$X_k = c$	4	3	2

Furthermore, let the sum of all other messages arriving at X_a, assuming omnidirectional propagation, be the vector $(5, 5, 5)$, giving the utilities of $X_a = (\mathtt{d}, \mathtt{e}, \mathtt{f})$ in the rest of the problem. Note that this vector is not influenced by A_l.

Agent A_l can influence the value of X_a only through the utilities it gives to the three different values of X_k. Letting these be $U_l(X_k)$, and factoring the utilities reported in the rest of the problem, the propagation would choose the maximum in each row of the following table, indicated in bold:

X_a=	d	e	f
$X_k = a$	$\mathbf{8 + U_l(X_k = a)}$	$7 + U_l(X_k = a)$	$6 + U_l(X_k = a)$
$X_k = b$	$7 + U_l(X_k = b)$	$\mathbf{8 + U_l(X_k = b)}$	$6 + U_l(X_k = b)$
$X_k = c$	$\mathbf{9 + U_l(X_k = c)}$	$8 + U_l(X_k = c)$	$7 + U_l(X_k = c)$

where the chosen row depends on the value of X_k. Now note that A_l can never force $X_a = \mathtt{f}$, since this will never give the maximum utility. Thus, A_l's label for X_a is `(1,1,0)`. Had A_l's label for X_k been `(1,0,1)`, its label for X_a would have been `(1,0,0)`, meaning that only $X_a = \mathtt{d}$ is possible and thus A_l has in fact no possibility to influence X_a's value.

Note that the number of "1"s in a label can never increase during such propagation, since for every choice of input value there can be only one optimal output value. This means that propagation will eventually converge to labels with a single "1". By propagating labels in the same way as propagating messages in MDPOP, we can determine the variables that an agent can potentially influence.

The presence of backedges in the DFS tree somewhat complicates the algorithm. The full algorithm will be described in a longer version of this paper.

6. CONCLUSIONS

We presented a multiagent constraint optimization algorithm for use in efficient social choice problems when agents are self interested and have private information about their utility for different outcomes. Our algorithm is faithful, in the sense that no agent can improve its utility either by misreporting its local information or deviating from any aspect of the algorithm. The only centralized control we assume is that of a bank that is able to receive messages about payments and collect payments. In addition to promoting efficient decisions we also seek to return payments back to agents, to further improve the net utility of outcomes. Future work should provide a comprehensive empirical analysis, in order to understand the scheme's scalability and budget balance properties on realistic problem instances.

7. REFERENCES

[1] C. d. Aspremont and L. A. Gerard-Varet. Incentives and incomplete information. *Journal of Public Economics*, 11:25–45, 1979.

[2] R. Cavallo. Optimal decision-making with minimal waste: Strategyproof redistribution of VCG payments. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-06)*, Hakodate, Japan, May 2006.

[3] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[4] E. Ephrati and J. Rosenschein. The Clarke tax as a consensus mechanism among automated agents. In *Proceedings of the National Conference on Artificial Intelligence, AAAI-91*, pages 173–178, Anaheim, CA, July 1991.

[5] B. Faltings. A budget-balanced, incentive-compatible scheme for social choice. In *Workshop on Agent-mediated E-commerce (AMEC) VI*. Springer Lecture Notes in Computer Science, 2004.

[6] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. In *Proceedings of the 2002 ACM Symposium on Principles of Distributed Computing*, pages 173–182, 2002.

[7] J. Feigenbaum and S. Shenker. Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13, 2002.

[8] J. Green and J.-J. Laffont. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica*, 45:427–438, 1977.

[9] T. Groves and M. Loeb. Incentives and public inputs. *Journal of Public Economics*, 4:211–226, 1975.

[10] S. Izmalkov, S. Micali, and M. Lepinski. Rational secure computation and ideal mechanism design. In *FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 585–595, Washington, DC, USA, 2005. IEEE Computer Society.

[11] M. O. Jackson. Mechanism theory. In *The Encyclopedia of Life Support Systems*. EOLSS Publishers, 2000.

[12] V. Krishna. *Auction Theory*. Academic Press, 2002.

[13] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham. Taking DCOP to the realworld: Efficient complete solutions for distributed multi-event scheduling. In *AAMAS-04*, 2004.

[14] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *AI Journal*, 161:149–180, 2005.

[15] D. Monderer and M. Tennenholtz. Distributed games: From mechanisms to protocols. In *Proc. 16th National Conference on Artificial Intelligence (AAAI-99)*, pages 32–37, July 1999.

[16] D. C. Parkes and J. Shneidman. Distributed implementations of Vickrey-Clarke-Groves mechanisms. In *Proc. 3rd Int. Joint Conf. on Autonomous Agents and Multi Agent Systems*, pages 261–268, 2004.

[17] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence, IJCAI-05*, Edinburgh, Scotland, Aug 2005.

[18] A. Petcu and B. Faltings. Superstabilizing, fault-containing multiagent combinatorial optimization. In *Proceedings of the National Conference on Artificial Intelligence, AAAI-05*, Pittsburgh, USA, July 2005.

[19] J. Shneidman and D. C. Parkes. Specification faithfulness in networks with rational nodes. In *Proc. 23rd ACM Symp. on Principles of Distributed Computing (PODC'04)*, St. John's, Canada, 2004.

[20] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Asynchronous search with aggregations. In *AAAI/IAAI*, pages 917–922, Austin, Texas, 2000.

[21] M. Yokoo and K. Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, 2000.

Examining DCSP Coordination Tradeoffs

Michael Benisch
School of Computer Science,
Carnegie Mellon University
mbenisch@cs.cmu.edu

Norman Sadeh
School of Computer Science,
Carnegie Mellon University
sadeh@cs.cmu.edu

ABSTRACT

Distributed Constraint Satisfaction Problems (DCSPs) provide a model to capture a broad range of cooperative multi-agent problem solving settings. Researchers have generally proposed two different sets of approaches for solving DCSPs, backtracking based approaches, such as Asynchronous Backtracking (ABT), and mediation based approaches, such as Asynchronous Partial Overlay (APO). These sets of approaches differ in the levels of coordination employed during conflict resolution. While the computational and communication complexity of the backtracking based approaches is well understood, the tradeoffs in complexity involved in moving toward mediation based approaches are not. In this paper we comprehensively reexamine the space of mediation based approaches for DCSP and fill gaps in existing frameworks with new strategies. We present different mediation session selection rules, including a rule that favors smaller mediation sessions, and different mediation strategies, including a decentralized hybrid strategy based on ABT. We present empirical results on solvable 3-coloring and random binary DCSP problems, that accurately capture the computational and communication tradeoffs between ABT and various mediation based approaches. Our results confirm that under some circumstances the newly presented strategies dominate previously proposed techniques.

General Terms

Algorithms, measurement

Keywords

Agent systems, constraint processing, cooperative distributed problem solving, computational complexity

1. INTRODUCTION

Distributed Constraint Satisfaction Problems (DCSPs) provide a formalism for representing problems where different agents are each responsible for instantiating different sets of variables subject to constraints. The agents aim to assign values to their variables, such that all constraints are satisfied. Many real-world scenarios can be modeled as DCSPs, such as supply chain coordination [10], product co-design, and distributed meeting scheduling [8]. Some of these scenarios give rise to naturally distributed problems, others benefit from distributed reasoning for privacy or confidentiality reasons.

There have been several distributed algorithms developed for solving DCSPs, each with particular strengths and weaknesses. The DCSP algorithm that has received the most attention is an asynchronous version of Constraint Backtracking (CBT), called Asynchronous Backtracking (ABT) [12]. The original proposal for ABT has seen several improvements [3, 2, 14], and is considered one of the most effective techniques for solving DCSPs.

Recently the multi-agent systems community has become interested in DCSP algorithms for modeling agent coordination tasks. This idea was introduced by Mailler and Lesser in the context of an algorithm called Asynchronous Partial Overlay (APO) [5]. In APO agents involved in a conflict select a mediator to solve a centralized version of a subproblem capturing key elements of that conflict.

APO has been shown to involve significantly less communication than backtracking based techniques, such as ABT, by reducing the amount of unsuccessful instantiation attempts that repeatedly violate the same constraints (*thrashing*) [5]. On the other hand, in contrast with ABT, current versions of mediation based techniques have failed to fully exploit opportunities for concurrent processing. This tradeoff has not been explicitly identified in prior experimental investigations due to somewhat coarse computational complexity measurements.

In this paper we measure computational complexity based on a finer metric recently proposed by Meisels *et. al.* [6], that involves counting non-concurrent constraint checks. Comprehensive testing based on this finer metric reveals tradeoffs that had not been captured previously. Our results involve solvable 3-coloring problems, as well as random binary DCSPs. These results have motivated us to develop a hybrid technique that reconciles the tradeoffs we have identified. The basic intuition of our hybrid technique is to avoid thrashing on solvable problems through coordinated mediation, while maximizing concurrent work in the mediation process itself.

In Section 2 we introduce the DCSP formally, and discuss previous work on measuring problem solving complexity of DCSP algorithms. In Section 3 we discuss ABT and APO,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

along with new variations including our decentralized hybrid. Experimental results are presented and discussed in Section 5, and concluding remarks are provided in Section 6.

2. BACKGROUND

2.1 DCSP Definition

The distributed constraint satisfaction problem was first discussed by Sycara *et. al.* and Yokoo *et.al.* as a way of formalizing Cooperative Distributed Problem Solving [12, 10]. A DCSP is formally defined as a constraint satisfaction problem (CSP) of the following form:

- a set of n variables, $V = \{x_1, \ldots, x_n\}$
- a set of discrete finite domains for each variable, $D = \{D_1, \ldots, D_n\}$
- a set of constraints $R = \{R_1, \ldots, R_m\}$ where each $R_i(d_{i1}, \ldots, d_{ij})$ is a predicate on the Cartesian product of the domains of all the variables referenced by that constraint. The constraint is said to be *satisfied* if the assignments of each referenced variable satisfy the constraint.

An agent, i, is said to *know* about a particular set of variables, V_i, and constraints, C_i. We will call the group of agents that agent i is connected to by constraints i's neighborhood, N_i. The goal of the agents is to instantiate their variables so that all constraints are satisfied. As in other work, for the sake of simplicity, we restrict our presentation to situations where each agent controls ("owns") a single variable and knows about all the constraints that refer to that variable. We further limit our discussion to binary constraints (constraints between two variables). It has been shown that more general problems can be reduced to problems that conform to these restrictions. In Section 6 we briefly discuss why the trends observed in this study should extend to these more complex classes of DCSPs.

2.2 Related Work

The original descriptions of ABT and APO appear in [12] and [5] respectively. In empirical studies of APO and backtracking based algorithms, such as ABT and Asynchronous Weak Commitment [11] (AWC), computational complexity measurements were often based on somewhat coarse performance metrics. In particular, these experiments typically did not differentiate between steps that required significantly different amounts of computation, e.g. equally counting as a single step a consistency check in AWC and a full backtracking search in a mediation session. Because they also focused primarily on solvable 3-coloring problems these experiments also raised the question of whether they were representative of other problem classes. In this paper we revisit the coordination tradeoffs involved in solving DCSPs using finer computational metrics and a more extensive class of problem instances - both solvable 3-coloring problems and random binary DCSPs. Our experiments reveal a richer set of performance tradeoffs than had been reported earlier. In addition, we present novel search configurations that aim to reconcile these tradeoffs.

Our empirical analysis uses a metric introduced by Meisels *et.al.* [6] that records the number of non-concurrent constraint checks (NCCCs) associated with different distributed search procedures. This metric distinguishes between complex computational steps involving multiple constraint checks, and much simpler ones. Constraint checks are the preferred computational unit used to measure performance of centralized constraint processing algorithms (see for example [9]). By counting non-concurrent checks the NCCCs metric is akin to the notion of a make-span or throughput measurement in scheduling [4].

The results presented in this paper build on our preliminary work comparing different mediation session selection rules for APO [1].

3. BASIC BACKTRACKING AND MEDIATION ALGORITHMS

This section outlines key features of ABT and APO, focusing on elements that impact computation and communication complexity.

3.1 Asynchronous Backtracking

ABT is an asynchronous implementation of the traditional backtracking search approach to solving centralized CSPs.

A summary of the procedures involved in the original description of ABT is shown in Figure 1. The algorithm begins with an initialization process, during which each agent i is given a static priority, p_i (typically based on lexicographic order). Agent i then arbitrarily assigns a value, d_i, to its variable, x_i, and broadcasts it to all lower priority neighbors. When agent i receives an assignment message from another agent, j, it stores the assignment as a tuple in the set called its *agentview*. Agent i then checks the consistency of its *agentview* and its own assignment. When it finds that its own assignment violates a constraint it tries to assign a value to its variable that resolves consistency. If it cannot find such an assignment it begins the backtrack process, and informs a higher priority agent that its *agentview* is *nogood* (leads to an unsatisfiable situation for the agent)[1]. The algorithm proceeds until all of the agents have variable assignments consistent with their *agentviews*, or one of the agents discovers that the problem is infeasible. For a full description of the ABT algorithm along with proofs of completeness and correctness, the reader is directed to its original description in [12].

An agent executing ABT uses constraint checks[2] to determine the consistency of its variable assignment. Because the agent knows only about the assignments of higher priority agents (as defined in its *agentview*), this amounts to checking each constraint with a higher priority agent. Notice that the original description of ABT involves recording each received nogood as a constraint (see "**when received** `nogood`" in Figure 1). As a result, each learned nogood is among the set of constraints that must be checked for consistency. Since the storage of nogoods can grow exponentially on harder problem instances, checking nogoods as constraints can significantly increase the number of NCCCs used by ABT. It has been shown, however, that agents can avoid much of the constraint checking work associated with nogoods by eliminating them from storage once they become *stale* (they

[1]We are aware that an agent can potentially resolve the subset of its agentview that resulted in the conflict. However, this has been shown to provide little benefit.

[2]Note that this refers to constraint checks in general, whether or not these checks are concurrent.

```
procedure init()
  calculate priority, p_i (usually lexicographic order);
  choose d_i ∈ D_i arbitrarily; set x_i ← d_i;
  agent_view_i ← ∅;
  broadcast();

when received ok?(j, d_j)
  update_agent_view(j, d_j); /* stores j's assignment */
  check_agent_view();

procedure update_agent_view(j, d_j)
  if agent_view_i contains assignment for j then
    remove assignment of j from agent_view_i;
  end if
  agent_view_i ← agent_view_i ∪ {⟨j, d_j⟩};

procedure check_agent_view()
  if ¬consistent(agent_view_i ∪ {⟨i, x_i⟩}, C_i) then
    if choose_value() then
      broadcast();
    else
      resolve_conflict();
    end if
  end if

procedure broadcast()
  for agent j ∈ N_i | p_j < p_i do
    send ok?(i, x_i) to agent j;
  end for

procedure choose_value()
  for d_i ∈ D_i do
    if consistent(agent_view_i ∪ {⟨i, d_i⟩}, C_i) then
      x_i ← d_i; return ⊤;
    end if
  end for
  return ⊥;

procedure resolve_conflict()
  if agent_view_i ≠ ∅ then
    find agent j ∈ agent_view_i with lowest priority, p_j;
    send nogood(i, agent_view_i) to agent j;
  else
    No solution!
  end if

when received nogood(j, agent_view_j)
  for agent k in agent_view_j and not in agent_view_i do
    let d_k be the value of agent k in agent_view_j;
    update_agent_view(k, d_k);
    request that k add i to N_k;
  end for
  let c_j be the constraint that agent_view_j is nogood;
  C_i ← C_i ∪ {c_j};
  check_agent_view();
```

Figure 1: Procedures Defining the Behavior of an ABT Agent i

refer to a branch of the search tree that has already been discarded) [2]. The necessary revisions to ABT for pruning stale nogoods are given in Figure 3.1. When measuring the computational requirements of this revised version of ABT, it is important to take into account the additional checks that have been introduced to determine whether or not a nogood is stale. However, because the number of stale nogoods tends to grow exponentially, the savings typically far outweigh this additional computational cost. We would like to note additionally, that this improvement cannot be used in A WC because priorities are assigned dynamically and stale nogoods may become relevant again. This makes AWC less competitive when measuring computational requirements based on consistency checks. For this reason we do not report experiments with AWC in this paper.

Another configuration of ABT involves having agents read all queued messages before trying to re-assign their variables. In contrast to checking consistency after each incoming assignment update or nogood, agents processes all the information they have received before performing their next consistency check ("**procedure** `check_agent_view`"). This small variation has recently been reported to provide significant improvements [13].

3.2 Asynchronous Partial Overlay

A summary of the procedures involved in a basic mediation framework for DCSPs is given in Figure 3. Procedures that are not redefined are assumed to be identical to the description in Figure 1. The initialization process of a mediation based algorithm proceeds the same way as in ABT. However, when agent i broadcasts its value assignment, it does not consider priority and sends the assignment to all its neighbors. When agent i receives an assignment message it proceeds in the same fashion as ABT to update its *agentview* and check the consistency of relevant constraints. If agent i detects a conflict and cannot resolve it locally by changing the value of its variable it initiates a conflict resolution process. In a mediation based algorithm the this involves agent i inviting all of its neighbors to join a mediation session - the size of the mediation session is defined by agent i's neighborhood. Once all of agent i's neighbors have each either accepted or rejected the invitation, a mediation session begins. A mediation session only involves the agents that have accepted the invitation. An agent can only accept to participate in one mediation session at a time. Agents choose which invitation to accept according to a mediation session selection rule, which we denote $R_{\prec}$. Additional details of the APO algorithm, a specific mediation based implementation, can be found in [5].

In the following Section we review different possible medi-

```
when received ok?(j, d_j)
  update_agent_view(j, d_j);
  prune_stale_nogoods();
  check_agent_view();

procedure prune_stale_nogoods()
  for nogood constraint c^ng ∈ C_i do
    let agent_view_c be the agentview that created c^ng;
    if agent_view_c ⊈ agent_view_i ∪ {⟨i, x_i⟩} then
      C_i ← C_i \ {c^ng};
    end if
  end for

when received nogood(j,agent_view_j)
  for agent k in agent_view_j and not in agent_view_i do
    let d_k be the value of agent k in agent_view_j;
    update_agent_view(k, d_k);
    request that k add i to N_k;
  end for
  if agent_view_j ⊆ agent_view_i ∪ {⟨i, x_i⟩} then
    let c_j be the constraint that agent_view_j is nogood;
    C_i ← C_i ∪ {c_j};
    check_agent_view();
  end if
```

Figure 2: Revisions to ABT for Nogood Pruning

```
procedure broadcast()
  for all agents j ∈ N_i do
    send ok?(i, x_i) to agent j;
  end for

procedure resolve_conflict()
  accepted_i ← ∅;
  for agent j ∈ N_i do
    send invitation(i) to agent j;
    rsvp_i ← rsvp_i ∪ {j};
  end for

when received I : {invitation(j_1),...,invitation(j_n)}
  /* I is a set of n invitations received at the same time */
  if agent i is already involved in a mediation session then
    send reject_invitation(i) to all inviting agents;
  else
    find invitation(j_k) ∈ I specified by selection rule, R_≺;
    send accept_invitation(i) to agent j_k;
    send reject_invitation(i) to all other inviting agents;
  end if

when received accept_invitation(j)
  accepted_i ← accepted_i ∪ {j};
  rsvp_i ← rsvp_i \ {j};
  if is_empty(rsvp_i) then
    mediate(accepted_i);
  end if

when received reject_invitation(j)
  rsvp_i ← rsvp_i \ {j};
  if is_empty(rsvp_i) then
    mediate(accepted_i);
  end if
```

Figure 3: Mediation DCSP Algorithm Framework (procedures not redefined are given in Figure 1)

ation session selection rules and mediation procedures, and discuss their constraint checking requirements. Independently of the mediation session selection rule and mediation procedure, mediation based agents use constraint checks to determine the consistency of their variable assignments.

4. MAPPING MEDIATION DIMENSIONS

The space of possible mediation strategies has been the subject of limited investigation. In this paper we focus on two mediation dimensions that we have found to have a significant impact on the performance of mediation based algorithms, namely mediation session selection rules and mediation procedures themselves.

4.1 Mediation Session Selection Rules

The mediation session selection rule described in the original APO algorithm is biased toward the selection of larger mediation sessions. Preliminary work reported in [1] presented a computational model along with initial results suggesting that, under certain circumstances, focusing first on smaller sessions can yield significant performance improvements. Intuitively, because mediation sessions may involve full backtrack search, the complexity of large mediation sessions will tend to dominate that of smaller sessions. Accordingly, focusing first on large sessions will lead to unnecessary computational efforts, when smaller sessions are sufficient. In this study we explore the performance impact of mediation session size by examining the following two mediation session selection rules.

- $\boldsymbol{R}_{\prec}^{\texttt{APO}}$: this mediation session selection rule was suggested in the original definition of APO. Agents evaluate invitations based on the size of the mediation session they would lead to, and pick the invitation corresponding to the largest one.
- $\boldsymbol{R}_{\prec}^{\texttt{IAPO}}$: this mediation session selection rule is the inverse of the original selection rule. It instructs agents to choose the smallest mediation session.

These two rules can be implemented by including the size of the inviting agent's neighborhood in each invitation. For all practical purposes the computational complexity of evaluating mediation session invitations using these two rules can be ignored. The evaluation can be performed by each agent in time linear in the number of invitations, which does not exceed the size of the agent's neighborhood. This computation is typically dominated by the computation performed as part of the mediation procedure itself.

4.2 Mediation Procedures

procedure `mediate`^{BB}`(accepted`$_i$`)`
  let I be the CSP instance defined by `accepted`$_i$;
  $P^* \leftarrow$ a conflict minimizing solution to I using BB;
  `broadcast-solution(`P^*`)`;

procedure `mediate`^{BT}`(accepted`$_i$`)`
  let I be the CSP instance defined by `accepted`$_i$;
  $P^* \leftarrow$ a feasible solution to I using CBT;
  `broadcast-solution(`P^*`)`;

procedure `mediate`^{ABT}`(accepted`$_i$`)`
  let I be the sub-DCSP defined by agents in `accepted`$_i$;
  `init()` ABT on I (as described in Section 3.1);

procedure `broadcast-solution(`P`)`
  if $P \neq \emptyset$ **then**
    for $\langle j, d_j \rangle \in P$ **do**
      send `accept_value(`d_j`)` to agent j;
    end for
  else
    No solution!
  end if

when received `accept_value(`d`)`
  $x_i \leftarrow d$;
  `check_agent_view();`

Figure 4: Summary of Mediation Procedures

The mediation procedure originally proposed with APO involved a *mediator* (the agent that sent the invitations) centralizing the sub-problem defined by the agents in the session, and solving it locally. The mediator used a branch and bound search to find a solution that minimized conflicts with agents outside of the session. In this Section we describe the original APO branch and bound mediation procedure, and propose two new mediation procedures (summarized in Figure 4). We introduce a backtrack search mediation procedure to explore the performance tradeoffs involved in finding a conflict minimizing solution. We also introduce a novel hybrid procedure where mediation is performed in a distributed fashion using ABT. This hybrid algorithm helps us explore the performance tradeoffs between concurrency and coordination. The different mediation procedures we examine require different information to accompany `accept_invitation` messages. The requirements are described in detail along with the procedures below.

Branch and Bound (APO-BB): The branch and bound mediation strategy was the method originally proposed with APO. When using this mediation strategy, a mediator must gather all of the information about the local sub-problems of agents involved in the session (the agents that accepted the invitation) including:

- the set of constraints, C_j, that apply to agent j's variable,
- the entire domain of agent j's variable, D_j,
- and a label function defined on agent j's domain, $L_j : d_j \rightarrow \{k_1, \ldots, k_n \mid \text{agent } k_i \text{ will be conflicted with } d_j\}$, that indicates all of the agents known to be in conflict with each of j's values.

This information is used by the mediator to perform a branch and bound search. This search finds a feasible solution, if one exists, to the sub-problem pertaining to agents that are part of the mediation session. The solution minimizes the number of constraints violated for agents *outside* of the session, which can be determined using the label functions. In terms of measuring computation, the branch and bound procedure uses constraint checks as it proceeds to construct all solutions that cannot be ruled out by bounding. However, it does not use constraint checks to determine how many external violations a partial or full solution results in. The number of external violations is simply the size of the set returned by the label function.

Backtracking (APO-BT): This mediation procedure proceeds exactly as the BB variant, except that the mediator attempts to find any feasible solution. The solution may not minimize conflicts with out-of-session agents. Using this mediation procedure, agents need not send fully labeled domains to the mediator[3]. In terms of measuring computation, the backtracking mediation process uses constraint checks in the same fashion as the branch and bound process, except it short-circuits computation as soon as a feasible solution is found.

Asynchronous Backtracking (APO-ABT): We introduce this mediation procedure as a novel hybrid between APO and ABT where mediations are performed using the ABT protocol. This mediation strategy downplays the importance of the mediator, who no longer needs to collect local constraint information from the other agents involved. Because agents can be involved in one session at a time APO-ABT synchronizes the overall problem solving process, while exploiting opportunities for concurrency at the lower level. The ABT mediation process uses constraint checks as described in 3.1.

Any of these three mediation strategies can be used with either of the mediation session selection rules, making for a total of six proposed configurations. Also, each of the mediation procedures uses the values of agents at the time of the mediation as arbitrary starting values. This was suggested in the original specification of APO and helps to respect work done during earlier sessions.

5. EMPIRICAL EVALUATION

5.1 Empirical Setup

The experiments in this paper are intended to compare the performance of the six mediation configurations outlined in the previous section (five of which represent novel configurations), and ABT. We present results from two different sets of experiments, one involving random solvable distributed 3-coloring (D3C) problems and another on random binary DCSP instances. A D3C problem involves each agent choosing one of three colors. The problem is satisfied if no two

[3]Agents may, however, need to inform the mediator of a conflict after assignments have been dictated. This allows the mediator to add an out-of-session agent to its neighborhood to ensure completeness.

agents connected by a constraint share the same color. A random binary DCSP problem involves arbitrary random constraints between pairs of agents. In all of our experiments all of the algorithms were provided with the same instances and initial random values for variables.

Random binary CSP instances are typically characterized by the number of variables, n, the number of values in each variable's domain, k, the density of the constraint graph, $p1$, and the tightness of the constraints, $p2$ [9]. Our random DCSP experiments use $n = 10$, and $k = 10$. $p1$ specifies the probability that a constraint exists between any two agents. In this paper we fix $p1 = 0.7$, a relatively high density value, behaviors on problems with different values of $p1$ result in similar observations. We vary $p2 = 0.1$ to $p2 = 0.9$, which specifies the probability that two values in constrained agent's domains are in conflict. We draw at least 100 instances for each value of $p2$, and about 500 instances near the phase transition where $p2 \in [0.4, 0.6]$. The "phase transition" refers to an area of the problem parameters where problems transition from being under-constrained, and thus easy to solve, to over-constrained but still difficult to prove infeasible [9].

The D3C instances are characterized by two parameters, the number of agents involved, n, and the number of constraints per agent m. Since we are looking at solvable D3C problems with no phase transition effects, we do not see a need to vary problem density and instead fix $m = 2.7$ (typically considered high density). We vary the number of agents from $n = 15$ to $n = 60$ in increments of 9 (these values must be multiples of 3 to properly guarantee solvable instances). With these instances we examine how the algorithms scale to larger problems that are guaranteed to be solvable. We draw at least 100 instances for each value of n (although some algorithms are removed as they become incapable of scaling based on a 5 minute cpu-time cutoff per instance) according to the method provided in [7], for some larger values of n we draw up to 500 instances to ensure the statistical significance of our results.

The results presented indicate the mean number of messages and non-concurrent constraint checks (NCCCs) used by each algorithm across the different simulations. The NCCCs are measured using the techniques described in Sections 3 and 4, and [6]. We provide 95% confidence intervals when necessary, results missing intervals can be assumed to be significant.

5.2 Results on Random CSP Instances

The NCCC results on random CSP instances are shown in Figures 5(a) and 5(c). Examining these results, we can see that the phase transition is clearly accentuated when $p2 = 0.5$ for all the algorithms. Before this point problems are relatively easy to solve and there is little difference observed. When $p2 = 0.5$ we can see that ABT, with the improvements described in Section 3.1, uses about 30,000 NCCCs on average, which is consistent with results reported in [13]. In addition the APO-ABT and IAPO-ABT hybrids both use about the same amount of NCCCs as ABT itself (see Figure 5(c)). This is not surprising, since about half of the problems in this area require a mediation of the entire network to prove infeasibility, and the hybrid algorithms reduce to running ABT on the whole problem at that point. The branch and bound based algorithms, APO-BB and IAPO-BB, perform poorly during the phase transition, requiring about 200 times more NCCCs than ABT. This is due to the fact that many of these problems must be fully centralized with these procedures to prove infeasibility, and finding the solution that minimizes external conflicts amounts to exploring nearly the entire search tree during each mediation. The backtracking based mediation algorithms, APO-BT and IAPO-BT, require nearly 100 times fewer NCCCs than the branch and bound algorithms, showing that finding external conflict minimizing solutions costs a significant number of NCCCs and provides far less less benefit on random problems. However, the ABT based algorithms use significantly fewer NCCCs than the backtracking ones, showing the cost of centralizing the problem solving effort.

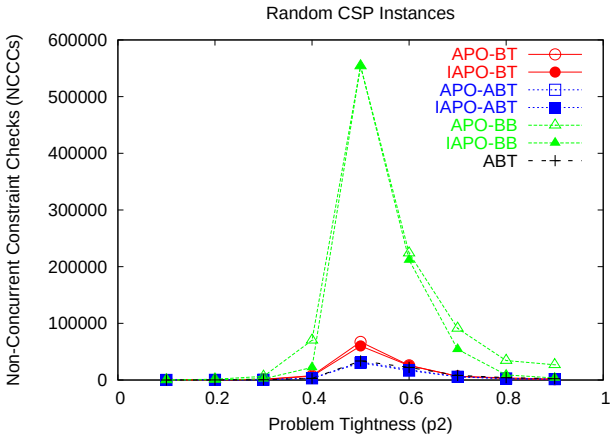
The performance in terms of communication requirements on random instances is shown in Figures 5(b) and 5(d). These results are basically a mirror image of the NCCC results, with ABT and the ABT based mediation algorithms requiring on average about 30 times more messages than the other algorithms (again this is consistent with results reported for ABT in [13] and [5]). Additionally we can see from the results in Figure 5(d) that the mediation techniques with the IAPO preference rule use about twice as many messages as those with the APO preference rule. This is not surprising considering that the IAPO preference rule chooses smaller sessions and therefore needs more of them.

5.3 Results on Solvable D3C Instances

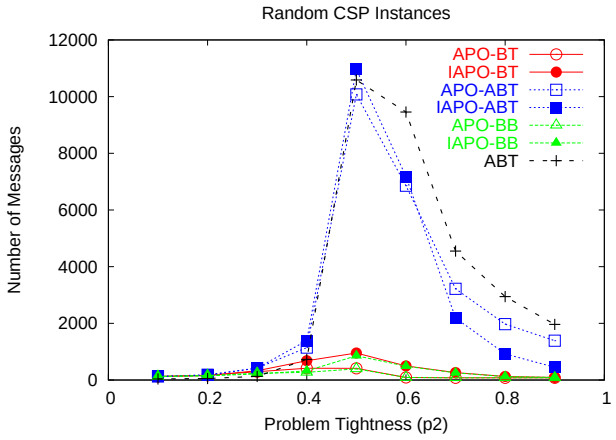
The NCCC results for solvable D3C instances are shown in Figure 6(a). These results provide some very interesting contrasts with the random CSP NCCC results. The algorithm that performs best on the solvable D3C instances by a large margin is our IAPO-BB algorithm. This algorithm was the only one capable of solving 60 agent problems within an allotted 5 cpu-minutes per instance, and it was able to solve smaller instances orders of magnitude faster than the other algorithms. Recall that on the random CSP instances the branch and bound based algorithms performed nearly 200 times *worse* than the other algorithms, yet on the high density solvable instances our IAPO-BB scales extremely well. What we are seeing is that on high-density solvable instances the benefit of finding solutions that minimize conflicts outside of a mediation session far outweighs the extra work necessary to do so.

Another interesting result on the solvable D3C instances is the benefit of the IAPO selection rule over the APO selection rule. All of the variants with our IAPO selection rule used significantly fewer NCCCs than their APO-based counterparts. Favoring smaller mediation sessions over larger ones, as IAPO does, helps mediators avoid solving unnecessarily large problems until absolutely needed. Our results suggest that on solvable D3C instances it is often not necessary to mediate over larger groups, and consequently IAPO avoids some of the unnecessary computation of APO. These findings are consistent with preliminary results we had reported in [1].

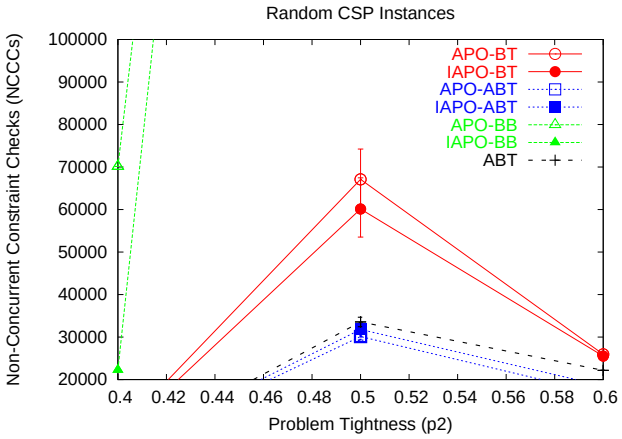
When we look at the handful of algorithms that were able to solve 51 agent instances we can see that only the ABT-based algorithms remain along with IAPO-BB. Among the ABT-based algorithms it is interesting to note that both our APO-ABT and IAPO-ABT hybrid algorithms require much fewer NCCCs than ABT itself on average (the IAPO-ABT algorithm requiring significantly fewer). This begins to



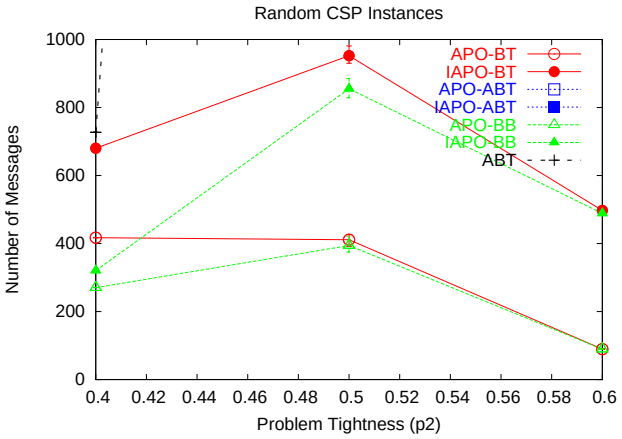
(a) Mean NCCCs



(b) Mean Messages



(c) Mean NCCCs with 95% confidence intervals (zoomed from Figure 5(a), $p2 = 0.4$ to $p2 = 0.6$)



(d) Mean Messages with 95% confidence intervals (zoomed from Figure 5(b), $p2 = 0.4$ to $p2 = 0.6$)

Figure 5: Performance on Random DCSP Instances, $(n = 10, k = 10, p1 = 0.7)$

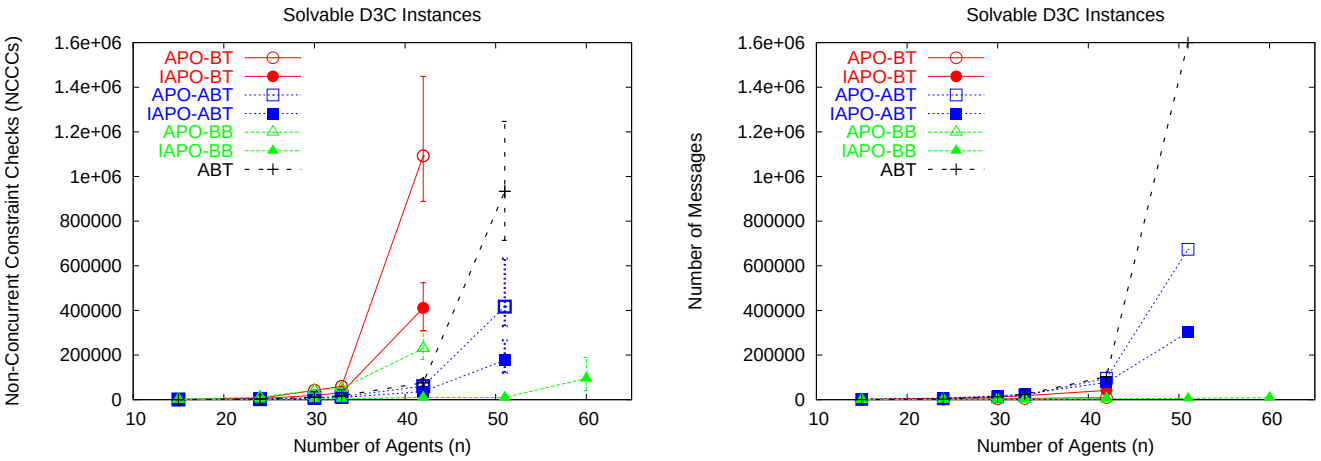
show the benefits of partially synchronizing problem solving efforts on solvable D3C instances to cut back on the amount of backtracking in ABT. The fact that all three of these algorithms perform significantly better than the BT-based algorithms shows the benefit of distributing work load on larger problem instances.

When we look at the performance in terms of communication requirements for solvable D3C instances in Figure 6(b), it is not surprising to find that all of the centralized approaches use significantly fewer messages than the decentralized ones. What is interesting is the fact that our APO-ABT and IAPO-ABT hybrid algorithms both used fewer messages than ABT itself (recall these two used fewer NCCCs as well). Not only that, but the IAPO variant, IAPO-ABT, used fewer messages than the APO variant, whereas on the random instances the opposite was true. Both of these observations are due to the fact that the mediation based techniques required less backtracking than ABT and among the mediation based techniques, the IAPO variant required less backtracking than the APO variant. On the solvable D3C instances the decrease in backtracking during problem solving led to less communication as well.

6. CONCLUSION

In this paper we provided a comprehensive abstraction for the space of DCSP mediation based approaches. In the process we identified gaps in existing frameworks, and proposed new effective strategies to fill them. The new strategies include a mediation session selection rule that favors smaller mediation sessions, a centralized mediation approach that shortcuts a significant amount of excess computation on random instances, and a decentralized asynchronous mediation strategy based on ABT.

Our empirical results represent the first truly comprehensive view of the tradeoffs between backtracking and mediation based approaches for DCSPs. We tested the different algorithms on solvable 3-coloring and random binary DCSPs.



(a) Mean NCCCs with 95% confidence intervals

(b) Mean Messages

Figure 6: Performance on Solvable D3C Instances, ($m = 2.7$)

Our results confirmed that our newly proposed strategies in this paper dominate previous techniques under some circumstances. Specifically, our mediator selection rule that favors smaller sessions, IAPO, dominates the previously used rule that favors larger ones, APO, in computational complexity across all instances. IAPO also dominates APO in communication complexity on solvable D3C instances. Additionally, our hybrid decentralized mediation strategy based on ABT is as good as ABT in communication and computation on random instances, and strictly better in both areas on solvable D3C instances. This suggests that the benefits of synchronization for DCSPs reported elsewhere [5, 13] may exist primarily on hard solvable instances.

7. ACKNOWLEDGEMENTS

The research reported in this paper has been funded by the National Science Foundation under ITR Grant 0205435. The authors would also like to thank Roie Zivan and Amnon Meisels for their help calibrating the NCCC metric.

8. REFERENCES

[1] M. Benisch and N. Sadeh. How (not) to choose mediators for distributed constraint satisfaction. In *Proceedings of LSMAS at AAMAS'05*, 2005.

[2] C. Bessiere, A. Maestre, and P. Messeguer. Distributed dynamic backtracking. In *Proceedings of DCR Workshop at IJCAI'01*, 2001.

[3] Y. Hamadi. Distributed interleaved parallel and cooperative search in constraint satisfaction networks. In *Proceedings of IAT'01*, 2001.

[4] L. Lamport. The parallel execution of do loops. *Commun. ACM*, 17(2):83–93, 1974.

[5] R. Mailler and V. Lesser. Using Cooperative Mediation to Solve Distributed Constraint Satisfaction Problems. In *Proceedings of Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2004)*, volume 1, pages 446–453, New York, 2004. IEEE Computer Society.

[6] A. Meisels, E. Kaplansky, I. Razgon, and R. Zivan. Comparing performance of distributed constraints processing algorithms. In *Proceedings of DCR Workshop at AAMAS'02*, 2002.

[7] S. Minton, M. D. Johnston, A. B. Phillips, and P. Laird. Minimizing conflicts: A heuristic repair method for constraint satisfaction problems. *Artificial Intelligence*, 58(1-3):161–205, 1992.

[8] P. J. Modi, M. Veloso, S. Smith, and J. Oh. Cmradar: A personal assistant agent for calendar management. In *Agent Oriented Information Systems, (AOIS) 2004*, 2004.

[9] B. M. Smith and M. E. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artif. Intell.*, 81(1-2):155–181, 1996.

[10] K. Sycara, S. F. Roth, N. Sadeh, and M. S. Fox. Distributed constrained heuristic search. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1446–1461, December 1991.

[11] M. Yokoo. Asynchronous weak-commitment search for solving distributed constraint satisfaction problems. In *Proceedings of the First International Conference on Principles and Practice of Constraint Programming*, pages 88–102, 1995.

[12] M. Yokoo and E. H. Durfee. Distributed constraint satisfaction for formalizing distributed problem solving. In *12th IEEE International Conference on Distributed Computing Systems*, pages 614–621, 1992.

[13] R. Zivan and A. Meisels. Synchronous vs asynchronous search on DisCSPs. In *Proceedings of the First European Workshop on Multi-Agent Systems (EUMA)*, 2003.

[14] R. Zivan and A. Meisels. Dynamic ordering for asynchronous backtracking on discsps. In *Proceedings of CP'05*, 2005.

Multiply-Constrained Distributed Constraint Optimization

E. Bowring, M. Tambe
Computer Science Dept.
Univ. of Southern California
Los Angeles, CA 90089
{bowring,tambe}@usc.edu

M. Yokoo
Dept. of Intelligent Systems
Kyushu University
Fukuoka, 812-8581 Japan
yokoo@is.kyushu-u.ac.jp

ABSTRACT

Distributed constraint optimization (DCOP) has emerged as a useful technique for multiagent coordination. While previous DCOP work focuses on optimizing a single team objective, in many domains, agents must satisfy additional constraints on resources consumed locally (due to interactions within their local neighborhoods). Such resource constraints may be required to be private or shared for efficiency's sake. This paper provides a novel *multiply-constrained DCOP* algorithm for addressing these domains which is based on mutually-intervening search, i.e. using local resource constraints to intervene in the search for the optimal solution and vice versa. It is realized through three key ideas: (i) transforming n-ary constraints to maintain privacy; (ii) dynamically setting upper bounds on joint resource consumption with neighbors; and (iii) identifying if the local DCOP graph structure allows agents to compute exact resource bounds for additional efficiency. These ideas are implemented by modifying Adopt, one of the most efficient DCOP algorithms. Both detailed experimental results as well as proofs of correctness are presented.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Design, Theory

Keywords

constraint reasoning, DCOP, multiagent systems, privacy

1. INTRODUCTION

Distributed Constraint Optimization (DCOP) [1,6,9,11] is a useful technique for multiagent coordination with applications in distributed meeting scheduling and distributed factory and staff scheduling [8]. In a DCOP, cooperative distributed agents, each in control of a set of variables, assign values to these variables, so as to optimize a global objective function expressed as an aggregation of utility functions over combinations of assigned values.

While recent advances in efficient DCOP algorithms are encouraging [1, 6, 9, 11], these algorithms focus on optimizing a single objective and fail to capture the complexities that arise in many domains where agents must adhere to resource constraints, e.g. budgets or fuel. These resource constraints necessitate multiply-constrained DCOP algorithms that optimize a global objective, while ensuring that resource limits are not exceeded. While in some domains agent must keep these resource constraints private (e.g. travel budgets in distributed meeting scheduling [8]), in others resource constraints may be non-private (e.g. overtime limits in staff allocation for distributed software development).

There are three main challenges that must be addressed in designing multiply-constrained DCOP algorithms. First, agents' n-ary resource constraints add to DCOP search complexity. Hence, agents must quickly prune unproductive search paths. Second, harnessing state-of-the-art DCOP algorithms (henceforth, referred to as "singly-constrained" DCOP algorithms) is crucial. However, existing algorithms are not deisgned to handle resource constraints that are local, possibly private and defined over n-ary domains. Third, algorithms must exploit constraint revelation to gain efficiency when privacy is not required. (Unlike work which uses cryptographic techniques in DisCSP/DCOP for privacy [12, 15], we do not insist on such watertight privacy, given potentially significant communication and computational overheads.)

This paper presents a novel multiply-constrained DCOP algorithm which employs *mutually-intervening searches* to address the first challenge above: while an agent immediately intervenes in the search for the global optimal if its local resource constraint is violated, opportunistic search for the global optimal solution obviates testing all partial solutions for resource constraint satisfaction. There are three key ideas in this algorithm, which are facets of mutually-intervening search: constraint-graph transformation, dynamically-constraining search and local acyclicity. The first idea, *constraint-graph transformation*, is motivated by the second challenge above: harnessing singly-constrained DCOP algorithms for multiply-constrained DCOPs. The key innovation is in efficiently employing virtual variables to enforce an agent's resource constraints within the singly-constrained DCOP algorithms. These variables indicate resource constraint violations with asynchronous communications of high negative costs, which preemptively prune search paths. Yet, for correctness and privacy preservation, we restructure the DCOP graph and appropriately place the virtual variables in that graph. The following two ideas address the third challenge above by exploiting the privacy-efficiency tradeoff. In *dynamically-constraining search*, an agent reveals to its neigh-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

bors an upper-bound on any non-private resource constraint. When optimizing the global objective function, its neighbors only select values that abide by the bounds, improving algorithmic efficiency. The final idea, *local acyclicity*, further improves efficiency in locally acyclic DCOP graphs by allowing the communicated resource bounds to be tightened without sacrificing algorithmic correctness. In particular, we define T-nodes, which are variables whose local graph acyclicity allows for the dynamic use of exact bounds and do not require virtual variables. We show that these tighter bounds can not be applied at non-T-nodes.

While we illustrate these ideas by building on top of Adopt, one of the most efficient DCOP algorithms [9], our techniques could be applied to other algorithms, e.g. OptAPO, SynchBB [6, 14]. We present a multiply-constrained Adopt algorithm that tailors its performance to three situations: 1) when a resource constraint must be kept private, 2) when a constraint is sharable but the variable is not a T-node, and 3) when a constraint is sharable and the variable is a T-node. These different techniques can be employed simultaneously by different variables in the same problem. We experimentally compare these techniques and illustrate problems where agents gain the most efficiency by sharing resource constraints and problems where agents lose no efficiency by maintaining privacy.

2. PROBLEM DEFINITION

2.1 DCOP

A DCOP [1, 9, 11] consists of n variables, $\{x_1, x_2, \ldots, x_n\}$, assigned to a set of agents who control the values they take on. Variable x_i can take on any value from the discrete finite domain D_i. The goal is to choose values for the variables such that the sum over the set of constraint cost functions (f_{ij}) is minimized, i.e. find an assignment, A, s.t. F(A) is minimized: $F(A) = \sum_{x_i,x_j \in V} f_{ij}(d_i, d_j)$, where $x_i \leftarrow d_i, x_j \leftarrow d_j \in A$. Variables x_i and x_j are considered neighbors since they share a constraint.

In the example in Figure 1, x_1, x_2, x_3, and x_4 are variables each with domain {0,1} and identical f-cost functions on each link (shown for the x_1 to x_2 link), $F(x_1 \leftarrow 0, x_2 \leftarrow 0, x_3 \leftarrow 0, x_4 \leftarrow 0) = 4$ and the optimal assignment would be ($x_1 \leftarrow 1, x_2 \leftarrow 1, x_3 \leftarrow 1, x_4 \leftarrow 1$). While the above commonly used definition of DCOP emphasizes binary constraints, *general DCOP representations* may include n-ary constraints.

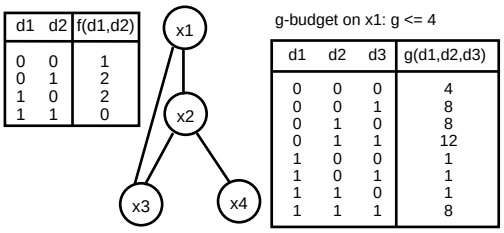


Figure 1: Multiply-Constrained DCOP graph

2.2 Multiply-Constrained DCOP

In Multiply-Constrained DCOP we make use of n-ary DCOP constraints by adding a new cost function g_i on a subset of x_i's links and a g-budget G_i which the accumulated g-cost must not exceed. Together this g-function and g-budget constitute a g-constraint. Figure 1 shows an example g-constraint on x_1. In the example, if x_1, x_2, x_3 each take on the value of 1 (leading to an optimal f-cost) then the g-cost is 8, which violates x_1's g-budget of 4. Since g-cost functions cannot be merged with f-cost functions, each value must be selected based on both f and g, and hence this is a multiply-constrained DCOP. It is straightforward to extend this framework to multiple g-constraints on a variable.

In general, the combined g-cost can be an arbitrary function on the values of x_i and its neighbors.[1] We define g-constraints to be private or shared. If a g-constraint can be shared and the g-cost function is the sum of the g-costs of the links impinging upon x_i, we can improve efficiency by exploiting the additive nature of the function. In the rest of the paper, we assume such additivity for shared g-constraints but make no such assumption for private g-constraints. Given the g-cost functions and g-budgets, we now modify the DCOP objective to be: find A s.t. F(A) is minimized: $F(A) = \sum_{x_i,x_j \in V} f_{ij}(d_i, d_j)$, where $x_i \leftarrow d_i, x_j \leftarrow d_j \in A$
and $\forall x_i \in V$
$g_i(d_i, \{d_j | x_j \in neighbors(x_i)\}) \leq G_i$

Multiply-Constrained DCOP is situated within the space of general DCOP representations mentioned in Section 2.1. However, it emphasizes three features: (i) n-ary (g-cost) constraints, (ii) private g-constraints, and (iii) the need to exploit the interaction between f- and g-constraints for pruning the search space. No current DCOP algorithm, including leading algorithms, is able to address all three issues. Specifically, Adopt [9] can handle (ii) but not (i) or (iii). OptAPO [6] and DPOP [11] handle (i) but not (ii) or (iii).

2.3 Motivating Domains

In this section we demonstrate the need for multiply-constrained DCOP algorithms with two example domains. These domains require agents to optimize an f-cost function and yet adhere to (private) g-constraints.

Distributed Meeting Scheduling: When members of organizations in separate locations collaborate, personal assistant agents must optimize their meeting schedules and yet adhere to travel budgets. Consider the example in Figure 2, where researchers, A, B, C and D are divided between organizations at Loc1 and Loc2. A and C need to meet, and B and D also need to meet. A and C are group leaders who manage travel budgets for their groups and wish to keep budgetary constraints private.

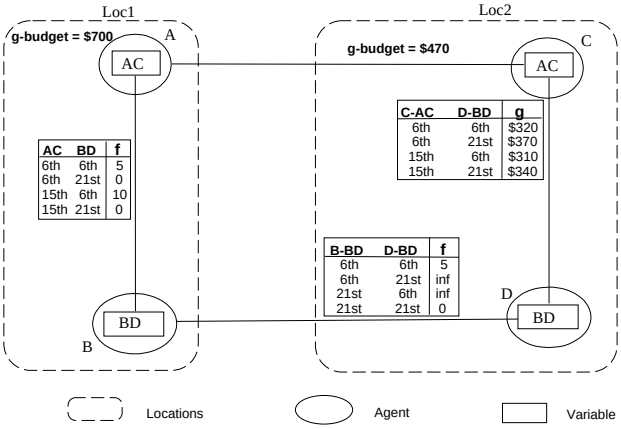


Figure 2: Meeting Scheduling Example

Multiply-constrained DCOP allows us to model this problem. We assume that agents share all their scheduling preferences (the exact representation of these preferences is not important for this example). The domain of each variable is a tuple of time-of-meeting

[1]While for simplicity we assume that the set of f-neighbors and g-neighbors are the same, this is not necessary.

and location, e.g. {6th Oct at Loc1, 15th Oct at Loc2}. The f-cost expresses agents' scheduling preferences and that they must agree on the meeting time (or else receive an infinite cost). The f-cost function between C and D reflects C's preference that the BD meeting precede the AC meeting (for readability not all of the f- and g-functions are shown in Figure 2). Since a meeting may require flying to that destination, the g-cost represents travel costs. The cost varies depending on the date of travel and whether all members are traveling on the same day and can share a cab. C has a travel budget of $470, represented via a g-budget on C's variable AC. C must pay for expenses incurred by C or D in either the AC or BD meeting if they are held at Loc1. A and C may not wish to reveal their budgets (hence private g-constraints).

Distributed Software Development: Many software companies have campuses across the globe and teams collaborate across both distances and time zones [3, 5]. While interdependent tasks within a project must be scheduled for their timely completion, a team liaison must videoconference — often after normal workhours — with the team to which the code is being sent to facilitate the handoff. Figure 2 shows an example involving five tasks $\{t_1, \ldots, t_5\}$, where agents in Team1 must complete t_1 and t_4 and Team2 must complete t_2, t_3 and t_5. The domain values are times at which a task can be completed. The f-function captures both the temporal precedence constraints and a preference function over the potential times (e.g. t_1 must complete before t_2, and prefer t_1 to start early in the day). A team liaison from Team1 must videoconference with Team2 during the first hour of t_2 and t_3, requiring the liaison to work overtime. To avoid burnout corporate policy may limit liaison overtime to 8 hours, establishing a non-private g-constraint.

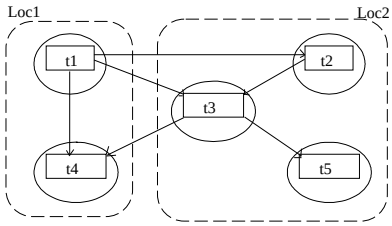


Figure 3: Software Development

3. BACKGROUND: ADOPT

Adopt [9] is an asynchronous complete DCOP algorithm. Adopt has been discussed in detail in the literature [1, 2, 9], so we provide only a brief description. Adopt organizes variables into a Depth-First Search (DFS) tree in which constraints are allowed between a variable and its ancestors or descendants, but not between variables in separate sub-trees. The constraint graph in Figure 1 is organized as a DFS tree. x_2 is a child of x_1 and x_3 is a descendant (but not a child) of x_1. In this paper, we will use the terms variable and node interchangeably.

Adopt employs two basic messages: VALUE and COST.[2] Assignments of values to variables are conveyed in VALUE messages that are sent to variables lower in the DFS tree, that share a constraint with the sender. For example, x_1 will send its VALUE messages to x_2 and x_3. To start, variables take on a random value and send out VALUE messages to get the flow of computation started. A COST message is sent from children to parents indicating the f-cost of the sub-tree rooted at the child (e.g. x_3 will send its COST messages to x_2 and x_2 sends COST messages to x_1). A variable keeps its current assignment until the lower bound on cost accumulated, i.e. the lower bound of its children's sub-trees plus the f-cost of its constraints with its ancestors, exceeds the lower-bound cost of another assignment. When this occurs, the variable will *opportunistically* switch its value assignment (unexplored assignments have a lower bound of zero). The root's upper and lower bounds represent the upper and lower bounds on the global problem; when they meet the optimal has been found and the algorithm terminates. Since communication is asynchronous, messages have a context, i.e. a list of the variable assignments in existence at the time of sending, attached to them to help determine information relevance.

[2]Adopt also uses THRESHOLD messages for improved efficiency, but this is orthogonal to the contributions in this paper.

4. MULTIPLY-CONSTRAINED ADOPT

This section is organized as follows: Section 4.1 describes the key ideas in Multiply-Constrained Adopt (MCA), Section 4.2 describes MCA in detail, and Section 4.3 provides correctness and complexity results.

4.1 Basic Ideas

The mutually intervening approach of MCA is realized through four techniques: constraint-graph transformation, dynamically constraining search, local acyclicity and exploiting local independence.

Constraint-graph transformation To exploit existing DCOP algorithms and maintain privacy, we add virtual variables to enforce n-ary g-constraints. The virtual variables asynchronously communicate infinite costs on g-constraint violations, preemptively pruning search paths, e.g. in Figure 4a, x'_1 is added to represent a constraint between x_1, x_2, x_3. While using extra variables to enforce n-ary constraints using binary ones has appeared in the centralized CSP literature [13], the novelty of our contribution is threefold. First, we embed these virtual variables in a singly-constrained DCOP algorithm and prove the correctness of the resulting asynchronous multiply-constrained DCOP algorithm. To that end, we restructure Adopt's DFS tree and appropriately place the virtual variable in the tree. Second, since such constraint-graph restructuring can change local acyclicity properties of other nodes, the preprocessing identifies and preserves local acyclicity properties where feasible. Third, by encapsulating an n-ary constraint in a virtual variable owned by the original variable's owner and placing it as a leaf in the DFS tree, we can protect the privacy of both the g-function and the g-budget.

Dynamically Constraining Search When permissible, it is important to exploit g-constraint revelation to gain efficiency in the f-optimization. We achieve this by requiring descendant nodes to only consider assignments that will not violate their ancestors' g-constraints. Specifically, we pass each descendant a bound (termed *g-threshold*) specifying how large a g-cost it can pass up, limiting its domain. This g-threshold represents an exact bound when the local graph is acyclic and an upper bound otherwise. If a value fails to satisfy an ancestor's g-threshold, nodes will not explore this value for f-optimality. Additionally, the opportunistic search for an optimal f necessitates checking for g-constraint violations only for those value combinations that are of low f-cost. Thus, the searches dynamically constrain each other, leading to performance improvements.

Local Acyclicity (T-nodes) The notion of local acyclicity is captured formally in our definition of T-nodes: variable x_i is a T-node if all neighbors of x_i lower in the DFS tree are children of x_i. In Figure 1, x_1 is not a T-node because x_3 is not its child, but x_2, x_3 and x_4 are all T-nodes. T-nodes enable the calculation of exact g-thresholds and elimination of virtual variables because their children respond independently to allocated g-thresholds.

```
% In preprocessing we assume an initial DFS tree has been built.
% Convention: x_i is self, x_j is a higher priority neighbor
% and x_k and x_l are lower priority neighbors.
Preprocessing
(1)   for each x_i from highest priority to lowest
(2)     if Tnode_i == false or private_i == true
(3)       x'_i is a new virtual variable
(4)       Neighbors(x'_i) ← Neighbors(x_i) ∪ x_i
(5)       Neighbors(x_i) ← Neighbors(x_i) ∪ x'_i
(6)       forall x_k ∈ Children(x_i)
(7)         if x_k is not a neighbor of x_l ∈ Children(x_i)
(8)           Neighbors(x_k) ← Neighbors(x_k) ∪ x_l
(9)       rebuildDFStree(x_1 ... x_n)
(10)  forall virtual variables x'_i,
(11)    parent(x'_i) ← lowest priority Neighbor of x'_i

Initialize
(12)  CurrentContext ← {}
(13)  initialize structures to store lb and ub from children
(14)  d_i ← d that minimizes LB(d)
(15)  if private_i == false and Tnode_i == true
(16)    forall x_l ∈ Children
(17)      for gt ← 0 ... G_i
(18)        GFmap(x_l, gt) ← min f(d_i, d_l) s.t. g(d_i, d_l) ≤ gt
(19)  if Tnode_i == true and private_i == false
(20)    calcOptimalSplit
(21)  else if private_i == false
(22)    calcUpperBound
(23)  backTrack

when received (VALUE, x_j, d_j, gThresh_j)
(24)  if private_j == false
(25)    add (x_j,d_j,gThresh_j) to CurrentContext
(26)  else
(27)    add (x_j,d_j) to CurrentContext
(28)  reset lb and ub if CurrentContext has changed
(29)  if private_i == false and Tnode_i == true
(31)    if CurrentContext has changed
(30)      forall x_l ∈ Children
(32)        GFmap(x_l, gt) ← min f(d_l, d_i) s.t. g(d_l, d_i) ≤ gt
(33)    calcOptimalSplit
(34)  backTrack;

when received (COST, x_k, context, lb, ub)
(35)  update CurrentContext
(36)  if context compatible with CurrentContext
        and Tnode_i == true and private_i == false
(37)    GFmap(x_l, gThresh_il) ← lb s.t.
          (x_i, d_i, gThresh_il) is part of context from x_k
(38)    calcOptimalSplit
(39)    if any gThresh_il has changed, reset lb,ub
(40)    else store lb,ub
(41)  else if context compatible with CurrentContext and gThresh_il
(42)    store lb and ub
(43)  backTrack

procedure backTrack
(44)  if x_i not a virtual variable
(45)    if no d satisfies gThresh_j LB, UB ← ∞
(46)    else if LB(d_i) > LB(d) for some d
(47)      d_i ← d that minimizes LB(d) and satisfies gThresh_j
(48)      if Tnode_i == true and private_i == false
(49)        GFmap(x_l, gt) ← min f(d_i, d_l) s.t. g(d_i, d_l) ≤ gt
(50)        calcOptimalSplit; reset lb and ub
(51)      if private_i == false
(52)        SEND (VALUE, (x_i, d_i, gThresh(x_k)))
              to each lower priority neighbor x_k
(53)      else SEND (VALUE, (x_i, d_i))
              to each lower priority neighbor x_k
(54)    if LB == UB:
(55)      if TERMINATE received from parent or x_i is root:
(56)        SEND TERMINATE to each child
(57)        Terminate execution;
(58)    SEND (COST, x_i, CurrentContext, LB, UB)
(59)  else % else a virtual variable
(60)    if g(d_i, CurrentContext) > gBudget(x_i)
(61)      SEND (COST, x_i, CurrentContext, ∞, ∞) to parent
(62)    else
(63)      SEND (COST, x_i, CurrentContext, 0, 0) to parent
```

Figure 4: Multiply-Constrained Adopt Pseudo-code

Exploiting Local Independence In applying different techniques to different nodes in a single problem, we exploit the locality of the g-constraints.

4.2 MCA Algorithm Description

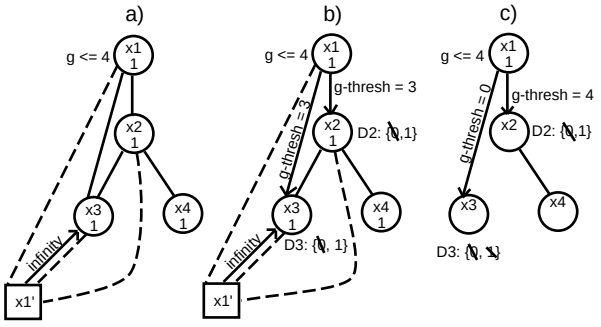


Figure 5: a) MCAP b) MCAS c) MCASA

Figure 4 presents pseudo-code for MCA. To emphasize new elements we include only brief descriptions of functionality unchanged from Adopt [9]. Lb and ub refer to the lower and upper bounds on children's f-costs for each value d_i while LB and UB refer to the overall lower and upper bound at x_i. MCA uses three subalgorithms: (i) MCA Private (MCAP) is used when a constraint may not be revealed, and it uses constraint-graph transformation, (ii) MCA Shared (MCAS) is used when a constraint is non-private but the variable is not a T-node, and it employs constraint-graph transformation and dynamically constraining search, and (iii) MCA Shared and Acyclic (MCASA) is utilized when a constraint is non-private and the variable is a T-node, and it uses dynamically constraining search. We take privacy to mean that neither the g-constraints nor the variables involved are explicitly revealed to other agents. We will discuss each technique separately and then indicate how they interact in the unified algorithm.

4.2.1 MCAP

An example snapshot of MCAP's execution is shown in Figure 5a. MCAP searches for an optimal solution for f and when an assignment violates a g-constraint, a feedback signal of infinite cost is sent to the cluster of nodes involved, zero cost otherwise (lines 60-63 in Figure 4). The feedback is sent by a virtual variable (x'_1) which is responsible for enforcing the g-constraint of a single variable (x_1); the virtual variable is controlled by the agent that owns the original variable (lines 3-5). The feedback is sent to the lowest priority variable involved in the violated constraint (x_3). Using the f-cost optimization mechanisms of Adopt, the feedback will propagate to the node(s) that must change their values to find the optimal satisfying solution. Specifically, if a variable (e.g. x_3) receives an infinite cost for all of its domain values, it will send a COST message to its parent (e.g. x_2) indicating an lb and ub of ∞ which will prompt the parent to change its value. Since feedback must be able to percolate up to all the nodes in a constraint, Adopt's DFS tree must be restructured to put all the variables in a g-constraint in the same subtree (lines 6-8). MCA preprocessing creates the virtual variables and places them as leaves, lower in the DFS tree than the variables in the g-constraint they enforce (lines 10-11). We revisit preprocessing of DFS tree in Section 4.2.4.

4.2.2 MCAS

While feedback signals maintain the privacy of the g-constraints and the variables involved by encapsulating the g-constraint in-

side a variable, MCAP's partial search of unsatisfying assignments slows the algorithm. When a g-constraint is non-private, we reveal the g-function of a link to those nodes connected to it and use this information to implement MCAS and MCASA shown in Figures 5b and 5c respectively.

MCAS and MCASA exploit g-function revelation by having nodes send their descendants g-thresholds (line 52) indicating that the descendant may take on no value with a g-cost greater than the g-threshold. In the snapshots from Figures 5b and 5c we can see the g-thresholds being passed down from x_1 to x_2 and x_3 (as discussed earlier, we assume that the g-functions are additive here). If the variable is not a T-node (in MCAS), the g-thresholds for a child (x_l) are an upper bound: the total g-budget minus the minimum g-cost that could be consumed by each of the other links ($G_i - \sum_{x_j \in Neighbors(x_i) \neq x_l} min\ g_{ij}(d_i, d_j)$). In Figure 5b, the total g-budget at x_1 is 4, and each link consumes at least 1, so the g-thresholds are 3. Given this bound, a node can prune values from its domain (e.g. 0 is pruned from the domain of x_3) leading to speedups over MCAP.

4.2.3 MCASA

For T-nodes, it is possible to calculate an exact bound (MCASA) enabling more values to be pruned out and further speeding up the search (e.g. x_1 in 5c now a T-node, calculates exact bounds). Calculating the exact bounds requires a node to maintain a mapping from potential g-thresholds to lower bounds on f-costs (GFmap) for each of its children. GFmap is dynamically initialized for the current value from the link function (line 17-18; line 50) and then updated as COST messages arrive (line 38). A T-node uses these g-threshold to f-cost mappings to calculate how to split its remaining g-budget (its initial g-budget minus the g-cost consumed on each of the links with its higher priority neighbors) among its children. This calcOptimalSplit function can be implemented using a straightforward dynamic program and thus is omitted from the pseudo-code. Since x_i's lb and ub for the current context now depend upon the way x_i splits its g-budget among its children, we store lb and ub based upon x_i's current split. We reset lb and ub to 0 and ∞ respectively whenever x_i changes its split (line 39) as well as when the context changes (line 28). If a node can't satisfy any of its g-thresholds, it will send up a cost of ∞ which will cause its ancestors to switch their values (line 45).

4.2.4 Combining Techniques

Until now we have discussed each technique separately. However, the techniques may be applied simultaneously to different nodes within the same problem. Since the g-constraints are local constraints, we only need worry about situations where a parent and a child are using different techniques: (i) Parent = MCAS/MCASA, Child = MCAP: The child can restrict its domain based upon the g-threshold of its parent and the parent will automatically change its value if the child's virtual variable passes up an infinite cost. (ii) Parent = MCAP, Child = MCAS/MCASA: The child will automatically adjust its value if the parent's virtual variable passes up an infinite cost and the child's technique makes no difference to the parent.

One key issue is whether the tree transformations can turn a T-node somewhere else in the tree into a non-T-node. Two steps can cause links to be added: (i) adding empty links between x_i's children to force them into the same subtree (lines 6-8) and (ii) adding a virtual variable and empty links to each of the variables in the g-constraint it represents (lines 10-11). If a link is added between two children of a T-node, then despite having no f-function or g-function it can turn a T-node into a non-T-node. For instance, in

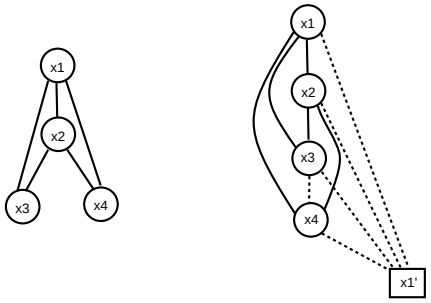


Figure 6: a) original constraint graph b) after adding x'_1

Figure 6, x_1 has a g-constraint with x_2, x_3 and x_4. During preprocessing, a link must be added between x_3 and x_4 to keep them from being in separate subtrees and one must be chosen (arbitrarily) to be the parent of the other (x_3 is made parent of x_4). x_4's COST messages no longer flow directly to x_2 but are sent via x_3 where they are combined with x_3's local costs. This combination prevents x_2 from calculating an exact bound and renders it no longer a T-node.

In contrast, in Figure 4a the addition of virtual variables and their accompanying links, does not change a T-node to a non-T-node. While x_2 is originally a T-node, the addition of the virtual variable x_1' creates a lower priority neighbor for x_2 that is not a child. However, in this case, x'_1 is a virtual variable, i.e. it has no f-functions or g-functions on any of its links and it has no domain of its own. Since x'_1 need be given no g-threshold, the fact that it is not one of x_2's children does not prevent x_2 from being a T-node.

With these two cases in mind, preprocessing (lines 1-11) starts by walking through the tree in priority order adding in the empty links between children where necessary and adjusting the priorities accordingly. Once a node, x_i, has been determined to be a T-node no links added between its lower priority non-neighbors will change x_i into a non-T-node, so one sweep of the tree is sufficient to correctly determine which nodes are T-nodes. After this sweep the virtual variables themselves can be added as leaves without causing any of the previously determined T-nodes to lose that property.

4.3 Correctness and Complexity of MCA

In this section we will again separate out the proofs for each technique for the sake of clarity. As previously described, the interaction of the techniques in the combined algorithm does not change their properties. Recall in the following that a context is the set of variable assignments upon which a piece of information is predicated.

PROPOSITION 1. *For each node x_i for the current context, MCAP finds the assignment whose f-cost, local cost (δ) plus the sum of each of x_i's children's (x_l's) costs, is minimized while satisfying the g-constraint:*

$$OPT(x_i, context) \stackrel{def}{=}$$
$$min_{d \in D_i}[\delta(d) + \sum_{x_l} OPT(x_l, context \cup (x_i, d))]$$
$$\quad if\ g_i(d_i, \{d_j | x_j \in Neighbors(x_i)\}) \leq G_i$$
$$\infty\ otherwise$$
$$where\ \delta(d) \stackrel{def}{=} \sum_{x_j \in ancestors} f_{ij}(d_i, d_j)$$

Proof: To show this we start from the correctness of the original Adopt algorithm [9]. Adopt will find at every node x_i:

$$OPT'(x_i, context) \stackrel{def}{=}$$
$$min_{d \in D_i}[\delta'(d) + \sum_{x_l} OPT'(x_l, context \cup (x_i, d))]$$
$$where\ \delta'(d) \stackrel{def}{=} \sum_{x_j \in ancestors} f_{ij}(d_i, d_j)$$

To show that MCAP finds $OPT(x_i, context)$ we show that 1) it

never returns an assignment containing a violated g-constraint, unless the g-constraint is unsatisfiable and 2) it finds the minimum f-cost solution.

The proof of part 1) starts with the fact that the virtual variables introduce an infinite f-cost into the subtree containing the violated constraint. This infinite f-cost enters lower in the priority tree than any variable in the constraint which allows the normal flow of COST messages to eventually carry it to all of the involved nodes. Since any assignment that does not violate the g-constraint will have a finite f-cost, it follows from the correctness of Adopt that by choosing the assignment that minimizes f, MCAP will never take on an assignment that violates a g-constraint unless the g-constraint is unsatisfiable. Part 2) follows directly from the correctness of Adopt because the virtual variables report a zero cost if all constraints are satisfied, which means that by Adopt's normal mechanisms it will find the minimum f-cost solution. ■

Proving that MCAS is correct requires a minor addition to the MCAP proof from Proposition 1 which is shown below.

PROPOSITION 2. *If the g-constraint for each node x_i is $\sum_{x_j \in Neighbors(x_i)} g_{ij}(d_i, d_j) < G_i$, then no satisfying solution can contain on link l_{il} a g-cost greater than $G_i - \sum_{x_j \in Neighbors(x_i) \neq x_l} min\ g_{ij}(d_i, d_j)$.*

Proof: Each link consumes a certain minimum g-cost, and we are only subtracting the sum of the absolute minimum costs on all links. ■

We next turn to MCASA; if we can prove the g-thresholds are assigned optimally, then taken with original Adopt's correctness it will show that MCASA is correct. After that we will show that when a node is not a T-node then exact bounds do not apply.

PROPOSITION 3. *For each T-node x_i, MCASA always terminates with an optimal division of the g-budget given d_i and the current context.*

Proof by Contradiction: Throughtout the proof we assume $x_k \in Children(x_i)$ and we assume there are no T-nodes as descendents of x_i – using induction we can generalize. Assume MCASA terminates with g-thresholds g'_{ik} $(\forall x_k)$ which are not optimal. Thus there exists another set of g-thresholds (g^*_{ik}) s.t. $\delta(d_i) + \sum_{x_k} min\ lb(d_i, x_k)$ where $d_k \in \{D_k | g_{ik}(d_i, d_k) \leq g_{ik}*\} < \delta(d_i) + \sum_{x_k} min\ lb(d_i, x_k)$ where $d_k \in \{D_k | g_{ik}(d_i, d_k) \leq g'_{ik}\}$. Since local cost $\delta(d)$ is constant for all g-thresholds, we will drop it. To have been selected, g'_{ik} must have seemed optimal based on current information at some point. To distinguish these two states of knowledge we coin the following terms: $f_{actual}(g_{ik}, x_k)$ is the f-cost (specifically $min\ lb(d_i, x_k)$ where $d_k \in \{D_k | g_{ik}(d_i, d_k) \leq g_{ik}\}$) when all costs have percolated up from all descendants and $f_{current}(g_{ik}, x_k)$ is the current lower bound on f-cost. When g'_{ik} is selected:

1. $\sum_{x_k} f_{actual}(g'_{ik}, x_k) > \sum_{x_k} f_{actual}(g^*_{ik}, x_k)$
2. $\sum_{x_k} f_{current}(g'_{ik}, x_k) < \sum_{x_k} f_{current}(g^*_{ik}, x_k)$
3. $f_{current}(g_{ik}, x_k) \leq f_{actual}(g_{ik}, x_k)$ for any g_{ik}
4. $\sum_{x_k} f_{current}(g_{ik}, x_k) \leq \sum_{x_k} f_{actual}(g_{ik}, x_k)$

Since there are a finite number of nodes in the tree below x_i, before termination $\sum_{x_k} f_{current}(g'_{ik}, x_k) = \sum_{x_k} f_{actual}(g'_{ik}, x_k)$ will become true. At this point:

$$\sum_{x_k} f_{current}(g'_{ik}, x_k) = \sum_{x_k} f_{actual}(g'_{ik}, x_k)$$

$$\sum_{x_k} f_{current}(g'_{ik}, x_k) > \sum_{x_k} f_{actual}(g^*_{ik}, x_k) \quad \text{by (1)}$$

$$\sum_{x_k} f_{current}(g'_{ik}, x_k) > \sum_{x_k} f_{current}(g^*_{ik}, x_k) \quad \text{by (4)}$$

Thus based upon current information ($f_{current}$) MCASA will switch from g'_{ik} to $g_{ik}*$ because it has a lower associated f-cost. This contradicts our assumption that MCASA will terminate with g-thresholds g'_{ik}. ■

If x_i is not a T-node, then MCASA is not guaranteed to find the assignment whose f-cost, local cost (δ) plus the sum of x_i's children's costs, is minimized while satisfying the g-constraint. We can show this with a counter-example.

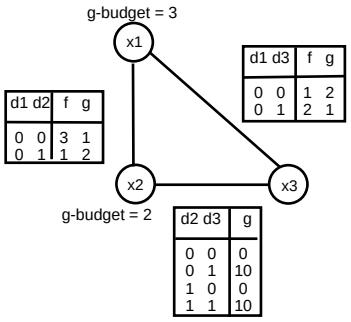


Figure 7: MCASA fails on non-T-nodes

As we can see in Figure 7, x_1 is a non-T-node since x_3 is a child of x_2 not x_1. If we assume that x_1 has a g-budget of 3 and only one value in its domain, then it must choose how to split its g between its two children. Based upon the functions on the links, it will choose to give a g-threshold of 2 to x_2 and 1 to x_3, leading to a predicted f-cost of 2 + 1 = 3. This effectively removes the value 0 from x_3's domain and causes the link between x_2 and x_3 to incur a g-cost of 10 which in turn leads x_2's g-constraint to be unsatisfiable for any g-threshold it could receive from x_1 (Since x_2 tries out both values under given threshold of 2). Since no matter what g-threshold it assigns to x_2, x_2's g-constraint is violated (leading to an infinite f-cost), x_1 infers that the problem is unsatisfiable. It is at this point that condition (3) from the previous proof has been violated since x_1 estimates $f_{current}(gt, x_2) = \infty$ whereas $f_{actual}(gt, x_2) = 4$ where $gt \in \{1, 2, 3\}$. This counter example is based on having just a g-function on the $x_2 - x_3$ link, however, similar examples can be constructed using just an f-function or both an f- and a g-function on the link. Modifications to MCASA to make these cases feasible is an issue for future work. ■

We now turn to MCA complexity. The original DCOP problem is NP-hard. By illustrating a solution to the multiply-constrained problem which only adds a linear number of additional nodes we suggest that the complexity class has not worsened. With respect to space, a key feature of Adopt is that its space complexity is linear in the number of nodes, n, specifically $|D_i|\,n$. In MCAP and MCAS, the space used at each regular node is the same, but we add up to n virtual variables, so the space complexity for MCAP and MCAS is $(|D_i| + 1)n$. In MCASA there are no virtual variables, but each node stores a g-to-f mapping for each of its children. This causes the space complexity to be $|D_i|\,n + G_i n$.

5. EXPERIMENTAL RESULTS

This section presents five sets of experiments. The first compares the performance of MCAP, MCAS and MCASA on four settings that were motivated by the domains in Section 2.3. Setting 1 com-

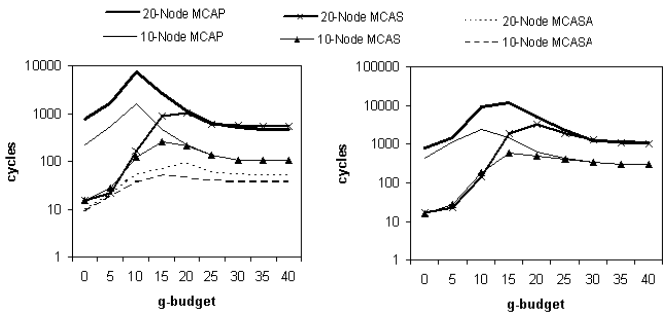


Figure 8: g-budget vs. run-time for a) 100% T-node problems b) 85% T-node problems

prised 20-node problems, with 3 values per node, an average link density of 1.9 and maximum link density of 4. It had 100% T-nodes and both the f- and g-costs were randomly chosen from a uniform distribution varying from 0 to 10. Setting 2 is similar to setting 1, except that the graph was 85% T-node (which increased the average link density to 2.2) to allow for comparison of the impact of T-nodes. Setting 3 (setting 4) is similar to setting 1 (setting 2), except that it is a 10-node problem. We created 15 sets of constraint functions for each of our domain settings, i.e. each data-point in our graphs is an average of 15 problem instances.

To highlight the tradeoff between the techniques, we show the performance of each technique when applied to all the nodes in a problem i.e. we either apply MCAP to all the nodes or MCAS or MCASA. There are a total of 1350 different runs summarized in the two graphs given all the g-budgets tested. Figure 8a shows the average run-times of MCAP, MCAS, MCASA in settings 1 and 3. The x-axis shows the g-budget applied to all variables and ranges from 0, which is unsatisfiable, to 40, which is effectively a singly-constrained problem. The y-axis shows runtime – run-time is measured in cycles where one cycle consists of all agents receiving incoming messages, performing local processing and sending outgoing messages [9]. The y-axis is logarithmically scaled.

The graphs show that due to privacy, MCAP has the poorest performance. The upper bounds calculated by sharing information in MCAS improve performance, while the exact bounds and lack of tree-restructuring in MCASA give it the best performance. The maximum speedup of MCAS over MCAP is 744 for setting 1 and 209 for setting 3, while the maximum speedup of MCASA over MCAS is 937 for setting 1, and 190 for setting 3. Figure 8b demonstrates similar results for setting 2 and setting 4. Only MCAP and MCAS results are shown given the switch to 85% T-node problems in these settings (we cannot apply MCASA to all the nodes in these settings). The switch from 100% T-node to 85% T-nodes causes a significant increase in run-time (note the y-axes are not identical).

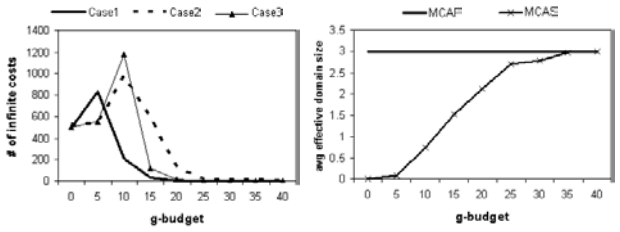


Figure 9: a) g-budget vs. number of infinite cost messages b) g-budget vs. number of values per domain

For all of the subalgorithms the runtime curve has a distinct inverted U-shape to it: the run-times are lowest at high g-budgets (no resource constraints) or at low g-budgets (tight resource constraints). Figure 9a begins to provide an explanation by plotting the g-budget on the x-axis and the total number of infinite cost messages received by any variable in the problem – due to g-constraint violation – on the y-axis. The figure shows results from three representative runs of MCAP on setting 3. We can see the same hump appearing at a g-budget of 10 and diminishing to almost 0 at a g-budget of 20. This agrees with the fact that the maximum g-cost on a link is 10 and the average link density is 1.9 (there are still one or two infinite cost messages all the way up to a g-budget of 35 because the maximum link density is 4). The larger number of infinite cost messages in the mid-g range indicates that it potentially takes longer to discover unsatisfiability of solutions, leading to longer run-times and hence the hump shape.

In all settings, for tighter g-budgets, the MCAS algorithm outperforms MCAP, however, for looser g-budgets, there is no difference in performance. Figure 9b provides an explanation based on the fact that MCAS sends g-thresholds that potentially prune descendents' domain values. We plot the g-budget on the x-axis and the average number of values remaining in the domain of a variable running MCAS on the y-axis. The numbers are plotted as an average over all nodes over all 15 problem instances of setting 3. For comparison, we also provide results of MCAP, which performs no pruning and hence is a flat line. This figure shows that when g-budgets are tight, MCAS provides significant pruning, but when g-budgets are loose MCAS upper-bounds provide no pruning (in comparison with MCAP) and thus there is no efficiency loss due to privacy. The domain sizes converge at a g-budget of 25, which is also where the runtimes converge in Figure 8.

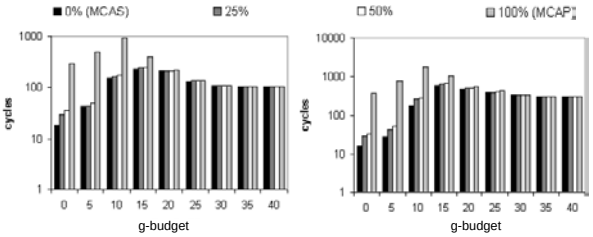


Figure 10: g-budget vs. runtime, varying percentages of private constraints: a) 100% T-node and b) 85% T-node problems

While we have performed several other explorations of privacy-efficiency tradeoffs in our techniques, space limitations preclude an extensive discussion. For example, one set of experiments demonstrate the benefits of the per-node application of the MCAP and MCAS techniques. Here, we took the examples from settings 1 and 2 from Figure 8a and b and randomly assigned first 25% and then 50% of the nodes to have private g-constraints while the remaining were assumed to be non-private. We then compared their performance to that of MCAP (100% private) and MCAS (0% private). In order to isolate the effect of privacy, not graph structure, we did not exploit T-nodes. The results are shown in Figure 10a and b. The x-axis again shows the g-budget applied and the y-axis measures the runtime in cycles on a logarithmic scale. Each bar in the graph shows an average over the 15 instances and we can see that as the percentage of nodes whose additional constraint is private increases, the runtime increases for smaller g-budgets. Interestingly, the increase in run-time is not proportional to the increase in the number of private g-constraints; there is a significant jump in run-time when all nodes have private g-constraints. However, as in Figure 8, when the g-budget on each variable is loose enough the

runtimes converge because no pruning takes place.

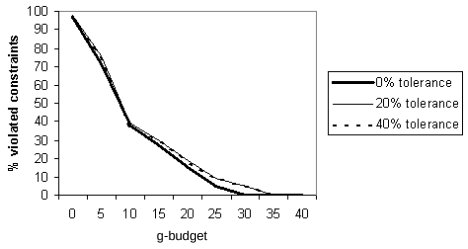


Figure 11: g-constraint violation by Adopt

While we developed MCA, one question was whether singly-constrained algorithms could automatically satisfy the g-constraint simply by loosening their tolerances on optimal solutions. We examined this by measuring the degree to which the g-constraint is violated by Adopt when ignoring g but specifying a tolerance on f. Figure 11 shows the results with each data point representing an average over 5 randomly generated runs. The x-axis gives the g-budget applied to each variable and again varies from 0 to 40. The y-axis is the percentage of nodes whose g-constraint was not satisfied by the solution obtained by Adopt. The tolerance is a percentage of the difference between the optimal and maximum cost solutions. The key result is that even with a large 40% tolerance on f, large numbers of g-constraints are violated with small g-budgets. Thus, running singly-constrained algorithms with tolerances is inadequate in addressing multiply-constrained DCOPs.

6. CONCLUSION AND RELATED WORK

This paper introduces a novel algorithm for multiply-constrained DCOPs, where agents must not only optimize a global objective, but must also satisfy resource constraints within their local neighborhoods. These resource constraints may be defined over an overlapping set of the variables in the original optimization constraints and they may be private constraints. Leading DCOP algorithms of today [9, 11, 14] are unable to fully address multiply-constrained DCOP problems; in particular, they are unable to exploit the interaction between the searches for the global optimal and local satisfaction, or address the privacy-efficiency tradeoffs engendered. To remedy this situation, this paper presents a novel multiply-constrained DCOP algorithm based on mutually-intervening search, which is operationalized via three ideas: (i) transforming the constraint graph to allow private n-ary constraints to be enforced; (ii) revealing upper-bounds on (resource) costs to neighbors, in order to gain efficiency while sacrificing privacy; (iii) identifying a local graph structure property – T-nodes – which allows agents to gain further efficiency by providing exact bounds on resource costs to neighbors. These ideas were realized via the MCA algorithm, which builds on Adopt, currently one of the most efficient DCOP algorithms. We proved the correctness of MCA, and presented detailed experimental results, illustrating the profile of the privacy-efficiency tradeoffs.

In terms of related work, we have already discussed the relationship of multiply-constrained DCOP and MCA with singly-constrained DCOP algorithms, such as Adopt, OptAPO and DPOP [9, 11, 14]. While we build on top of Adopt, we recognize that no DCOP algorithm currently dominates all others across all domains. However, the techniques developed here would transfer to other algorithms, e.g. MCAS and MCASA style techniques could be applied to algorithms like OptAPO. Whether DPOP [11] could similarly benefit from our techniques is an issue for future work.

Also related is previous work in multi-criteria collaboration [7, 10] which looks at finding a pareto-optimal for multiple objectives rather than optimizing a single objective subject to resource constraints. Furthermore, unlike our research, that work does not build on the recent theoretical or algorithmic advances in DCOPs. Approaches to multi-criteria constraint satisfaction and optimization problems have tackled the problem using centralized methods [4], but we focus on a distributed problem, which requires designing algorithms where agents function without global knowledge.

7. ACKNOWLEDGEMENTS

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010. We would like to thank Barbara Grosz, Hyuckchul Jung and Jay Modi for their valuable feedback.

8. REFERENCES

[1] S. Ali, S. Koenig, and M. Tambe. Preprocessing techniques for accelerating the DCOP algorithm ADOPT. In *AAMAS*, 2005.

[2] J. Cox and E. Durfee. A distributed framework for solving the multiagent plan coordination problem. In *AAMAS*, 2005.

[3] J. A. Espinosa and E. Carmel. The impact of time separation on coordination in global software teams: a conceptual foundation. *SOFTWARE PROCESS IMPROVEMENT AND PRACTICE*, 8, 2004.

[4] M. Gavanelli. An algorithm for multi-criteria optimization in CSPs. In *ECAI*, pages 136–140, 2002.

[5] P. Jalote and G. Jain. Assigning tasks in a 24-hour software development model. In *Asia-Pacific Software Engineering Conference (APSEC04)*, 2004.

[6] R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *AAMAS*, 2004.

[7] N. Matsatsinis and P. Delias. A multi-criteria protocol for multi-agent negotiations. *Lecture Notes in Computer Science*, 3025, 2004.

[8] A. Meisels and O. Lavee. Using additional information in discsps search. In *Distributed Constraint Reasoning Workshop (DCR)*, 2004.

[9] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence Journal*, 161:149–180, 2005.

[10] P. Moraitis and A. Tsoukias. A multicriteria approach for distributed planning and negotiation in multiagent systems. In *ICMAS*, 1996.

[11] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, Edinburgh, Scotland, Aug 2005.

[12] M. Silaghi and D. Mitra. Distributed constraint satisfaction and optimization with privacy enforcement. In *(IAT)*, 2004.

[13] K. Stergiou and T. Walsh. Encoding non-binary constraint satisfaction problems. In *AAAI*, 1999.

[14] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10:673–685, 1998.

[15] M. Yokoo, K. Suzuki, and K. Hirayama. Secure distributed constraint satisfaction: Reaching agreement without revealing private information. In *CP*, 2002.

The DynCOAA Algorithm for Dynamic Constraint Optimization Problems

Koenraad Mertens, Tom Holvoet, Yolande Berbers
AgentWise, Distrinet
Department of Computer Science, K.U.Leuven
200A, Celestijnenlaan
3001 Leuven, Belgium
{Koenraad.Mertens, Tom.Holvoet, Yolande.Berbers}@cs.kuleuven.be

ABSTRACT

Numerous problems in software coordination, operations research, manufacturing control and others can be transformed in constraint optimization problems (COPs). Moreover, most practical problems change constantly, requiring algorithms that can handle dynamic problems.

In this paper we present the dynamic constraint optimization ant algorithm (DynCOAA) that can solve dynamic COPs. DynCOAA is specifically designed for dynamic problems, as it is based upon the ant colony optimization (ACO) meta-heuristic that has already proven its merit in other dynamic optimization problems. DynCOAA is a distributed algorithm that is suited for a one-on-one mapping between variables and hosts, but it can effectively accommodate multiple variables per host. We compared our algorithm to two existing algorithms for dynamic constraint optimization: DynAWC and DynDBA. We find that DynCOAA outperforms DynAWC and DynDBA on some problems, while remaining competitive on others.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Design, Performance

Keywords

Dynamic, ACO, Distributed Constraint Optimization

1. INTRODUCTION

Numerous problems in software coordination, operations research, manufacturing control and others can be transformed in constraint satisfaction and constraint optimization problems (CSPs and COPs). In many practical applications there is not one static CSP or COP that has to be solved: practical problems change often. Changes can happen either because additional information is added to the model, or because of a changing nature in the problem that one tries to solve. An example of a changing problem is a GPS navigation system that has to recalculate the route when the car does not follow the proposed route.

When solving a changed version of a CSP or COP, the changed problem is often similar to the old problem: it only differs in a few constraints or variables. While it is possible that a small change in the problem creates a big change in the optimal solution, in most practical cases the optimal solutions of such similar problems do not differ a lot. E.g. in the GPS system when a car goes forward instead of turning left, the initial positions are almost the same and it may suffice to take the next left turn. Changing problems are called dynamic constraint satisfaction/optimization problems (DynCSP/DynCOP).

In this paper, we present a new distributed algorithm for constraint optimization that is specifically designed for dynamic problems. It is based upon the ant colony optimization (ACO) meta-heuristic [1]. In ACO, the environment of the multi-agent system represents the problem that has to be solved and the agents use this environment to give feedback to one another. This makes ACO-based algorithms naturally suited for dynamic problem solving, as changes in the problem are reflected automatically in the environment of the agents. In spite of this inherent suitability for dynamic problem solving, to this date and to the best of our knowledge, the dynamic constraint optimization ant algorithm (DynCOAA) that we present in this paper is the first distributed algorithm for dynamic constraint satisfaction or dynamic constraint optimization that uses the ACO meta-heuristic.

We compare DynCOAA to two other algorithms: the complete algorithm DynAWC [6] and the non-complete algorithm DynDBA [3]. We use two different cases for evaluating the algorithms. When the three algorithms are used to solve artificial graph coloring problems, DynDBA turns out to be the most effective algorithm. But when a real-world problem – scheduling ships that transport liquified natural gas – is tested, our DynCOAA clearly beats the other two algorithms.

In the following section we present our dynamic constraint optimization ant algorithm (DynCOAA) and the algorithms we use for comparison. Section 3 describes the tests we performed and the results we found. Section 4 concludes this paper with an overview of our findings and some pointers for future work.

2. ALGORITHMS

In this section we present the dynamic constraint optimization ant algorithm (DynCOAA). The construction of the graph struc-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

ture and the behavior of the agents for a static problem are described in [4, 5]. When adding or removing variables or values in a dynamic problem, the corresponding nodes and edges have to be added or removed on the appropriate hosts. When adding or removing constraints, references to those constraints must be registered or deregistered with the appropriate nodes. While it is possible to leave all existing pheromone trails unchanged and only to initialize new edges with the default pheromone quantity, better results can be achieved with dynamic ACO algorithms when all pheromone quantities are equalized to a certain degree. We use the Restart-Strategy that they describe in their paper, using a λ_R value of 0.5, which gave the best results in their experiments. This means that all pheromone quantities are set to the average of the previous pheromone quantity and the default pheromone quantity.

The first algorithm we compare DynCOAA to is DynBDA. This is a non-complete algorithm, based on the Distributed Breakout Algorithm (DBA) and first introduced by [3]. DynDBA is said to be a good algorithm for fast-changing dynamic problems.

The second algorithm we use is DynAWC. This is a complete algorithm that uses nogood recording [6]. The processing of nogoods allows it to reuse information that was found in a previous search after a modification to the problem was made.

3. EXPERIMENTAL RESULTS

In this section we discuss the results of the tests we have done to compare DynCOAA with DynAWC and DynDBA. The three algorithms were implemented in C++, using the same constraint processing engine. We compare the algorithms for two different categories of problems. The first category of problems, discussed in Sect. 3.1, is artificial graph coloring problems. In Sect. 3.2 we discuss the results of a real-world problem, in which a schedule for a number of ships that transport liquified gas must be assembled. Section 3.3 gives an interpretation of the obtained results.

3.1 Graph Coloring Tests

For our first series of tests, we use the same problems as can be found in [3], i.e. graph coloring problems with 30 variables, which require a coloring with 3 colors. Graph densities varied over $\{2.0; 2.3; 2.5\}$. These densities represent the three different phases of the phase transition for random 3-coloring graphs (respectively mostly satisfiable; within the transition; and mostly unsatisfiable). The number of changes between successive problems, indicated by δ, varied over $\{2; 4; 6; 8; 12\}$, where 6 changes means that 3 constraints were removed from the problem and 3 constraints added to the problem, so that the overall density remained unchanged. For each combination of graph density and change rate δ, 20 problems were constructed, each consisting of an initial COP and 100 successive COPs.

As proposed in the future work section of [3], we measured the real-time performance of the algorithms: for each successive COP, the algorithms could calculate during 2 seconds. The best solution at each moment in time was stored. This form of measurement provides a more realistic indication of the performance than can be obtained through cycle-based tests.

Table 1 gives an overview of the results of the three algorithms. The table includes the average time (in seconds) that passed before all 20 problems found the first solution, the average cost of that first solution and the average cost of the best solution that was found after 2 seconds.

When looking at the table, an interesting observation can be made. DynCOAA needs about 5 times longer to calculate its first solution than is necessary for DynDBA and DynAWC. This is due to the fact that DynCOAA uses a swarm of 5 agents, who all need

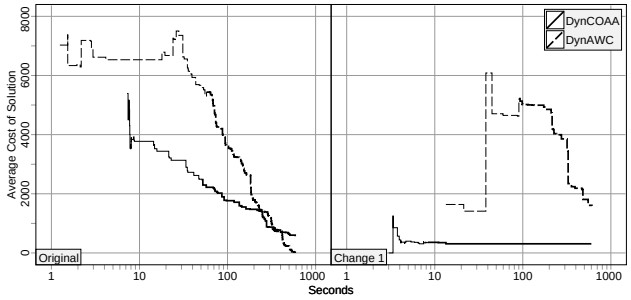


Figure 1: Anytime curves for DynCOAA (solid line) and DynAWC (dashed line) when assembling a schedule for ships. DynDBA was unable to find any solution.

about the same amount of time. But when we look at the final solution of DynCOAA and DynAWC during changes, DynCOAA clearly beats DynAWC, even though it takes about 5 times longer to find its first solution. The result of DynDBA remains better the that of DynCOAA.

As a conclusion, we can say that for these dynamic graph coloring problems, DynDBA is the best algorithm, closely followed by DynCOAA.

3.2 Real-World Ship Scheduling Tests

For our second series of tests, we decided to use a real-world problem instead of an artificial problem. This real-world problem is to assemble a schedule for transport ships. The ships transport liquified natural gas from different ports around the world to one destination port. Using data provided by Tractebel Engineering [2], the algorithms had to calculate a schedule for 7 ships, for a period of three months.

After a decent schedule had been found, we simulated a storm: one of the ships got delayed for three days. The algorithms had to recalculate the schedule, incorporating the delay. The goal was twofold: to obtain a good schedule, but also to leave the individual schedules for the 6 ships that did not get stuck in the storm as unchanged as possible.

This problem is quite complicated: the original scheduling problem is made up of a total of 69 variables and 101 constraints. The variable domains contain up to 200 possible values. 76 constraints are hard constraints (those have to be obeyed in each valid solution: it are constraints with a weight of infinity) and 25 are soft constraints, used for optimizing the cost function. 57 of the hard constraints are binary constraints. All other hard constraints involve at least 19 variables. The soft constraints even involve all variables. A solution with a cost of 1000 means that the local transport rate in the destination port has to deviate from the optimal rate during a total of one day (in smaller intervals distributed over the three months). At a cost of 2000, it has to deviate during a total of 2 days, etc.

Each of the three algorithms made 20 schedules. For the original schedule as well as for the storm-adaptation, 10 minutes of calculation were allowed. The parameters that were used for the DynCOAA are the same as described in Sect. 3.1, except for the swarm size, which was increased to 30. The anytime curves for this problem are shown in Fig. 1.

The first thing to notice is that DynDBA is missing from Fig. 1. During the 10 minutes of calculation time, none of the 20 runs were able to find any valid schedule (one that satisfies all hard constraints). The most probable reason for this is that a number of the constraints are global constraints, making it very hard for a lo-

Density	δ	DynCOAA			DynAWC			DynDBA		
		First Solution		Best Solution	First Solution		Best Solution	First Solution		Best Solution
		Time	Cost	Cost	Time	Cost	Cost	Time	Cost	Cost
2.0	2	0.13	3.99	2.42	0.053	19.24	8.48	0.034	1.90	1.40
	4	0.13	4.85	3.00	0.044	19.86	9.29	0.034	2.65	1.87
	6	0.14	5.47	3.37	0.044	20.70	9.84	0.032	3.35	2.23
	8	0.13	6.27	4.02	0.039	21.05	9.71	0.034	3.69	2.25
	12	0.13	7.35	4.79	0.054	22.08	10.61	0.034	4.51	2.48
2.3	2	0.13	5.43	3.67	0.055	22.65	12.23	0.025	3.07	2.60
	4	0.14	6.21	4.25	0.056	23.36	12.20	0.028	3.61	2.84
	6	0.14	6.73	5.49	0.060	24.09	12.89	0.026	4.33	3.23
	8	0.13	7.51	5.05	0.049	24.62	13.29	0.025	4.77	3.37
	12	0.13	8.90	6.18	0.047	25.42	13.64	0.027	5.60	3.59
2.5	2	0.13	6.42	4.53	0.057	25.76	14.70	0.018	4.07	3.59
	4	0.15	7.43	5.26	0.058	26.13	15.06	0.028	4.88	4.09
	6	0.17	8.14	5.87	0.057	26.85	15.63	0.027	5.45	4.33
	8	0.14	8.71	6.21	0.043	27.50	15.76	0.036	5.87	4.46
	12	0.14	10.01	7.25	0.057	28.79	16.59	0.028	6.70	4.70

Table 1: Results of testing DynCOAA, DynAWC and DynDBA on 30-node, dynamic 3-coloring problems. Averages over 20 problem instances per Density-δ combination. Time is expressed in seconds.

cal search algorithm to find a solution. Because no initial schedule was found, DynDBA could not even begin to search for a solution after the change.

The two algorithms that did find a valid solution did so in all 20 runs. But the behaviors of DynCOAA and DynAWC were completely different for the original and the changed problem. For the original schedule, it took both algorithms about 1 minute to find a first solution for all 20 runs. The average solution of DynCOAA was twice as good as the average solution of DynAWC at that time. The average solution of DynAWC quickly became better and after 10 minutes DynAWC had found an almost perfect solution in all 20 runs, while DynCOAA still had a cost of about 550.

As for the schedule after the storm, it took DynCOAA less than 10 seconds to find a valid schedule in all 20 runs. Even more, the first average solution was almost as good or even better than the last average solution before the storm. DynCOAA could further improve this first schedule to end at an average cost of about 300 after 10 minutes of calculation. DynAWC on the other hand took almost 100 seconds to find a first solution in all 20 runs. The first average solution had a cost of about 5000. In the remaining time, the solution could be improved to less than 2000, but this still was a lot higher than the average solution of DynCOAA.

One of the reasons of the slower response rate of DynAWC is the preprocessing of the nogoods. After 10 minutes of calculation for the initial schedule, more than 10.000 nogoods had been accumulated. It took DynAWC about 10 seconds to filter out the nogoods that had become invalid after the change. By that time, DynCOAA had already found a new solution.

3.3 Evaluation

The results of DynDBA are the most dissimilar when we look at the behavior over the two different categories of tests. For the artificial graph coloring problems, it was the clear winner of the three algorithms. For the real-world ship scheduling problems, it could not find any solution.

DynCOAA and DynAWC on the other hand behave rather similar for the two different categories of problems. For the first, static problem, DynAWC is a better choice. But when the static problem becomes a dynamic problem by changing some constraints, DynCOAA reacts faster to those changes and finds a better solution.

4. CONCLUSIONS AND FUTURE WORK

In this paper we presented DynCOAA. It is based on the ACO meta-heuristic and designed from the start for solving dynamic constraint optimization problems (DynCOPs). We compared this algorithm against two other algorithms in two different categories of problems: artificial graph coloring problems and real-world ship scheduling problems.

There are two important conclusions that can be drawn from this comparison: (1) DynCOAA is suited for solving DynCOPs, as it beats DynAWC in all two and DynDBA in one of the two categories of problems, and (2) Choosing the right algorithm for a specific problem is very important, because the performance of algorithms can depend greatly on the type of problem.

This work has been funded by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen)

5. REFERENCES

[1] M. Dorigo, V. Maniezzo, and A. Colorni. Positive Feedback as a Search Strategy, Technical Report 91016, Dipartimento di Elettronica e Informatica, Politecnico di Milano, Italy, 1991.

[2] T. Engineering. http://www.engineering.tractebel.com/.

[3] R. Mailler. Comparing two approaches to dynamic, distributed constraint satisfaction. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '05)*, pages 1049–1056, Utrecht, The Nederlands, July 25-29 2005.

[4] K. Mertens and T. Holvoet. CSAA; a Constraint Satisfaction Ant Algorithm Framework. In *Proceedings of the Sixth International Conference on Adaptive Computing in Design and Manufacture (ACDM'04)*, pages 285–294. Springer-Verlag, 2004.

[5] K. Mertens and T. Holvoet. CSAA; a Distributed Ant Algorithm Framework for Constraint Satisfaction. In *Proceedings of the 17th International FLAIRS Conference*, pages 764–769. AAAI Press, 2004.

[6] T. Schiex and G. Verfaillie. Nogood Recording for Static and Dynamic Constraint Satisfaction Problem. *International Journal of Artificial Intelligence Tools*, 3(2):187–207, 1994.

Experimental analysis of privacy loss in DCOP algorithms

Rachel Greenstadt*
Harvard University
greenie@eecs.harvard.edu

Jonathan P. Pearce, Emma Bowring and Milind Tambe†
University of Southern California
{jppearce,bowring,tambe}@usc.edu

ABSTRACT

Distributed Constraint Optimization (DCOP) is rapidly emerging as a prominent technique for multiagent coordination. Unfortunately, rigorous quantitative evaluations of privacy loss in DCOP algorithms have been lacking despite the fact that agent privacy is a key motivation for applying DCOPs in many applications. Recently, Maheswaran et al. [3, 4] introduced a framework for quantitative evaluations of privacy in DCOP algorithms, showing that early DCOP algorithms lose more privacy than purely centralized approaches and questioning the motivation for applying DCOPs. Do state-of-the art DCOP algorithms suffer from a similar shortcoming? This paper answers that question by investigating the most efficient DCOP algorithms, including both DPOP and ADOPT.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms, Performance, Security

Keywords

constraint reasoning, DCOP, privacy

1. INTRODUCTION

Promising approaches in distributed constraint optimization (DCOP) [5, 7], enable distributed conflict resolution and coordination while maintaining users' privacy. Indeed, maintaining privacy is a fundamental motivation in DCOP [5, 7, 9]. One approach to privacy in DCOP is to use cryptographic techniques [11] that ensure watertight privacy but require the use of external servers or computationally intensive cryptographic operations. Instead, we focus on an approach in which researchers provide metrics for quantifying the privacy loss in DCOP algorithms [1,3,4,6,10]. If we can bound privacy loss in specific DCOP algorithms, then cryptographic techniques may be avoidable in situations where they are impractical.

There are three key weaknesses in the previous work on privacy loss analysis in DCOP. First, recent cross-algorithm privacy loss analysis focused on a limited number of DCOP algorithms but indicated that these algorithms preserve less privacy than a centralized approach [3], seriously undermining a key motivation for these algorithms. Thus, it is crucial to analyze some of the most used and most recent DCOP algorithms to see whether they are similarly undermined and to measure their cross-algorithm performance. Two notable omissions in previous analysis are ADOPT [7] and DPOP [8], both among the most efficient DCOP algorithms.

This paper overturns the significant negative results from [3, 4] by providing positive privacy results for several DCOP algorithms not considered in [3, 4]. This paper analyzes ADOPT, DPOP and SynchID [7], three recent DCOP algorithms, via a large-scale experimental investigation of privacy loss in DCOP algorithms in the VPS (Valuations of Possible States) analysis framework [3, 4], using several distributed meeting scheduling scenarios.

A predecessor to the recently introduced algroithms above, SynchBB [2] is an early algorithm for DCOP. Previous work has provided a comparison of privacy loss of a centralized approach with SynchBB, suggesting that the centralized approach may lead to lower privacy loss. Hence, this paper focuses on the remaining algorithms above. These algorithms were chosen because they present novel design choices, or occupy a prominent place in the algorithmic space. The following describes key characteristics of these algorithms:

Adopt is an asynchronous complete DCOP algorithm, guaranteed to find the optimal solution. In Adopt, an agent communicates only one value from its domain at a time, or one message indicating the cost of an assignment to a set of variables at a time.

SynchID is an iterative deepening algorithm similar to Adopt, with two primary differences: agents are organized into a linear chain, rather than a tree, and messages are sent synchronously.

DPOP [8] is an synchronous complete DCOP algorithm, using a tree topology. DPOP is a variable elimination algorithm, where all relevant information is sent up the tree in one large message.

SynchBB or synchronous branch-and-bound, was studied in [4]. However, we focus on a slightly modified SynchBB where information irrelevant to the problem is not communicated.

2. EXPERIMENTAL METHODOLOGY

We focus our investigation on privacy loss in the distributed meeting scheduling problem, since this domain presents inherent privacy concerns [1, 4]. However, the results of this work can be generalized to other DCOP settings where privacy matters.

We define a meeting/event scheduling problem based on the for-

*Supported by a U.S. Department of Homeland Security (DHS) Fellowship, a program administered by the Oak Ridge Institute for Science and Education (ORISE). ORISE is managed by Oak Ridge Associated Universities under DOE contract number DE-AC05-00OR22750.
†This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

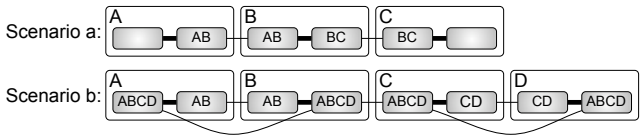


Figure 1: Scenarios: Transparent boxes represent agents and the dark, inner boxes are meeting variables. Thick lines are intra-agent contraints and thin lines are inter-agent constraints.

malism of [5], expressed using the PEAV-DCOP representation [5], which is motivated by privacy considerations.

Scenarios in PEAV-DCOP: The majority of scheduling instances in a functional personal assistant agent system will consist of a small number of meetings that need to be negotiated simultaneously. While larger-scale problems may present themselves, if privacy is a critical factor, the coordination protocols must be effective for these small-scale instances. We consider two scenarios of three (A, B, C) or four (A, B, C, D) agents. The PEAV-DCOP graphs in Figure 1 show the events, labeled by their attendees, and decomposed into variables and constraints.

2.1 VPS: Measuring Privacy Loss

The Valuation of Possible States (VPS) framework [4] was proposed to quantitatively evaluate privacy loss among a group of agents $R_1 \cdots R_n$. Quantification of privacy loss in VPS is based on a valuation on the other agents' estimates about (i.e. a probability distribution over) an agent's possible states. There are three key elements in VPS: (i) agent R_n's private information, modeled as a state $s_n \in S_n$, where S_n is a set of possible states that R_n may occupy; (ii) other agents' estimates about agent R_n's possible states, expressed as a probability distribution $\mathbb{P}_n((S_n)^{N-1})$, (iii) the utility that agent R_n derives from the distribution of other agents' beliefs about R_n's states, yielding value function $\mathbb{V}_n(\mathbb{P}_n((S_n)^{N-1}))$. Note that $\mathbb{P}_n((S_n)^{N-1}) = [\mathbb{P}_n^1(S_n)\ \mathbb{P}_n^2(S_n)\ \cdots \mathbb{P}_n^N(S_n)]$, where $\mathbb{P}_n^j(S_n)$ provides agent R_j's probability distribution over states of agent R_n.

Before negotiation, each agent knows only that the other agents exist in one of $|\mathcal{V}|^T$ possible states. Afterwards, each agent will be modeled by all other agents whose estimate of the observed agent is captured by $\mathbb{P}_n((S_n)^{N-1})$. In this analysis, we focus on the information-theoretic analysis introduced in [1] or the EntropyTS metric [3]. Privacy loss between two agents is studied on a per-timeslot basis, averaged over all timeslots.

We can scale this function and average the privacy loss between all pairs of agents such that the valuations span [0 1] with zero is no loss and one is complete loss of privacy.

2.2 Inference Algorithms

Based on the VPS framework, we define a process by which agents can infer information about other agents while running various DCOP algorithms, in order to measure the likely privacy loss between agents in a DCOP. All inference experiments for all algorithms (including the centralized method) start with the same initial assumptions. We assume that the constraint graph and the valuation of each *meeting* is known to all agents, but the valuations of *time slots* are private. These assumptions are exactly as in [3, 4], allowing comparison of the results; in addition, for the scenarios with few meetings, it is reasonable to assume that the valuations for meetings are public knowledge. Based on these assumptions, we developed the following methods for agent inference for SynchID, Adopt and DPOP.

Centralized: In a centralized algorithm, the agents all send their valuation information to one agent, who computes the result and returns. In every case the centralized agent can "infer" the valuations perfectly [4]. Since we express our results as an average of each agent's privacy loss, the privacy loss of the centralized algorithm is $\frac{1}{N}$ where N is the total number of agents.

SynchID: SynchID is a synchronous algorithm in which agents are ordered in a chain, and messages are passed up and down the chain. An upward message from agent R_n contains a number m_n, which is equal to the best currently known total reward for the subchain of agents under and including R_n. For PEAV, the total reward for the chain is equal to the sum of differences between the valuation of a scheduled meeting and the valuation of the time slot it occupies for every scheduled meeting for every person. We henceforth use $\Delta_{R_n}^{E_k}(t) = V_n^k - V_n^0(t)$ to denote the change in utility to the nth agent for scheduling the kth event at time t. When agent R_n receives an upward message it knows that m_n = a sum of Δ terms lower in the chain from R_n.

To illustrate how possible states can be eliminated in SynchID, we outline the inferences that one can make from messages received in Scenario a. In SynchID, upward messages to agent R_n contained information of the form:

$$m_n = \sum \Delta_{R_n}^{E_k}(t_{E_k}) + \sum \Delta_{R_n}^{E_k}(\tilde{t}_{E_k}), \quad (1)$$

where the summations include events downstream from R_n. t_{E_k} is the time of an event E_k when that time is known to R_n (because R_n is a participant in event E_k), and $\tilde{t}_{E_k}$ is the time of an event E_k when that time is not known to R_n. For example, since B knows when meeting BC is scheduled, as well as the value of meeting BC, a message from C to B (m_B) allows B to know $V_C(t_{BC})$ (the valuation vector component of C at the time at which meeting BC is scheduled). Similarly, a message from B to A (m_A) allows A to know $v_B(t_{AB}) + v_B(\tilde{t}_{BC}) + v_C(\tilde{t}_{BC})$, where $\tilde{t}_{BC}$ is some time not equal to t_{AB}, but otherwise unknown to A. Each of these relations allows the observing agent to reduce the number of possible states the other agents could be in. We obtain the privacy loss for SynchID by allowing each agent to collect these relations, iterate over them, and test each relation against a list of possible states for the other agents, discarding states that conflict with any of the relations.

Adopt: Adopt contains the same type of upward messages as in SynchID, but, due to its asynchrony, it may be impossible for agents to tell how many Δs are contained in the reward component of each message. When a message is received, we know it contains rewards for at least one agent more than the previous message it sent. However, due to asynchrony, our agent might have included more descendants in the message. So, for our inference, we use a $\leq$ sign. The inference equation is:

$$m_n \leq \sum \Delta_{R_n}^{E_k}(t_{E_k}) + \sum \Delta_{R_n}^{E_k}(\tilde{t}_{E_k}), \quad (2)$$

This relation changes to an equality in the special case when only one agent is downstream from agent R_n.

DPOP: In the DPOP algorithm, each agent sends exactly one cost message to its parent. This message is a table of all possible assignments of constrained upstream events and the aggregate costs of those assignments to the agents downstream of R_n. Each entry in the table is used to create inference rules as in equation 1. The events in the entry are the $\Delta_{R_n}^{E_k} t_{E_k}$ terms and other events with participating agents downstream of R_n are the $\Delta_{R_n}^{E_k} \tilde{t}_{E_k}$ terms.

SynchBB: Inference rules for SynchBB are as described in [4].

3. EXPERIMENTAL RESULTS

In this section, we present experimental results from two scenarios. We compare privacy loss in the studied algorithms according to the EntropyTS metric and we introduce a new metric to highlight privacy benefits of distribution over centralized approaches.

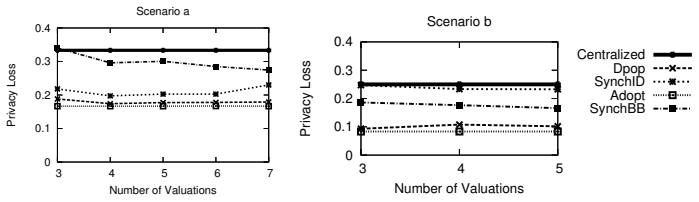


Figure 2: Privacy loss comparisons of the different algorithms.

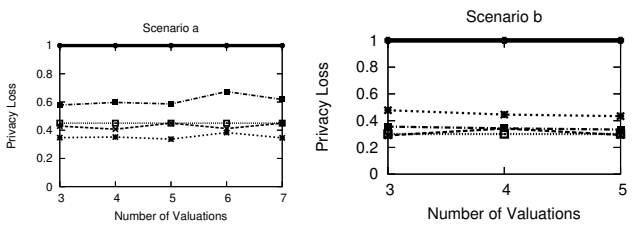


Figure 3: Algorithms compared using the MAX metric.

For the three agent scenario, we varied V, the number of valuations, from 3 to 7 while holding the number of timeslots = 3. For the four agent scenario, for reasons of computational complexity, we varied $|\mathcal{V}|$ from 3 to 5 while holding $T = 3$. For each $(T, |\mathcal{V}|)$ pair, we performed 10 runs per algorithm. For each run, the privacy loss for each agent was measured using the inference algorithms given in Section 3. The systemwide privacy loss was expressed as the arithmetic mean of each agent's privacy loss: $\sum_N \mathbb{V}_n/N$.

In each of our graphs, each data point is an average of 10 runs, and we provide statistical significance results to support our main conclusions. We use a chain topology for all algorithms, to allow fair comparison, since not all algorithms can use a DFS tree.

Cross-algorithm comparison: Figure 2 shows the comparison of privacy loss for the four algorithms mentioned above, for two scenarios, as well as providing a comparison of privacy loss with the centralized approach. The *x*-axis plots the different number of valuations (with number of time-slots fixed at 3) and the *y*-axis plots privacy loss. The thick horizontal line shows the centralized approach, for scenario a (three agents), its privacy loss is 0.33, but for scenario b (four agents) it is 0.25. We use the EntropyTS metric as the metric for privacy loss in this result. We obtained results for other metrics, and mostly these metrics agree with the conclusions drawn using the EntropyTS metric.

We conclude the following from Figure 2: (1) All algorithms but SynchBB have a privacy loss that is lower than the centralized approach. In contrast with the negative results presented in [4], which illustrated DCOP algorithms as having worse privacy loss than a centralized approach, this is a significant positive result. Indeed, the privacy loss in Adopt and DPOP is less than half that of the centralized approach. Furthermore, statistical tests show that Adopt performs better than centralized in all scenarios and DPOP performs better than centralized in all but one (significance level of 5%). (2) DPOP and Adopt had similar privacy loss, despite their vastly different approaches. Despite DPOP's one-shot communication of all information, it performed surprisingly well in terms of privacy loss. Adopt does perform slightly better than DPOP for privacy loss, but not to the level anticipated. (3) Adopt significantly outperformed SynchID in terms of privacy protection. The asynchrony in Adopt was expected to be significantly detrimental to privacy due to the increased numbers of messages. Instead, we found that the uncertainty introduced by asynchrony as to which agents participate in each cost message provides significant privacy gains compared to synchronous algorithms such as SynchID. (4) Despite modifications to improve privacy, SynchBB still performed the worst in terms of its privacy loss; often worse than centralized. The key reason for SynchBB's low performance is its bi-directional messaging of cost information. Thus, it is important to avoid bi-directional cost propagation in DCOP algorithms when privacy is a goal.

MAX metric: In Figure 2, we measured the loss of privacy between pairs of agents, averaged to find the systemwide privacy loss. The effect of one agent learning more than others, and gaining an asymmetric advantage over them, is not considered. The MAX metric addresses this effect by considering only the single agent that learns the most information about other agents (by EntropyTS), rather than the mean of the individual privacy loss figures. Thus, a centralized algorithm will have a value of 1, since the MAX agent learns all the information. Figure 3 shows the results for all the algorithms according to the MAX metric. The number of valuations is plotted on the *x*-axis and the privacy loss is plotted on the *y*-axis.

The MAX metric suggests that there is always a privacy benefit obtained by using DCOP algorithms, even those that perform worse than centralized by our other metrics, when the major privacy concern is one agent accumulating excessive knowledge. In the MAX metric, DPOP and Adopt tended to outperform SynchBB, while SynchID varied widely from scenario to scenario.

4. CONCLUSION

Distributed Constraint Optimization (DCOP) is rapidly emerging as a tool for multiagent coordination. Previous work [4] showed a negative result on privacy loss in early DCOP algorithms, casting doubt on the efficacy of DCOP in privacy requiring domains. This paper presents an investigation of several leading algorithms, including ADOPT and DPOP, and overturns earlier negative results. Our metrics show DCOP algorithms outperforming a centralized approach, with DPOP, SynchID and ADOPT performing best.

5. REFERENCES

[1] M. S. Franzin, F.Rossi, E. C. Freuder, and R. Wallace. Multi-agent meeting scheduling with preferences: efficiency, privacy loss, and solution quality. *Computational Intelligence*, 20(2), 2004.

[2] K. Hirayama and M. Yokoo. Distributed partial constraint satisfaction problem. In G. Smolka, editor, *Principles and Practice of Constraint Programming*, 1997.

[3] R. T. Maheswaran, J. P. Pearce, E. Bowring, P. Varakantham, and M. Tambe. Privacy loss in distributed constraint reasoning: A quantitative framework for analysis and its applications. *JAAMAS*, 2006.

[4] R. T. Maheswaran, J. P. Pearce, P. Varakantham, E. Bowring, and M. Tambe. Valuations of possible states (VPS):a quantitative framework for analysis of privacy loss among collaborative personal assistant agents. In *AAMAS*, 2005.

[5] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham. Taking DCOP to the real world: efficient complete solutions for distributed multi-event scheduling. In *AAMAS*, 2004.

[6] A. Meisels and O. Lavee. Using additional information in DisCSPs search. In *DCR*, 2004.

[7] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence Journal*, 161:149–180, 2005.

[8] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, 2005.

[9] M. C. Silaghi and B. Faltings. A comparison of distributed constraint satisfaction approaches with respect to privacy. In *DCR*, 2002.

[10] M.C. Silaghi. Meeting scheduling guaranteeing n/2-privacy and resistant to statistical analysis (applicable to any discsp). In *3rd IC on Web Intelligence*, 2004.

[11] M. Yokoo, K. Suzuki, and K. Hirayama. Secure distributed constraint satisfaction: Reaching agreement without revealing private information. In *CP 2002*, Berlin, 2002.

No-Commitment Branch and Bound Search for Distributed Constraint Optimization

Anton Chechetka
Robotics Institute, Carnegie Mellon University
5000 Forbes Ave, Pittsburgh PA 15213, USA
antonc@cs.cmu.edu

Katia Sycara
Robotics Institute, Carnegie Mellon University
5000 Forbes Ave, Pittsburgh PA 15213, USA
katia+@cs.cmu.edu

ABSTRACT

We present a new polynomial-space algorithm for solving Distributed Constraint Optimization problems (DCOP). The algorithm, called NCBB, is branch and bound search with modifications for efficiency in a multiagent setting. Two main features of the algorithm are: (a) using different agents to search non-intersecting parts of a search space concurrently, and (b) communicating lower bounds on solution cost every time there is a possibility the bounds might change due to changed variable assignments. The first leads to a better utilization of computational resources of multiple participating agents, while the second provides for more efficient pruning of search space.

Experimental results show that NCBB has significantly better performance than another polynomial-space algorithm, ADOPT, on random graph coloring problems. Under assumptions of cheap communication it also has comparable performance with DPOP despite using only polynomial memory as opposed to exponential memory for DPOP.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Coherence and Coordination*

General Terms

Algorithms

Keywords

Cooperative distributed problem solving in agent systems

1. INTRODUCTION

In recent years Distributed Constraint Optimization (DCOP) [2] has received considerable attention as a powerful paradigm for multiagent coordination. Example applications include target tracking in distributed sensor networks [7] and distributed event scheduling [1].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Several efficient distributed algorithms for solving DCOP have been developed recently, both optimal (Synchronous Branch and Bound, ADOPT [7], OptAPO [5], DPOP [8]) and approximate (Iterative Distributed Breakout). Of the optimal algorithms ADOPT, DPOP and OptAPO have the best performance to date [5, 7, 8]. However, more work is needed to make optimal DCOP algorithms efficient enough for real-life problems.

In this paper we present a decentralized algorithm, called No-Commitment Branch and Bound search (NCBB), for solving DCOP, that addresses the problem of performance. It is based on well-known branch and bound search [3]. It runs several concurrent search processes in different partitions of search space. This enables one to speed up the search due to parallelization and also reduces the amount of information that has to be synchronized among agents. We demonstrate that NCBB performs better than ADOPT on distributed graph coloring problems under a variety of metrics. When communication is cheap, it also has comparable performance with DPOP despite using only polynomial memory as opposed to exponential memory for DPOP.

2. DCOP

A Distributed Constraint Optimization problem consists of a set of variables $V = \{x_1, \ldots, x_n\}$, a set of discrete finite domains for each of the variables $D = \{D_1, \ldots, D_n\}$ and a set of constraints $F = \{f_1, \ldots, f_m\}$ where each constraint is a function $f_i : D_{i_1}, \ldots, D_{i_j} \rightarrow \mathcal{N} \cup \infty$. Each variable is controlled by an agent that can communicate with other agents. The goal of the agents is to find an assignment for all the variables $B^* = \{d_1, \ldots, d_n | d_i \in D_i\}$ such that the global cost, $\sum_{i=1}^{m} f_i$, is minimized. DCOP is NP-complete.

In this article we consider a restricted version of DCOP. We assume all the constraints to depend on exactly two variables, although our approach can in principle be extended to k-ary constraints. We also assume that each variable is controlled by an independent agent and use the terms *agent* and *variable* interchangeably.

We will use the term *constraint graph* to refer to a graph where variables are nodes and two nodes have an edge between them if and only if they share a constraint. We also assume that two agents have a direct communicational link if and only if they share a constraint.

3. NCBB ALGORITHM

Before the algorithm can be executed, the agents need to be prioritized in a depth-first search (DFS) tree, in which

```
function mainLoop()
1  if (¬ IAmRoot) updateContext();
2  while (true)
3    search();
4    if (IAmRoot ∨ updateContext()) break;
5  end while;
6  costs[resultValue] = 0;
7  (for ∀ y ∈ chldren) subtreeSearch(resultValue, y);
8  (for ∀ y ∈ children) send "STOP" to y;
end mainLoop;

function updateContext()
10 while (true)
11   receive message m from y ∈ ancestors;
12   if (received "search") bound ← m.BOUND; return false;
13   else if (received "y=d")
14     context[y] ← d;
15     send LB(x,context,indx_y) - LB(x,context,indx_y-1) to y;
16   else if (received "STOP") return true;
end if; end while; end updateContext;
```

Figure 1: Main loop for agent x

each agent has a single parent and multiple children. Constraints are only allowed between agents that are in an ancestor-descendant relationship in the DFS tree. An agent does not have information about all its ancestors or descendants, but only about those with whom it shares a constraint.

The purpose of such an ordering is to decompose the global cost function: given the assignments to all ancestors, agents in a given subtree can work on minimizing their part of the solution cost independently of agents in other subtrees. Any constraint graph can be ordered into some DFS tree using distributed algorithm from [4].

3.1 Auxiliary definitions

In further explanation all variables names refer to local variables of the considered agent.

DEFINITION 1. *Agent cost for agent x and assignment B*

$$AgentCost(x, B) \equiv \sum_{y \in ancestors_x} f_{x,y}(B(x), B(y))$$

where $B(x)$ is the value of x under assignment B.

DEFINITION 2. *For agent x, assignment B and integer* $k \in 0..|ancestors_x|$ *denote LB(x,B,k)*

$$LB(x, B, k) = \min_{d_x \in D_x} \min_{B_1 \in B|_{ancestors_x[1..k]}} AgentCost(d_x, B)$$

where $B|_{vars}$ is the set of all assignments that agree with B on the values of variables in $vars$.

3.2 Initialization

During the initialization stage agents compute global upper and lower bounds on the solution cost. Each agent chooses its value greedily given values of its ancestors so as to minimize its *AgentCost*. The costs are propagated up the tree and `bound` variable of the root agent takes the value

$$bound = \sum_{x \in V} AgentCost(x, B_{greedy}) - \sum_{x \in V} LB(x, \emptyset, 0).$$

`bound` variable of an agent plays the role of an upper bound on the cost of this agent's subtree. Thus one can see that NCBB uses nontrivial lower bounds on the solution cost during the search. This idea is closely related to preprocessing

```
function search()
20 idle ← children;
21 costs ← ∅; unexplored ← ∅; anncdVals ← ∅;
22 minCost ← LB(x, context, |ancestors|);
23 for ∀ d ∈ D : AgentCost(d,context) ≤bound + minCost
24   costs[d] ← AgentCost(d,context) - minCost;
25 end for;
26 for (∀ d ∈ D : costs[d] ≠ ∅) unexplored[d] ← children;
27 while (unexplored ≢ ∅ ∨ anncdVals ≢ ∅)
28   while idle ≢ ∅
29     choose child ∈ idle; idle ← (idle \ child);
30     choose d : child ∈ unexplored[d];
31     unexplored[d] ← (unexplored[d] \ child);
32     if (¬ subtreeSearch(d, child) ∧
33             (∃ c ∈ D: child ∈ unexplored[c]))
34       idle ← (idle ∪ child);
35   end if; end while;
36   if anncdVals ≢ ∅
37     receive cost_y from y ∈ children;
38     d ← anncdVals[y]; anncdVals[y] ← ∅;
39     costs[d] ← costs[d] + cost_y;
40     if (costs[d] > bound) prune();
41     else if (unexplored[d] ≡ ∅) ∧ (y ∉ ∪_{c∈D} anncdVals[c])
42       bound=costs[d]; resultValue = d; prune();
43   end if; end if;
44   (if ∃ c ∈ D: child ∈ unexplored[c]) idle ← (idle ∪ child);
45 end while;
46 if (¬ IAmRoot) send min_{d∈D} costs[d] to parent; end if;
end search;

function subtreeSearch(d, child)
50 send "x=d" to all descendants[child]; anncdVals[child]=d;
51 for ∀ y ∈ descendants[child]
52   receive cost_y from y; costs[d] ← costs[d] + cost_y; end for;
53 if (costs[d] > bound) prune(); anncdVals[chld]=∅;
54   return false;
55 else
56   send "SEARCH, BOUND=bound-costs[d]" to child;
57   return true;
end if; end subtreeSearch;
```

Figure 2: Main search loop subroutines for agent x

heuristics for lower bound estimation for ADOPT [1] that resulted in speedups by an order of magnitude.

3.3 Main loop

After initialization all agents execute the main search loop listed in Figures 1 and 2. An agent x does not start actively searching until it receives a $SEARCH$ message from its parent (line 12). Before this message is sent by x's parent, all x's ancestors select their values and announce them to their descendants. Therefore, when x receives a message to start searching, it has consistent knowledge about the relevant ancestors' values.

The `search` function contains two most distinct features of the algorithm. One of them is the method of branching: when x selects the next value to explore on a given subtree, it may choose a value different from those announced to x's other subtrees (30). For each value an already known part of the total cost is stored and updated (see `costs` variable). Because a particular value is generally used at different times for different subtrees, one can take into account the

costs for already completed subtree searches and compute a tighter upper bound on the solution cost for the next subtree (`subtreeSearch`, (52, 56)). It can be viewed as running simultaneous search processes in different parts of search space, and exploits inherent parallelism of the problem.

The second major feature of NCBB is eager propagation of cost lower bound changes. As agent x learns about its ancestors' assignments, it can compute $LB(x, context_x, k)$ for larger and larger k. Every time x receives a message with the value of its ancestor y, it immediately sends back the amount of change in lower bound on x's *AgentCost* (15). This leads to more precise lower bounds on partial assignment optimal cost at the point of control (agent y in this case) and in turn to better pruning.

An agent keeps track of the solution costs known so far for each one of its values (`costs` map), which subtrees have not yet explored which of its values (`unexplored`) and what value is currently used in what subtree (`anncdVals`). It updates this information as the subtrees return their search results (lines 39,52).

4. ALGORITHM PROPERTIES

PROPERTY 1. *NCBB always terminates after finite time.*

PROPERTY 2. *The amount of memory required for agent x is $O(|D_x| \times |neighbors_x|)$.*

PROPERTY 3. *After terminating, resultValue variables values of all agents form optimal variable assignment B^*:*

$$B^* = \arg\min_B \sum_{i=1}^{m} f_i(B)$$

PROPERTY 4. *The number of messages processed by any agent between receiving SEARCH message and reporting results in corresponding $cost_x$ message is $O(|D_x| \times |neighbors_x|)$. Also, the size of each message does not depend on the problem, i.e. algorithm uses only messages of size $O(1)$.*

5. EVALUATION

The main metric for evaluation was concurrent constraint checks (CCC) [6]. It combines computational and communicational cost of solving the problem and allows for parallelism. The unit of cost is time needed to perform one constraint check. The cost of a message is defined in terms of constraint checks and does not depend on message size. It is easy to add CCC as a benchmark for a simulated distributed system by adding a special field with a value of local "tick counter" v to every message. Upon receiving the message, the receiver updates its counter value according to

$$v_{receiver} = \max\{v_{receiver}, v_{sender} + cost_{message}\},$$

When an agent evaluates a constraint, v is increased by 1. The cost of solving the problem is the maximal value of v over all agents.

Fig. 3(a), 3(b) and 3(c) show the dependency of the problem solving cost on the number of variables for sparse (3(a)) and dense (3(b), 3(c)) settings. The respective message costs were $cost_{message} = 10^3, 0, 10^6$ ($cost_{message} = 0$ corresponds to pure computational cost of an algorithm). In Fig. 3(d) one can see the total number of messages sent by all agents during the solution process depending on the problem size.

Summarizing the experimental results, we conclude that for almost all parameter settings NCBB performance-wise dominates ADOPT, and DPOP dominates NCBB. However,

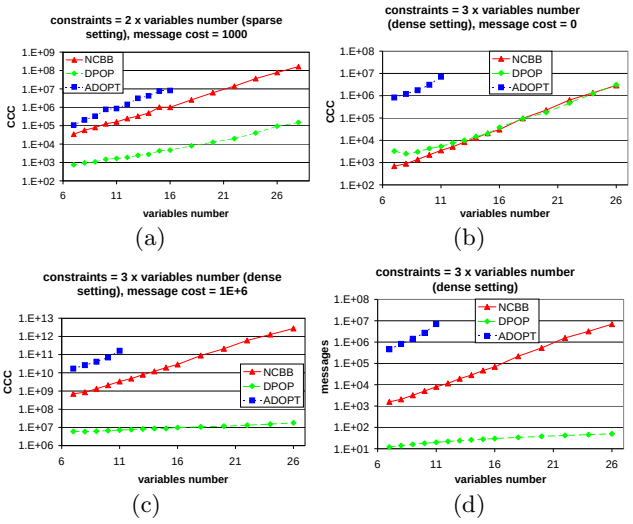


Figure 3: NCBB performance results

DPOP requires exponential memory on each agent, so it is not always applicable. For example, nodes in sensor networks have scarce memory, and it is enough even for one agent to not have enough space for DPOP to fail. For such applications NCBB may be the optimal choice given its performance advantage over another polynomial algorithm and the fact that NCBB is itself polynomial-space.

6. CONCLUSIONS

We have presented a novel backtracking-based algorithm for solving DCOP. It has polynomial memory requirements and uses messages of constant size. It offers significant performance improvements over the state of the art polynomial-memory algorithm ADOPT, and thus may be the optimal choice for settings with scarce memory, such as sensor nets.

7. REFERENCES

[1] S. Ali, S. Koenig, and M. Tambe. Preprocessing techniques for accelerating the DCOP algorithm ADOPT. In *Proc. of AAMAS*, 2005.

[2] K. Hirayama and M. Yokoo. Distributed partial constraint satisfaction problem. In *Principles and Practice of Constraint Programming*, pages 222–236, 1997.

[3] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.

[4] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[5] R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *Proc. of AAMAS*, 2004.

[6] A. Meisels, E. Kaplansky, I. Razgon, and R. Zivan. Comparing performance of distributed constraints processing algorithms. In *Proc. of DCR Workshop, AAMAS*, 2002.

[7] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 16(1–2):149–180, 2005.

[8] A. Petcu and B. Faltings. An efficient constraint optimization method for large multiagent systems. In *Proc. of the LSMAS Workshop, AAMAS*, 2005.

An Agent Based General Solution Model For the Course Timetabling Problem

Yan Yang
Electronic Systems Engineering
TRLabs Regina
University of Regina
1-306-7916141
roc_young@hotmail.com

Raman Paranjape
Electronic Systems Engineering
TRLabs Regina
University of Regina
1-306-7916137
raman.paranjape@uregina.ca

Luigi Benedicenti
Software Systems Engineering
TRLabs Regina
University of Regina
1-306-5854855
luigi.benedicenti@uregina.ca

ABSTRACT
This paper proposes a general solution model using agent technology in order to solve two critical open issues in the course timetabling problem: application-specific issues and dynamic-constraint issues. In our solution model, each agent in the system represents a constraint in the course timetabling problem. Compared to existing course timetabling solutions, the proposed agent solution model has much better generality, flexibility, dynamics, modularity and scalability.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence-*Multiagent systems, Intelligent agents.*

General Terms: Algorithms, Design, Experimentation

Keywords: Multi-agent Architecture, Timetabling Problem

1. INTRODUCTION
There are fundamentally two open issues in the course timetabling problem [1-3, 8]. One is the application-specific issue in which a solution to one course timetabling problem may not be suitable for another one. The other is the dynamic-constraint issue in which the course timetabling problem becomes much harder when the constraints change dynamically. This paper proposes a general solution model using agent technology [4, 5, 9, 10] and attempts to solve the above issues using a system design approach. In this paper, the agent technology is based on the TEEMA (TRLabs Execution Environment for Mobile Agents) platform [7].

2. AN AGENT SOLUTION MODEL
No matter how different they are, any course timetabling problem will have certain common hard constraints: 1) no student has more than one course at the same time; 2) only one course is scheduled in a classroom at any one time; 3) the classroom assigned to any course is large enough to hold all enrolled students. Identifying the common hard constraints provides the basis to establish a general solution model for the course timetabling problem using agent technology.

A course is an event involving course name, class time, classroom and people, i.e., students and instructors. There are a set of constraints which govern the relationships between these entities. In the proposed agent solution model, each agent represents a constraint in the course timetabling problem. The model includes the following agents: the **CHC** Agent which represents the **C**ommon **H**ard **C**onstraints (identified above); the **HC** Agents which represent other additional **H**ard **C**onstraints (such as no class scheduled during the lunch break time); the **SC** Agents which represent the **S**oft **C**onstraints (such as no student assigned more than two classes consecutively). There is only one CHC Agent in the system. The number of HC and SC Agents depends on the specific course timetabling problem. The constraint agents are independent of one another. The system design is presented in three different modes of operation: the passive mode, the active mode and the parallel-active mode.

2.1 Passive Mode

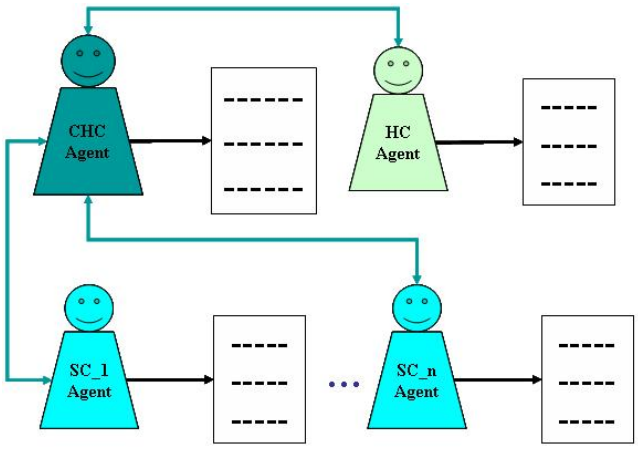


Figure 1. Passive mode of the general solution model

As shown in Figure 1, the passive mode general solution model is composed of one CHC Agent, one HC Agent and a number of SC Agents. The hard constraints can be represented by one HC agent (as shown in the figure) or a number of them (similar to the soft constraints). In order to simplify the illustration, only one HC

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. to copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

Agent is used. Generally, there are more than one soft constraint in the course timetabling problem. So in the solution model, the soft constraint agents are represented by the SC_1 Agent through SC_n Agent and the value of n depends on the number of soft constraints. The notepad symbol in the figure represents a course timetable. The solid polyline with bi-directional arrows between agents represents their communication. These symbol conventions will also apply in subsequent figures.

During the course timetabling process, the CHC Agent first produces a basic feasible course timetable consistent with the common hard constraints. Thereafter it sends a copy of the basic feasible timetable to each of other constraint agents. Each constraint agent evaluates the timetable based on its own constraint and sets penalty points on the number of constraint violations. These evaluations are sent to the CHC Agent and it develops an overall evaluation result on the basic feasible timetable. Following this, the CHC Agent creates a small disturbance to the timetable and guarantees that the disturbance does not violate the common hard constraints. The small disturbance can be realized by either moving one course to another available position in the timetable or exchanging the positions between two or three courses. As a timetable improvement proposal, the disturbance is sent to the constraint agents for evaluation. The constraint agents evaluate the proposal and provide feedback to the CHC Agent. After developing a new overall evaluation result on the modified timetable, the CHC Agent decides if the modified timetable represents an improved schedule. If the modified timetable is accepted, the CHC Agent will adopt it as the current best timetable and inform other constraint agents to make the same change. In this way, the system can guarantee that at any moment the course timetable carried by each agent is exactly the same. If the modified timetable is rejected, the CHC Agent will discard it. The CHC Agent then continues looking for a new proposal attempting to further improve the timetable. This process is repeated until all the constraints violations are eliminated and a final course timetable satisfying all the constraints is found. On the other hand, if the constraints violation can not be reduced further, the course timetabling process will go to the active mode described in the next subsection.

In this timetabling mode, only CHC Agent makes proposals and all the other constraint agents passively receive and evaluate proposals. For this reason we call this the passive mode.

2.2 Active Mode

The active mode is presented in Figure 2. From the figure it can be seen that each constraint agent has connections to all of others. In addition, there is a token which is passed around between the constraint agents. Only the agent owning the token has the right to make proposals to try to improve the course timetable and the proposals are guaranteed not to violate its own constraint. If a proposal violates the common hard constraints, it will be rejected because all timetables must always be basic feasible course timetable, even if they do not reduce other constraint violations. Otherwise, the timetabling process is similar to that described in the previous section.

In the active mode, all the agents equally contribute to improve the course timetable based on their own constraints. This allows the constraint violations to be reduced more effectively and efficiently. That is also the reason it is called the active mode.

If a solution can not be found after the token has been passed for a certain number of rounds among all the agents, the timetabling process is ended. Like other existing metaheuristics course timetabling solutions [6], the proposed method can not guarantee an optimal solution satisfying all the constraints. However, it has some key advantages over other existing methods which are identified in later sections.

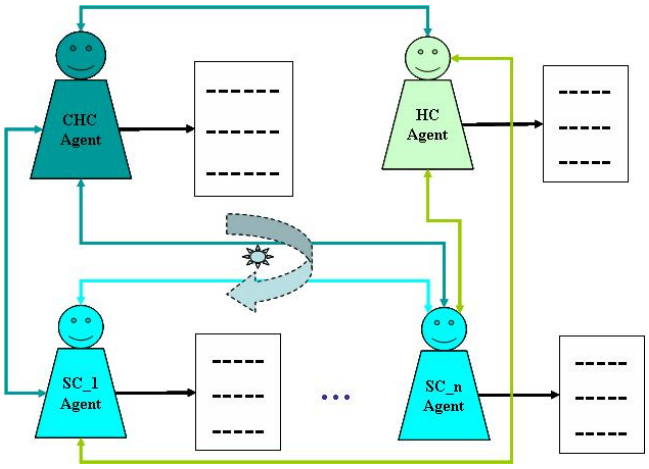


Figure 2. Active mode of the general solution model

2.3 Parallel-active Mode

The active mode can be further modified to make the token unnecessary allowing all the constraint agents to work in parallel. In this situation, the CHC Agent acts as a mediator to judge if the proposals are valid or not. Furthermore, each agent can adopt a different algorithm to create a hybrid and parallel course timetabling solution model. In such an operation mode, the system control would be much more complex, but it could be expected that the problem may be solved more quickly and more effectively.

3. EXPERIMENTAL RESULTS

The problem instances from Metaheuristic Network (http://www.metaheuristics. org/) are chosen to demonstrate the behavior of the new model. The problem description [6] is presented as follows:

The timetabling problem consists of a set of events to be scheduled in 45 timeslots (5 days of 9 hours each), a set of rooms in which events can take place, a set of students who attend the events, and a set of features satisfied by rooms and required by events. Each student attends a number of events and each room has a size. A feasible timetable is one in which all events have been assigned a timeslot and a room so that the following hard constraints are satisfied: 1) no student attends more than one event at the same time; 2) the room is big enough for all the attending students and satisfies all the features required by the event; 3) only one event is in each room at any timeslot.

Currently only the CHC Agent (representing the above three common hard constraints) and one SC agent (representing one of soft constraints in [6]: no student has a class in the last timeslot of

a day) have been implemented. The solution satisfying all the three common hard constraints and the mentioned soft constraint can be found through cooperation between the CHC Agent and SC Agent.

A portion of course timetabling results is shown in Figure 3. There are 45 timeslots (5 days of 9 hours each) and they are denoted by 0 through 44 in the implemented program. Timeslots No.8, 17, 26, 35, 44 are the last timeslots for each day of the week. Classrooms are denoted by integer numbers starting from 0. In the figure, the timeslot No.8 has no course scheduled. This is consistent with the soft constraint that students not have to attend the last timeslot of the day.

Time_Room	Room Size (No. of Students)	Event
	MONDAY	
7_0	9(9)	E28
7_1	2(2)	E74
7_2	3(3)	E29
7_3	11(10)	E25
7_4	15(15)	E32
Time Slot No.8: No event scheduled on this time slot!		
	TUESDAY	
9_0	9(9)	E31
9_1	2(2)	E83
9_2	3(3)	E88
9_3	11(11)	E37
9_4	15(15)	E43

Figure 3. A portion of course timetable

4. DISCUSSION AND FUTURE WORK

Although the quantitative assessment of the proposed solution model as compared with other existing course timetabling methods [6] has not been fully performed, some important qualitative advantages can be identified. Because each constraint agent is an independent module in the solution model, new constraint agents can be easily added into or removed from the timetabling system. This means that the dynamic-constraint issue is resolved through the system design. The proposed system design is extremely flexible, for example, course agents which represent individual courses can be added into the system. This would allow the proposed solution to adapt to the dynamic changes in schedule mandated even by individual courses.

The system design solves the application-specific issue in the course timetabling problem by allowing the user to select appropriate HC and SC Agents for their specific problem domain. The specific problem domain constraint agents make the solution model application-specific. Therefore the proposed solution is very general and flexible. The solution model's modularity means it can have any number of agents added into or removed from the system. This means there is a good scalability for the system as well.

The agents in the system can be mobile agents. In such a situation, the solution model can be used in the distributed environment and can take advantage of distributed problem solving. If the parallel- active mode (discussed in the subsection 2.3) is adopted, the proposed solution model will have the ability to solve problems in parallel.

To summarize, the advantages of the proposed solution model in this work over other existing approaches are as follows:

- Dynamic Problem Solving
- Good Flexibility and Generality
- Good Scalability and Modularity
- Distributed and Parallel Problem Solving

In order to explore the complete solution model, other soft constraint agent modules will be implemented. The parallel-active timetabling mode will also be investigated. Finally, quantitative comparison between the proposed solution model and other existing models will be made.

5. CONCLUSION

The proposed multi-agent general solution model for the course timetabling problem presented in this paper overcomes two key difficulties of the timetabling problem: application-specific issues and dynamic-constraint issues. The experimental results indicate that our model develops promising avenue of research in the course timetabling problem. It is reasonable to anticipate that the proposed solution model can be adapted to other scheduling problems.

6. REFERENCES

[1] Andrea Schaerf. *A Survey of Automated Timetabling.* Artificial Intelligence Review, 13: 87–127, 1999.

[2] Dominique de Werra. *An introduction to timetabling.* European Journal of Operational Research, 19: 151-162, 1985.

[3] Edmund Kieran Burke and Sanja Petrovic. *Recent research directions in automated timetabling*. European Journal of Operational Research, 140: 266-280, 2002.

[4] Hyacinth S. Nwana. *Software Agents: An Overview*. Knowledge Engineering Review, 11(3): 205-244, 1996.

[5] Michael J. Wooldridge, Nicholas R. Jennings. *Intelligent Agents: Theory and Practice*. Knowledge Engineering Review, 10(2): 115-152, 1995.

[6] Olivia Rossi-Doria, *et al*. *A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem.* E. Burke and P. De Causmaecker (Eds.): PATAT 2002, LNCS 2740: 329–351, 2003.

[7] Robert Martens, L. Benedicenti. *TEEMA TRLabs Execution Environment for Mobile Agents*. Technical Report, TRLabs Regina, Saskatchewan, Canada, 2001.

[8] Sanja Petrovic and Edmund Burke. *University Timetabling.* J. Leung (Eds.): Ch.45 in the Handbook of Scheduling: Algorithms, Models, and Performance Analysis, CRC Press, 2004.

[9] Yan Yang, Raman Paranjape and Luigi Benedicenti. *An Examination of Mobile Agents System Evolution in the Course-Scheduling Problem.* In Proceedings of the 2004 IEEE Canadian Conference on Electrical and Computer Engineering, Ontario, Canada, 5782-5785, May 2-5, 2004.

[10] Yan Yang, Raman Paranjape, Luigi Benedicenti and Nancy Reed. *A Mobile Agent System for University Course Timetabling*. In Proceedings of the 2nd Indian International Conference on Artificial Intelligence, Pune, India, December 20-22 2005.

Hierarchical Variable Ordering for Distributed Constraint Optimization

John Davin
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
jdavin@cs.cmu.edu

Pragnesh Jay Modi
Department of Computer Science
Drexel University
Philadelphia, PA 19104
pmodi@cs.drexel.edu

ABSTRACT

The Multiagent Agreement Problem (MAP) is a special form of Distributed Constraint Optimization (DCOP) that requires agents to choose values for variables to satisfy not only their own constraints, but also equality constraints with other agents. We introduce the AdoptMVA algorithm, an extension of the existing Adopt algorithm, designed to take advantage of MAP domains where agents often control multiple variables. We also propose an approach to agent ordering which leverages known ordering techniques from the centralized and distributed constraint satisfaction literature and applies them to MAPs. By combining ordering at the agent level with orderings at the variable level, we hope to obtain efficient global orderings. While the contributions discussed in this paper are applicable to general DCOPs, we focus our evaluation on MAPs because we feel it is a significant problem class worthy of specific attention.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Algorithms

Keywords

Distributed Constraint Satisfaction/Optimization

1. INTRODUCTION

The Multiagent Agreement Problem (MAP) is an instance of the more general Distributed Constraint Optimization Problem (DCOP) and is a convenient framework for representing coordination problems where multiple agents must come to an agreement for example on the time of a meeting or a joint action [7][5]. We propose AdoptMVA as an extension of the existing Adopt algorithm to operate more efficiently on MAPs where agents own multiple variables. The key idea is to use centralized search procedures to solve each agents' subproblem, in conjunction with the distributed Adopt algorithm to solve the broader problem. We show that when variable ordering is controlled for, AdoptMVA completes in fewer cycles than original Adopt on MAPs.

It is well known that the choice of variable ordering can have a large effect on the performance of a constraint-based search algorithm. Indeed, the choice of variable ordering becomes an even more important issue when we have multiple variables per agent. We define a set of inter-agent ordering heuristics for ordering at the macro-agent level and evaluate their comparative performance on MAPs using AdoptMVA. We present a general technique for converting existing effective variable orderings into effective agent orderings. In summary, by combining inter-agent ordering and intra-agent ordering heuristics and inventing AdoptMVA to exploit them, we enable a hierarchical approach to variable ordering in DCOP.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. MULTIAGENT AGREEMENT PROBLEM

The Multiagent Agreement Problem (MAP) [7] is a special class of DCOP [6] in which a variable can be shared among multiple agents. The model can be made equivalent to a DCOP by giving copies of the variable to each agent and using inter-agent equality constraints to ensure agreement. We will use MAP definitions to aid in explaining experiments described in this paper. The components of MAP are defined as follows:

- $\mathcal{A} = \{A_1, A_2, ..., A_n\}$ is a set of *agents*.
- $\mathcal{V} = \{V_1, V_2, ..., V_m\}$ is a set of *variables*.
- $\mathcal{D} = \{d_1, d_2, ..., d_k\}$ is a set of *values*. Each value can be assigned to any variable.
- *participants*$(V_i) \subseteq \mathcal{A}$ is the set of agents assigned the variable V_i. The participants are responsible for choosing the value of V_i.
- *vars*$(A_i) \subseteq \mathcal{V}$ is the set of variables assigned to agent A_i.
- For each agent A_i, C_i is an *intra-agent* constraint that must be defined *only* over the variables in *vars*(A_i). C_i can be used to model an agent's local preferences for certain values.[1]
- For each variable V_i, an *inter-agent* "agreement" constraint is satisfied if and only if the same value from $\mathcal{D}$ is assigned to V_i by all the agents in *participants*(V_i).

[1]This is a generalization of the model introduced in [7] where C_i was required to evaluate to either true or false.

3. ADOPTMVA

The existing Adopt algorithm can be trivially adapted to handle problems in which an agent owns multiple variables by treating variables within an agent as virtual agents. However, this previous approach has the drawback that it artificially segments problem information within an agent and fails to take advantage of the full information sharing that is possible. We aim to eliminate this inefficiency by allowing full sharing of information within an agent. Our approach is to create a single process for each agent and allow it to control all of the variables owned by the agent. Then, a centralized search procedure is used to find the optimal solution for the agent's subproblem. We call this **AdoptMVA**, for Multiple Variables per Agent.

In the Adopt formalism, a value assignment to a set of variables is called a context and is defined as follows [6].

- Definition: A *context* is a partial solution of the form $\{(x_j, d_j), (x_k, d_k)...\}$. A variable can not appear in a context more than once. Two contexts are *compatible* if they do not disagree on any variable assignment. *CurrentContext* is a context which holds an agent's current knowledge of variable values for higher priority neighbors.

As defined in MAP, we let $vars(A_i) \subseteq \mathcal{V}$ denote the variables owned by agent A_i. We then denote a context $s \in S$ where S is the set of all possible assignments to variables in $vars(A_i)$ and s is a particular one of those assignments.

Whereas previously an agent owned only a single variable and so its cost was determined only by its constraints with other agents, we now must define a function that also includes the cost of constraints between variables *within* an agent. We define the *local cost* δ for a particular value assignment s made by agent A_i for the variables it owns as:

$$\delta(s) = \sum_{(x_j,d_j)\in s} \sum_{(x_i,d_i)\in s} f_{ij}(d_i, d_j) \quad + \sum_{(x_j,d_j)\in CurrentContext} \sum_{(x_i,d_i)\in s} f_{ij}(d_i, d_j) \tag{1}$$

The first half of the definition sums the constraints between every possible pairing of variables within an agent, and the second half evaluates the constraints between each of the agent's variables and all of the external variables in its *CurrentContext*.

An agent computes a *lower bound for a solution s* using lower bound costs received from its children, denoted as $lb(s, x_l)$.

- Definition: $LB(s) = \delta(s) + \sum_{x_l \in Children} lb(s, x_l)$ is a *lower bound* for the subtree rooted at A_i when A_i chooses solution $s \in S$.

Similarly for an *upper bound on solution s*:

- Definition: $UB(s) = \delta(s) + \sum_{x_l \in Children} ub(s, x_l)$ is an *upper bound* for the subtree rooted at A_i when A_i chooses solution $s \in S$.

The overall lower and upper bounds for A_i are the minimums over the bounds for all possible solutions in S:

- Definition: $LB = min_{s \in S} LB(s)$ is a *lower bound* for the subtree rooted at A_i.

- Definition: $UB = min_{s \in S} UB(s)$ is an *upper bound* for the subtree rooted at A_i.

Algorithm 1: AdoptMVA Branch & Bound search

```
depth ⟵ 0
cost ⟵ 0
bestCost ⟵ ∞
procedure search( s ∈ S, depth, cost ):
if depth == s.size() then
    bestSolution ⟵ s
    bestCost ⟵ cost
    return;
end
V_i ⟵ variable at depth (order determined by variable ordering)
Domain ⟵ D_i
Reorder Domain with best-first heuristic (move current value of V_i in best known solution to be first)
for d_i ∈ Domain do
    s{V_i} ⟵ d_i
    cost+ = Σ_{(x_j,d_j)∈s} f_ij(d_i, d_j) +
    Σ_{(x_j,d_j)∈CurrentContext} f_ij(d_i, d_j)
    if depth == s.size() − 1 then
        cost ⟵ cost + Σ_{x_l∈Children} lb(s, x_l)
    end
    if cost < bestCost then
        search( s, depth+1, cost )
    end
    s{V_i} ⟵ null
end
```

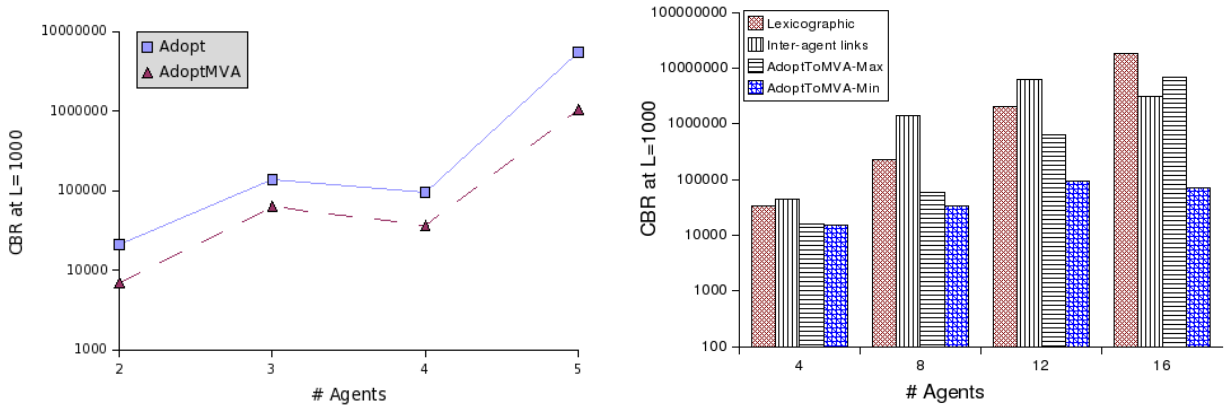
The above definitions are based on the original Adopt definitions [6] with the main change being that lower and upper bounds are now calculated using a local solution context s rather than a single variable x_i. Also, the most important change to the local cost δ is that it now includes the cost of intra-agent constraints.

In order to properly communicate the values of an agent's local variables to the other agents in the problem, we also have to modify Adopt's VALUE messages and its handling of COST messages. In particular, VALUE messages must now include the values of all variables owned by the agent, rather than just a single variable value. Therefore, we extend VALUE messages to include a solution context:

- **Definition:** *VALUE(* $s_i = \{(x_j, d_j), (x_k, d_k), ...\}$ *)* is the form of the new VALUE messages sent to neighbors lower in the tree.

Finally, each agent uses a centralized Branch & Bound search [4] to find the optimal solution to its local variables, given the values of its ancestors in the Adopt hierarchy, and knowledge of the costs of its children. The search is used to calculate both the lower and upper bounds (LB and UB) on the agent's variables. Algorithm 1 shows pseudo code for the lower bound search (upper bound search is identical except $lb(s, x_l)$ is replaced with $ub(s, x_l)$). The value ordering is a best-first heuristic that puts the best domain value first in the search order for a given variable, where the best value is taken from the current best solution (if known).

Also note that the cost calculation in each step of the recursive search is not exactly the same as the δ function in Equation 1. Rather, the true cost of δ is reached at the bottom of the recursion because we add in the lower or upper bounds from our children at the end. This is so we don't redundantly calculate this cost at each recursion. If a bound for a child at the given solution s is not



(a) AdoptMVA versus Adopt

(b) AdoptMVA agent ordering heuristics

Figure 1: Experimental results on MAPs in the meeting scheduling domain.

known, the lower bound defaults to 0 and the upper bound to ∞.

4. ORDERING HEURISTICS

We define four inter-agent orderings, which are macro-level variable orderings used to determine the parent/child relationships in AdoptMVA.

- **Lexicographic** - Agents are ordered alpha-numerically by agent name. This is an uninformed heuristic.
- **Inter-agent constraints heuristic** - Agents are ordered recursively by the number of inter-agent constraints to already ordered agents. Ties are broken by the number of inter-agent constraints to currently unordered agents, and then lexicographically. This is a similar to the existing Brelaz variable ordering heuristic [1] except rather than counting all constraints, we only count inter-agent constraints.

The next two orderings were created based on the theory that many effective variable ordering heuristics exist and so converting them to an agent ordering might produce similar results. In particular, we can convert any given global variable ordering to an agent ordering by assigning agent priorities based on the priorities of variables within that agent. Given a prioritization of variables, a general technique for computing agent A_i's agent priority is as follows:

$$priority(A_i) = max_{\forall V_j \in vars(A_i)}\, priority(V_j).$$

Below are two heuristics that instantiate this theme.

- **AdoptToMVA-Max** - We first compute a global variable ordering using Brelaz [1], and then convert it into an agent ordering for AdoptMVA using the above conversion.
- **AdoptToMVA-Min** - This is the same as AdoptToMVA-Max except that agent priorities correspond to the *minimum* priority variable within each agent.

We will present experimental results for the four agent ordering heuristics. Although not presented here, we also investigated a set of heuristics suitable for MAP for ordering variables *within an agent*, i.e., intra-agent variable orderings. We evaluated their comparative performance and are able to identify an intra-agent variable ordering heuristic for MAP that is the most efficient of the ones tested.

5. RESULTS

We conducted experiments on MAPs used to model a meeting scheduling domain. More results than those presented here are available [2]. We present results in terms of the Cycle-Based Runtime (CBR) metric [3] which we believe is a more informative and fairer metric because it more closely approximates true distributed run-time than the previously used synchronous cycles metric. The formula for CBR is $CBR = Cycles * L + CCC$, where cycles is the number of synchronous cycles consumed by the algorithm, L is a factor indicating the relative communication speed, and CCC is the total number of concurrent constraint checks.

First, we compare the performance of AdoptMVA versus Adopt. To ensure a level playing field, we control for agent ordering using a static lexicographic agent ordering for both algorithms. We found that when agent ordering is controlled for, AdoptMVA has a lower CBR than Adopt (see Figure 1a). The results show that AdoptMVA would perform better than Adopt in systems with a high communication cost (L=1000).

Next, we compare four agent ordering heuristics on MAPs. Figure 1b shows our results. AdoptToMVA-Min was the best performing heuristic on the problems tested. We also saw that the agent ordering can make a tremendous difference. For some of the problems in Figure 1b, the difference between two heuristics is one or two orders of magnitude in size.

6. REFERENCES

[1] D. Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.
[2] J. Davin. Algorithmic and domain centralization in distributed constraint optimization problems. Master's thesis, Carnegie Mellon University, 2005.
[3] J. Davin and P. J. Modi. Impact of problem centralization in distributed constraint optimization algorithms. In *Proc of AAMAS*, 2005.
[4] E. C. Freuder and R. J. Wallace. Partial constraint satisfaction. *Artif. Intell.*, 58(1-3):21–70, 1992.
[5] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham. Taking DCOP to the real world. In *Proc of AAMAS*, 2004.
[6] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees. *Artificial Intelligence Journal*, 2005.
[7] P. J. Modi and M. Veloso. Bumping strategies for the multiagent agreement problem. In *Proc of AAMAS*, 2005.

Demonstration Track

SeSAm: Implementation of Agent-Based Simulation Using Visual Programming

F. Klügl, R. Herrler and M. Fehler
University of Würzburg
Am Hubland, 97074 Würzburg
Germany
{kluegl, herrler, fehler}@informatik.uni-wuerzburg.de

ABSTRACT
In this paper, we present the most important features of SeSAm, a modeling and simulation platform for multi-agent simulations. Based on a declarative, explicit model representation and visual programming, it allows implementing models on specification level. Optimizing compilation allows efficient simulation of the explicit model representation. It was successfully applied in different areas, like biology, traffic or logistics simulation.

Categories and Subject Descriptors
I.6.7 [**Simulation and Modeling**]: Simulation Support Systems - *Environments*

General Terms
Design, Experimentation, Languages

Keywords
Agent-based Simulation, Visual Programming

1. INTRODUCTION
Agent-based simulation offers quite a lot of advantages compared to traditional (also microscopic) approaches, like the elegant representation of heterogeneous populations situated in heterogeneous environments, the explicit embodiment of local effects, or the possibility of flexible interaction between entities based on intelligent behaviour. These properties are the reason why the use of agent-based simulation becomes more and more popular. However, the development of a useful agent-based simulation makes great demands on the modeller as the properties that complicate the development of a multi-agent system also apply here. On a technical level this refers to concurrent processes, in-transparent feedback loops, etc. Additionally when constructing simulation models the freedom of design is restricted as the relation to the original system has to be established. The effort required for producing a structurally valid model with appropriate parameter settings, may be really high. A useful tool for modelling and simulation is highly valuable especially for modellers that are no programming experts. Meanwhile, a number of tools and frameworks for supporting the development of agent-based simulations are available. Most of them consist of a (Java-) class library that has to be extended by model-specific program code.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00

At the department of Artificial Intelligence and Applied Computer Science at the University of Würzburg (headed by Frank Puppe), we developed a modelling and simulation environment that combines concepts of declarative high-level model representation and visual programming. The initial aim behind our efforts was to provide a tool for beginners in modelling and programming. Scientists working mostly empirically should be equipped with a tool that allows them use modelling and simulation. The SeSAm team is headed by Franziska Klügl and currently consists of Rainer Herrler, Manuel Fehler, Cornelia Triebig and Gustavo Kuhn Andriotti. Also, several students are continuously contributing to SeSAm.

Besides usage in teaching, SeSAm is applied in several projects for simulation of social insects, reproduction of shopping behaviour, route-choice models in traffic simulation. Beyond such traditional application areas of agent-based simulation, SeSAm forms the technical base for virtual high bay warehouses that are used for software tests, requirements engineering or employee training.

SeSAm contains all components that are necessary for a useful simulation and modelling environment. Additional features – like debugging or refactoring – are inspired by tools for general software development and make SeSAm a valuable tool for rapid prototyping of agent-based simulations. The overall components are depicted in figure 1.

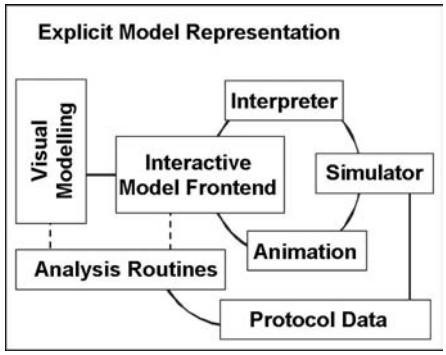


Figure 1: Components of the SeSAm system

In the following we want to focus on some innovative aspects, ignoring others like automatic calibration, integrated evolution, etc.

2. MODEL REPRESENTATION

Models are represented declaratively to assure the separation between simulator and actual model. The high-level modeling language consists of elements on different levels: So called primitives – agent actions, predicates for perception or other functions for processing information – are responsible for translating the declarative representation to executable code. With each model element a code fragment is associated that actually may realize the manipulations or queries. The description of a primitive is augmented with information concerning arguments or return value.

The state of agents, resources or the environment is represented by a body consisting of state variables. Behavior is specified using an activity graph that is derived from UML. An activity of an agent corresponds to a special state variable that determines the next action sequence of an agent. Rules are used for transition between activities. Thus, this agent behavior language has the power of a standard programming language. Not only agent models that are based on behavior description are represent-able using activity graphs, but – in combination with complex data structures – also simple agent reasoning may be reproduced.

Additional language elements allow a clear and efficient overall model representation. Examples are aggregated activities, which again may contain a complete activity graph, leading to a hierarchical structure. Model-specific primitives, data types or features enhance model entities with additional abilities. The language is extensible based on a plugin concept. Using the same language elements, the instrumentation and definition of experiment scripts containing several, differently configured simulation runs, can be managed. All information concerning the model is stored in a XML-file.

3. VISUAL PROGRAMMING

Based on the declarative model representation, visual programming is consequently applied not only for implementing the structure and behaviour of the agents and their environment, but also for data output generation, experiment scripting or constructing an interactive simulation GUI. The SeSAm design is based on the following principles:

- Complexity is hidden from beginners, but experts are allowed to use the full power
- Similar elements are always edited using similar editors
- No path for implementing a simulation model is a priori given, editors are accessible from all reasonable contexts
- Fault safety based on syntax tests and restriction of all input possibilities to syntactically correct ones.
- Documentation facilities for all model elements.

All elements of a SeSAm model and an executable simulation experiment are accessible via a central hierarchy. Structures are edited using forms, behaviour using graphs and situations may be constructed using maps. For activity transition rule conditions and all primitive calls, dialogs with three parts are given. On the left side the current function tree, in the middle possible absolute values and on the right side a list of all primitives that may be appropriate in the current model context. See figure 2 for a sample of screenshots.

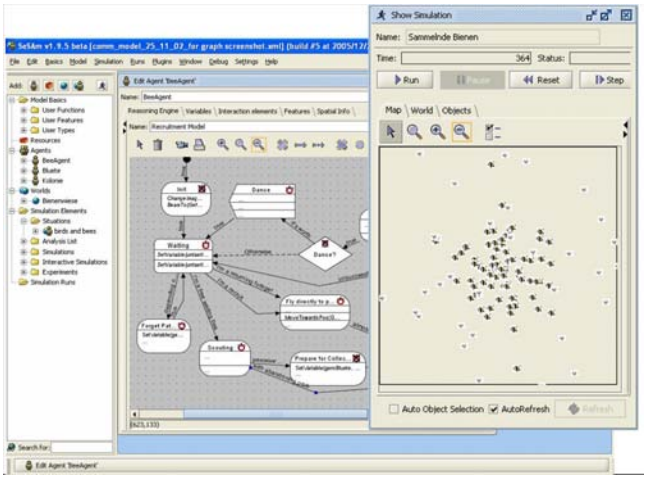


Figure 2: Screenshot of the SeSAm system: in the background an activity graph, in foreground a standard animation window.

Usability of SeSAm for development of agent-based simulation is additionally supported based on features known from modern software development environments. A stepper allows stepping through the behaviour of agents on the action level, a debugger allows to set break points for supporting the search for errors. A list of pre-defined refactoring actions helps to improve the model implementation design. However, a comfortable modelling environment is useless, if the actual simulations are not executable in sufficient speed. Therefore the declarative model representation is not interpreted directly, but compiled using optimization methods from compiler design.

Model-specific extensions or additional functionalities that are not available in standard SeSAm, can be added based on a plugin concept that enables to add primitives, data types or editors. A number of plugins, e.g. for data base connectivity, FIPA-ACL communication between simulated and real-world (JADE) agents, Ontology import (from Protegé projects) are downloadable from the SeSAm web site. Plugins also exist for different spatial representations like 2d continuous (standard), 3d or GIS vector data.

4. AVAILABILITY

SeSAm is developed under the LGPL license. Currently (Februar 2006), SeSAm 1.9.5 and several plugins may be downloaded for free from our SeSAm webpage: www.simsesam.de. We are preparing version 2.0 that will be available in summer 2006. SeSAm is based on Java 1.5, thus versions and comfortable installers for all popular operating systems are available. The possible number of simulated agents is depending on the memory configuration of the computer SeSAm runs on. Usually 512 MB are sufficient for larger models 1 GB and more are advisable.

CSC: Criticality-Sensitive Coordination *

Pedro Szekely†, Marcel Becker‡, Stephen Fitzpatrick‡, Gergely Gati⋆, David Hanak⋆, Jing Jin†, Gabor Karsai⋆, Rajiv T. Maheswaran†, Bob Neches†, Craig M. Rogers†, Romeo Sanchez†, Chris van Buskirk⋆

† Information Sciences Institute, University of Southern California, 4676 Admiralty Way - Suite 1001, Marina Del Rey, CA 90292

‡ Kestrel Institute, 3260 Hillview Avenue, Palo Alto, CA 94304

⋆ Institute for Software Integrated Systems, Vanderbilt University, Box 1829, Station B, Nashville, TN 37235

szekely@isi.edu, becker@kestrel.edu, fitzpatrick@kestrel.edu, gergely@isis.vanderbilt.edu, dhanak@isis.vanderbilt.edu, jingjin@isi.edu, gabor.karsai@vanderbilt.edu, maheswar@isi.edu, rneches@isi.edu, rogers@isi.edu, rsanchez@isi.edu, chris.vanbuskirk@vanderbilt.edu

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems, coherence and coordination*

General Terms

Algorithms, Design Experimentation, Performance

Keywords

Reasoning Under Uncertainty, Large-Scale Systems, Visualization

1. INTRODUCTION

Our Criticality-Sensitive Coordination (CSC) agents are designed to enhance the performance of a human-team working together in uncertain and dynamic settings by monitoring and adapting their plans as dictated by the evolution of the environment. Such situations model military scenarios such as a coordinated joint operations or enterprise settings such as multiple-project management. Among the many challenges in these situations are the large space of possible states due to uncertainty, the distributed / partial knowledge of current state and plan among the agents and the need to react in a timely manner to events that may not be in the original model. In fact, reaction alone is often insufficient as in environments where success depends on completing sequences of coupled actions, one needs to anticipate future difficulties and enable contingencies to alleviate potential hazards.

2. SYSTEM DESCRIPTION

To deal with these challenges, the CSC agents are constructed with components that interact in a multi-tiered manner with functionalities that operate in various time-scales and reasoning domains. The system architecture is depicted in Figure 1. The highest level of reasoning is performed by three components: (1) the deliberative scheduler, (2) the opportunistic scheduler, and (3) the downgrader. The deliberative scheduler is triggered when it has been determined that a portion of the existing plan is anticipated to fall below a certain likelihood of success. It then performs *dynamic partial-centralization* of the relevant agents and nodes in the plan, extracts potential solutions and proposes schedule modifications to remedy the problem. Due to the complexity of this task and the nature of the anticipated failure, this process occurs over several decision epochs with decision windows that begins slightly in the future. The opportunistic scheduler performs schedule modifications in the near-term. Using only local information, it is capable of making decisions on a faster time-scale, and buffers the existing plan by utilizing free resources. This provides robustness by increasing the probability of success and generating opportunities to obtain higher quality outcomes. The downgrader complements the previous two components by freeing resources both in the near-term and the future based on the evolution of the system up to the present. The schedule, estimates of system state and metrics of uncertainty are kept in the state manager which communicates with the state managers of other agents to distribute and collect locally visible and locally relevant information. The *profiles* capture the probability of success (p) and the importance (α) for the activities in the plan and are propagated in a decentralized manner. Each agent has profiles and schedules both for itself, for remote agents that affect it directly and for plan components for which it is responsible. The local schedule information is sent to an execution controller which interacts with the environment. This component is isolated such that it can operate at a much faster time-scale than the decision epoch intervals, such that the agent can continue to function even under high computational burdens in other components of the system.

3. DEMO DESCRIPTION

The capabilities of the system and the complexities of the problem are illustrated in a scenario where two subteams, represented by CSC agents Alpha and Bravo, participate in a three-phase joint project/operation. Intiially, they agree to engage in Plan A, which involve each agent's subteam performing certain activities at each phase. However, they can fall back to Plan B, which involves different activities at each phase, if needed. The activities are not independent as one may need to perform a certain activity in an earlier

*The work presented here is funded by the DARPA COORDINATORS Program under contract FA8750-05-C-0032. The U.S.Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Figure 2: Two-Agent Three-Phase Example

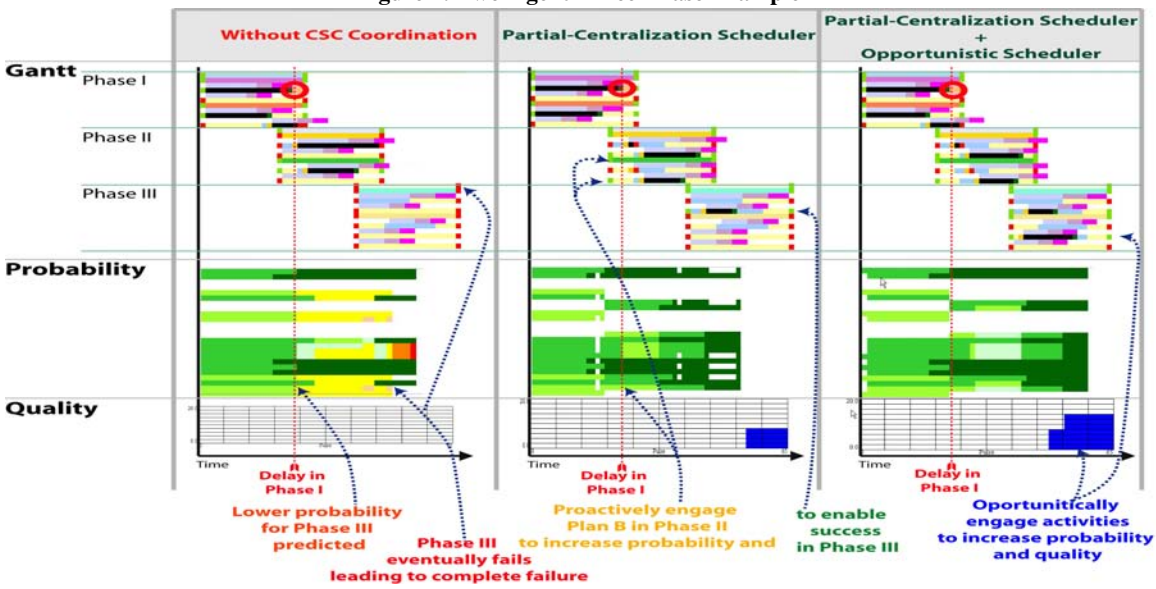


Figure 1: CSC Agent Architecture

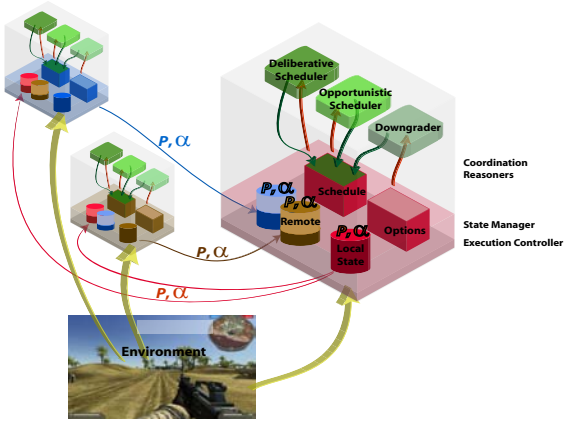
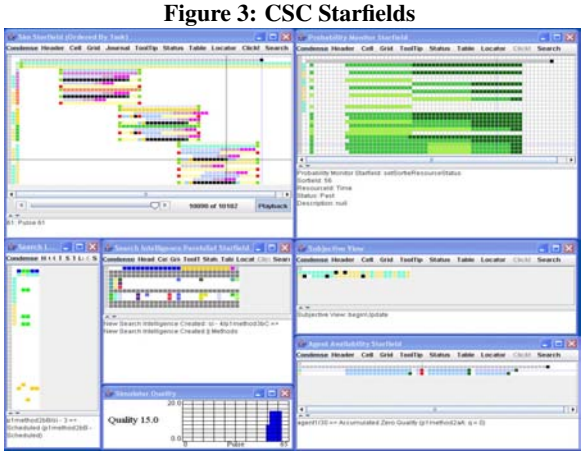


Figure 3: CSC Starfields



phase to have the option of performing an activity in a later phase. Thus, the necessity of anticipation in the presence of uncertainty. A depiction of an evolution is shown and discussed in Figure 2.

Without CSC, the initial plan can be undermined in many ways. A delay of an activity in Phase 1 cascades the delay on to Phase 2 implying a high chance of failure. While Phase 2 might succeed despite the failure of a single activity within it, Phase 3 will be completely damaged due to dependencies of activities across multiple phases. However, the various components of CSC interact to allow adverse circumstances to be ameliorated with ease. The state manager for the agent will detect in Phase 1 itself that the probability of success for activities in Phase 3 have fallen below critical levels and will instantiate dynamic partial-centralization by calling the deliberative scheduler. The deliberative scheduler will then consider alternatives, in this case Plan B, and install new activities in the second phase and third phase. The downgrader frees up the resources in Phase 2 that were operating under the directives of Plan A, such that they are available for Plan B to execute. Finally, the opportunistic scheduler utilizes resources that were freed up in Phase 3 (by the downgrader) to add an activity that enhances the quality of the solution.

4. STARFIELDS

One of the unique and extremely beneficial aspects of CSC is the vast suite of tools (referred to as *starfields*) available for visualizing various components and the evolution of the system. These tools offer the ability to observe behavior at a macro-level. This helps a user quickly and accurately gauge system behavior and also provides the ability to isolate individual components at particular instants in time which is an invaluable aid in debugging. A snapshot of selected starfields is displayed in Figure 3. A key functionality of starfields is the ability to play back any evolution in time to observe the system at chosen instants. This allows a user to identify exactly when and how the system identifies and resolves problems. Another key feature is the ability to search and cross-reference activities and plan components, which offers the capability to isolate an activity in multiple starfields and see how it couples with activities in other phases of the plan. The starfields facilitated rapid modification, development and debugging in addition to aiding in the understanding of the underlying problem which becomes difficult to grasp as the scale becomes large. A movie of the demonstration can be downloaded from *http://www.isi.edu/~szekely/csc/aamas06/csc-demo-v01.html*.

Synergistic Integration of Agent Technologies for Military Simulation

Sean Owens, Paul Scerri, Robin Glinton, Bin Yu and Katia Sycara
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{owens, pscerri, rglinton, byu, katia}@cs.cmu.edu

ABSTRACT

To perform large-scale coordination in real-world environments requires that many individually complex technologies come together to form integrated solutions. We present an application where several key technologies are integrated into a unified system via a multiagent infrastructure. We show how the synergistic behavior among heterogeneous technologies results in a significant improvement over the performance of the individual technologies acting alone. Critical extensions were required to the language describing required behavior to allow the pieces to work together. Initial experimental results show system performance on a task of coordinating a military convoy in an adversarial environment was significantly improved when all technologies worked together. However, experiments with a human user in the loop showed that significant advances must still be made before such systems can be fielded in the real-world.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Coherence and coordination

General Terms

Algorithms,Experimentation

Keywords

MultiAgent Coordination, Simulation

1. INTRODUCTION

Autonomous coordination has rapidly progressed over recent years, with significant advances in both specific algorithms[7] and complete approaches to coordination being successfully demonstrated[6]. High bandwidth, widely available, low cost communcation and improvements in algorithm scalability are allowing significant advances in the number of agents able to be efficiently coordinated[9, 11]. Moreover, advances in artificial intelligence[8] and robotics mean that the robots and agents being coordinated are significantly more individually intelligent than in most previous multi-agent systems. This progress is quickly putting important, exciting applications of multiagent technology within reach. However, because there have been relatively few demonstrations of large scale, highly complex multi-agent systems, it is unclear whether existing automated coordination techniques are sufficient for these very complex systems or whether key challenges remain.

Many exciting multi-agent and multi-robot applications have been demonstrated in recent years. Systems including RETSINA-OOA[4] and the Electric Elves[2], have demonstrated coordination of heterogeneous sets of entities to achieve complex objectives. However, in many of these systems, much of the intelligence can be attributed to the coordination, with the individual agents being relatively simple, e.g., web services. Some military simulation environments involve both very complex individual technologies and large numbers of heterogeneous entities, but do not typically make extensive use of state-of-the-art automated coordination technology[1]. Thus, despite the obvious potential for automated coordination of large numbers of intelligent heterogeneous entities, there is a lack of sufficiently high fidelity systems that can expose key remaining challenges. Moreover, in most of these systems effective user control was not required, yet many real-world applications cannot be deployed without such control.

To better understand the challenges of complex coordination of intelligent entities, we have developed a simulation environment, called Sanjaya, where successful performance requires effectively dealing with these type of challenges. Sanjaya is a large scale military simulation where many realistic issues including uncertain sensing, diverse terrain, an intelligent adversary, large numbers of units and complex plans, are modeled. A sophisticated terrain analysis agent[5] provides the team with expected locations of opposing forces and safe paths for travel. A sensor fusion process is available to take sensor readings from multiple sources and reduce uncertainty when distinguishing between opponent and civilian vehicles. Importantly, a human commander has high level control and makes decisions to strike identified opponents. Each of the system components is able to use input from the other components and provide input to those components. Intuitively, synergies should occur in Sanjaya, since the various agents should be able to mutually provide each other

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

with input to improve performance.

A variety of distinctly different approaches to coordination exist, developed to meet specific goals and have specific properties. Due to the dynamic nature of the fundamental task of moving and protecting convoys of ground vehicles, we choose to apply *teamwork* based coordination, since it is designed to be flexible and robust, key requirements for this domain[3]. The specific proxy-based implementation of teamwork that we use executes user designed *team oriented plans* instantiated at runtime from templates[10]. While team oriented plans have been used extensively in the past, when attempting to apply them to this domain, we encountered two key problems, both related to the production and use of information during coordination. First, the plans did not have sufficient semantic content to allow the proxy to determine where information produced during plan execution should be directed. For example, when the terrain analysis agent generated a least resistance path, it could not determine what to do with the path. Second, some tasks within a plan could not be initiated until other tasks within the plan produced information to completely specify the task, but there was no way of specifying this. For example, a task for protecting a convoy could not be allocated until the path of the convoy was determined. Extensions adding additional semantics, but not changing the declarative nature of the TOP's were implemented.

While automated coordination and intelligent robot and agent technology is rapidly advancing, human intelligence is staying constant. Several previous investigations have shown that human interaction with intelligent distributed systems is extremely difficult, sometimes even leading to poorer performance than the system on its own[12]. A key interface design issue is how much of the underlying intelligence to make opaque to the user and how much to make transparent. In Sanjaya, we took the approach that when the details of the intelligence reasoning were not critical to the human's reasoning, they were made completely opaque. We performed an initial set of user tests with some users being in control of a system with all the intelligent agents performing and another set of users using a less intelligent system with the same interface. Our results showed that users actually performed relatively better with a less intelligent system, indicating that significant advances need to be made before users can exert effective control over very complex multiagent systems.

2. SANJAYA

Sanjaya is a constructive simulation supporting simulation of ground, air and unmanned aerial vehicles. The scenario being tested consists of five convoys of ground vehicles each supported by a group of unmanned aerial vehicles (UAVs), traveling across a 50km by 50km map defended by 20 opposing tanks. This is a challenging scenario for a number of reasons.

The UAVs have a limited sensing range, and there are not enough of them in the scenario to patrol the entire map. Obviously intelligent coordination is necessary to use the UAVs efficiently. When an opponent has been found the UAVs may be used to attack the oppoent, but in doing so the UAV destroys itself. As we have a limited number of UAVs in the scenario, each UAV used to attack further reduces our ability to sense, and to destroy any further opponents found.

Sanjaya simulates an uncertain sensing environment. Sensor readings return a list of possible classifications of sensed objects, with a probability associated with each classification. In the scenario being simulated, sixty trucks, representing civilian noncombatants, are placed on the map and may be confused with enemy forces. UAVs, are a limited resource and must be conserved to attack those entities that we are reasonably certain are actual opposing forces. This means that we must have a reasonably high confidence classification of a sensed entity as a military, non civilian entity before we issue a command to attack it.

Unlike other currently available simulation environments, Sanjaya simulates the kind of uncertainty and confusion that can occur in large scale coordination in real-world environments. We find this type of simulator necessary to study the challenges that large scale real-world coordination poses. Furthermore, this type of simulator is necessary to determine potential synergies that can occur while coordinating complex entities and technologies.

Results illustrate that the infrastructure was able to produce synergies between the individual technologies. However, individual technologies did not always improve overall performance. Importantly for the future deployment of complex multiagent systems, initial user testing showed that users were not able to take full advantage of available technology and often made system performance worse.

3. REFERENCES

[1] S. M. Aircraft. The TACSI users guide. technical report GDIO-MI-98:356. Technical report, Saab Military Aircraft, 1995. Edition 5.2.

[2] H. Chalupsky, Y. Gil, C. A. Knoblock, K. Lerman, J. Oh, D. V. Pynadath, T. A. Russ, and M. Tambe. Electric Elves: Agent technology for supporting human organizations. *AI Magazine*, 23(2), 2002.

[3] P. R. Cohen and H. J. Levesque. Teamwork. *Nous*, 25(4), 1991.

[4] J. Giampapa and K. Sycara. Team oriented agent coordination in the RETSINA multi-agent system. In *Proceedings of Agents02*, 2002.

[5] R. Glinton, S. Owens, J. Giampapa, K. Sycara, C. Grindle, and M. Lewis. Terrain based information fusion and inference. In P. Svensson and J. Schubert, editors, *Fusion 2004*, volume I, Mountain View, CA, Jun 2004. ISIF

[6] M. Goodrich, D. Olsen, J. Crandall, and T. Palmer. Experiments in adjustable autonomy. In H. Hexmoor, C. Castelfranchi, R. Falcone, and M. Cox, editors, *Proceedings of IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents*, 2001.

[7] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. An asynchronous complete method for distributed constraint optimization. In *AAMAS*, 2003.

[8] R. Nair, M. Tambe, and S. Marsella. Role allocation and reallocation in multiagent teams: Towards a practical analysis. In *AAMAS*, 2003.

[9] C. L. Ortiz, R. Vincent, and B. Morisset. Task inference and distributed task management in centibots robotic systems. In *AAMAS*, 2005.

[10] D. Pynadath, M. Tambe, N. Chauvat, and L. Cavedon. Toward team-oriented programming. In *Intelligent Agents VI: Agent Theories, Architectures, and Languages*, 1999.

[11] P. Scerri, Y. Xu, E. Liao, J. Lai, and K. Sycara. Scaling teamwork to very large teams. In *AAMAS*, 2004.

[12] N. Schurr, J. Marecki, M. Tambe, P. Scerri, J. Levis, and N. Kasinadhuni. The future of disaster response: Humans working with multiagent teams using DEFACTO. In *AAAI Spring Symposium on Homeland Security*, 2005.

Asynchronous Chess Competition

Nathaniel Gemelli, Robert Wright, James Lawton and Andrew Boes

US Air Force Research Lab
Rome Research Site
Rome, NY, USA

{Nathaniel.Gemelli, Robert.Wright, James.Lawton, Andrew.Boes}@rl.af.mil

ABSTRACT

Asynchronous Chess (AChess) is a platform for the development and evaluation of real-time adversarial agent technologies. It is a two-player game using the basic rules of chess with the modification that agents may move as many pieces as they want at any time. Modifying chess in this way creates a new robust, asynchronous, real-time game in which agents must carefully balance their time between reasoning and acting in order to out-perform their opponent. As a fast-paced adversarial game, many challenges relevant to real-world applications arise which give it merit for study and use.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent Agents

General Terms

Experimentation, Algorithms

Keywords

Adversarial Environments, Real-Time Reasoning, Intelligent Agents

1. INTRODUCTION

We present Asynchronous Chess [2] as a real-time adversarial software-agent demonstration and competition for AAMAS 2006. Asynchronous Chess (AChess) is a platform for the development and evaluation of real-time adversarial agent technologies. It is a two-player (i.e. *agent*) game using the basic rules of chess with the modification that agents may move as many pieces as they want at any time. Modifying chess in this way creates a new robust, asynchronous, real-time game in which agents must carefully balance their time between reasoning and acting in order to out-perform their opponent. As a fast-paced adversarial game, many challenges relevant to real-world applications arise which give it merit for study and use. We believe AChess will be of added value to researchers in the Artificial Intelligence field, and be well received as an AAMAS demonstration/competition.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. ACHESS DESCRIPTION

AChess is a chess-like game. In the game, players are allowed to move as many pieces as they wish at any point during the game. It has the same board, the same pieces, and the same piece movement patterns as standard chess. The main difference between AChess and regular chess is that it removes the rule in which players must take turns. There are other differences between AChess and standard chess that were found to be necessary to make AChess a well-defined game. These differences include:

1. Any piece may be moved at any time as long as the piece is not already in the process of moving.
2. Pieces, including knights, traverse paths through squares to reach their destinations instead of teleporting.
3. Pieces traverse individual squares at specified rates that are specified based on the piece type (i.e. King, Queen, Bishop, Knight, Rook, Pawn).
4. Pieces cannot change directions while a move is being carried out. Agents must wait for a piece to finish its current movement before submitting a new move for the piece.
5. There is no check-mate. Kings must be captured.

3. ACHESS FEATURES AND CHALLENGES

AChess is a fully observable environment. A central server acts as a referee and supplies both players with a complete view of the game environment. This view includes the current positions of all the pieces and which pieces are currently in the process of moving. The only information agents do not have is where the moving enemy pieces are moving to and exactly how long it has been moving.

AChess is a stochastic environment. There is randomness in the AChess system. AChess simulates continuous time with extremely fine-grain discrete time steps. In the event that two or more pieces attempt to enter a square at the same time, the server randomly chooses one to enter the square. The other pieces are either killed or prevented from

moving. This challenges agents to handle the uncertainty in the environment.

AChess is an episodic environment. Agents in competition will compete in a set of matches consisting of a series of games between two agents. This will give agents with learning capabilities a chance to repeatedly measure their performance against competitors and learn from their experience.

AChess has an extremely large state and search space that the agents must manage. A normal game of chess has a search space size on the order of 10^{40} [1]. AChess has an exponentially larger search space because of the ability to move more than one piece, and potentially all the pieces, asynchronously. In addition, pieces can be in a state of movement (*not even on a square!*) which drastically increases the size of the state space.

Finally, AChess challenges agents with heterogeneous capabilities. The agents must effectively utilize pieces with different movement rates and movement patterns. This challenge is missing in many other competitive agent environments and is one of the more interesting features of AChess.

4. ENVIRONMENT ARCHITECTURE AND LOGISTICAL REQUIREMENTS

The AChess environment/server is implemented in Java and communicates to the agents with strings of text over standard TCP/IP sockets. This enables interested parties to develop their agent solutions using any language or platform they choose. It also allows the agents, environment, and visualizations to be resident on separate platforms, which in turn helps to prevent competitors from interfering with each other or the performance of the environment itself.

For the competition the matches will be run across four machines, two of which will be identical platforms for the agents (each team), over a local area network. One machine will host the AChess server (the environment), and the last machine will host the visualizations. Four machines are required to enable each agent, the server, and the visualization to have their own computing platform. The machines will all be using the same operating system and be configured to be identical platforms for all competitors. The final decision on the operating system will be made at a later date and will be made based on competitor preference and logistic considerations.

5. COMPETITION DESCRIPTION

The competition will be run as a tournament consisting of rounds. Agent implementations will compete against one another in matches which will consist of a specified number of games. The final number of games in a match will be decided just before the start of the tournament and will be based on the number of entries. A large number of games per match are preferred, but the number of games may need to be small in the interest of time. The winner of a match will be determined by which agent was able to accumulate the most wins during the course of the match. Winners of matches will proceed to the next round until ultimately there is a single winner.

For the conference we will invite any and all interested parties to enter their own AChess agent into the competition. Well before the conference a version of the AChess environment will be provided to the public (via download) along with an API for agent development. Competitors will have to submit a final version of their agent to be used in the competition by a specified date.

In the competition the board will be the standard 8 by 8 square sized board. Pieces will be positioned in the standard chess layout at the start of the game. The piece rates, which determine how quickly pieces may traverse individual squares, will be defined at a later date.

6. REFERENCES

[1] M. Beeler, R. W. Gosper, and R. Schroeppel. Hakmem. Technical report, Massachusetts Institute of Technology (TR AIM-239), Cambridge, MA, USA, 1972.

[2] N. Gemelli, R. Wright, and R. Mailer. Asynchronous chess. In *Proceedings of The 2005 AAAI Fall Symposium Workshop on Coevolutionary and Coadaptive Systems*, November 2005.

Deployment of A-globe Multi-Agent Platform

David Šišlák, Martin Rehák, Michal Pěchouček and Dušan Pavlíček

Agent Technology Group, Gerstner Laboratory
Department of Cybernetics, Czech Technical University
Technická 2, Prague, 166 27, Czech Republic

sislakd@fel.cvut.cz, rehakm1|pechouc|pavlicek@labe.felk.cvut.cz

ABSTRACT

$\mathcal{A}$-**globe** [2, 1] is a freeware, simulation-oriented multi-agent platform featuring agent migration, communication inaccessibility simulation and high scalability with moderate hardware requirements. Using dedicated simulation messaging together with 2D and 3D visualization support, large agent systems can be engineered, tested and visualized on a single machine. $\mathcal{A}$-**globe** agents are fully-fledged JAVA agents, each with its own independent thread, that can autonomously migrate between platforms running on different hosts. Thanks to the separation of simulation and agent code, deployment of agents to embedded devices is straightforward. The platform is not natively FIPA-compliant, as the inter-operability was sacrificed to offer superior scalability and efficiency.

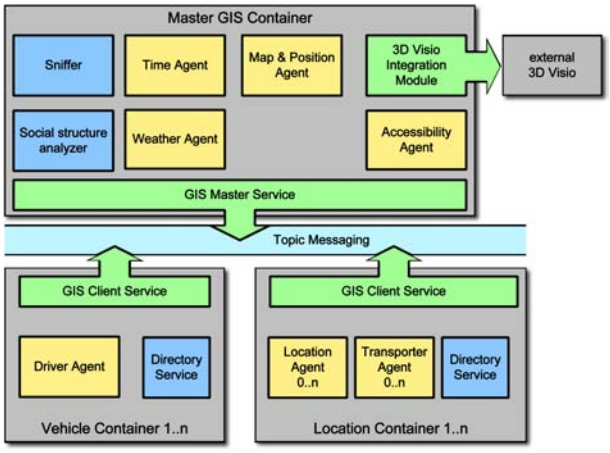


Figure 1: $\mathcal{A}$-globe based simulation.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; I.6.3 [**Simulation and Modeling**]: Applications

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

General Terms

Performance, Design, Experimentation

Keywords

Multi-Agent Platform, Simulation, Visualization



Figure 2: ACROSS scenario visualization.

1. A-GLOBE SIMULATION SUPPORT

$\mathcal{A}$-**globe** includes a special infrastructure for environmental simulation. Besides the *actor agents* that play roles in the simulated world, *environment simulation agents* (ES) are located in a dedicated master container. Instead of using messages, they communicate with actor agents by *topic messaging*, container-to-container messaging specifically reserved for environmental simulation, as shown in Fig. 1. Topic messaging is managed by GIS Service – a special $\mathcal{A}$-**globe** service to which the actor agents subscribe to receive various topics. If such a topic is received by the container, it is distributed to all subscribed agents. Note that all agents in the container receive the same data, because the environment perception shall be identical for all collocated agents. In addition, the agents who wish to act on the environment can submit topics to the GIS service. These topics are then sent to all ES agents in the master container subscribed to receive the topic. ES agent can be responsible for nearly any simulation layer. However, *accessibility agents*, who control the existence of communication links between agent containers, occupy a privileged place between ES agents. $\mathcal{A}$-**globe** message layer uses this information to actually prevent messaging between inaccessible nodes. Simulated accessibility can depend on many factors, typically including the distance.

2. ACROSS DEMONSTRATION

To illustrate the concepts of the platform design we will present the ACROSS scenario – Agent Complex Reasoning Simulation System that uses the $\mathcal{A}$-**globe** features to create a relatively rich world where diverse types of agents can interact. A simplified economy on an island based on Java (Indonesia) geography is simulated, see Fig. 2 and 3. Currently, we use this scenario as a common base for multiple research targets, where we need to investigate interactions between a relatively high number of fully-fledged agents. In the scenario we solve a logistics problem in a non-collaborative environment with self-interested agents. Agents that are part of the scenario have no common goals and their cooperation is typically financially motivated.

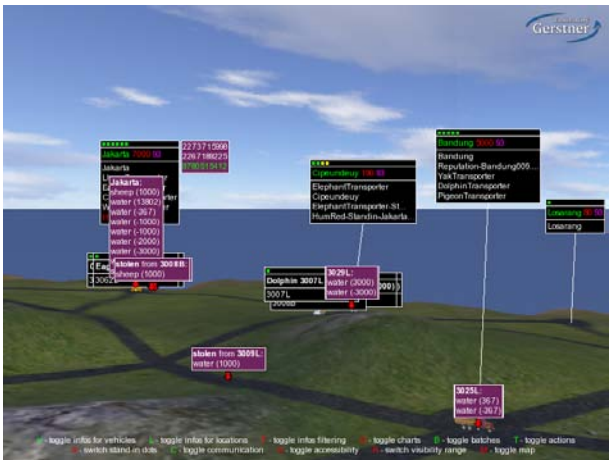


Figure 3: Representation of physical cargo in the ACROSS scenario.

3. ATC DEMONSTRATION

Our second demonstration illustrates how agents can be used for the negotiation-based de-confliction in the Air-Traffic Control (ATC), Fig. 4 and 5. The agents are responsible for aircraft simulation and airways planning. The core of the system consists of (1) the server component – that simulates positions of aircraft and other objects in the simulated world, aircraft hardware, weather conditions, communication ranges; and (2) the plane component – agents simulating aircraft behavior. One of them – a *pilot agent* is the main control unit of the simulated aircraft handling negotiation-based de-confliction. It processes radar data and communicates with the respective pilot agent. Each plane has its own flight mission which can be time constrained.

The system has been integrated with three external geographical and tactical data sources available online: NASA, U.S. Geological Survey (USGS) and Geographic Names Information System (GNIS) database. The system also provides real-time 2D/3D visualization: either locally and/or via a remote web client running inside any compliant internet browser.

4. CONCLUSION

Both demonstrated systems illustrate unique $\mathcal{A}$-**globe** features: superior scalability and efficiency, that allows us to define complex agent societies on restricted resources and advanced simulation support that clearly separates simulation and visualization from agent decision-making.

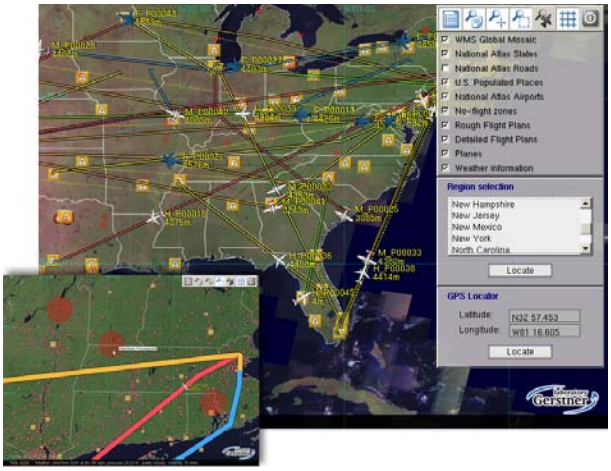


Figure 4: ATC system integrates several external data sources.

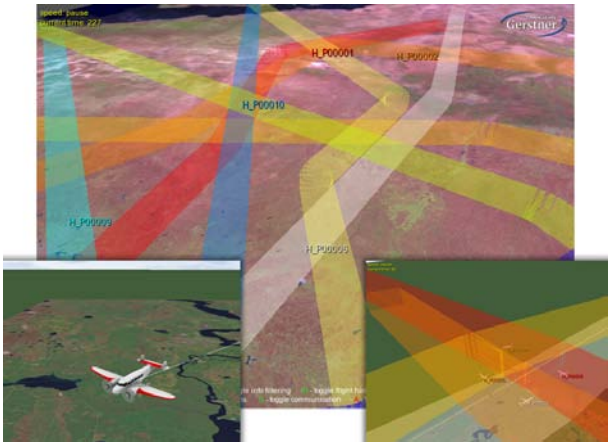


Figure 5: De-conflicted air-traffic control among several autonomous aerial vehicles.

Acknowledgement

Effort sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-04-1-3044, FA8655-04-1-3044-P00001 and FA8655-02-M-4057.[1]

5. REFERENCES

[1] A-globe. A-globe Agent Platform. `http://agents.felk.cvut.cz/aglobe`, 2006.

[2] D. Šišlák, M. Rehák, M. Pěchouček, M. Rollo, and D. Pavlíček. $\mathcal{A}$-**globe**: Agent development platform with inaccessibility and mobility support. In R. Unland, M. Klusch, and M. Calisti, editors, *Software Agent-Based Applications, Platforms and Development Kits*, pages 21–46, Berlin, 2005. Birkhauser Verlag.

[1]The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

UAV Airspace Management System *UAMS* *

Regis Vincent
SRI International
333 Ravenswood Ave
Menlo Park, CA 94403
vincent@ai.sri.com

Osher Yagdar
SRI International
333 Ravenswood Ave
Menlo Park, CA 94403
yagdar@ai.sri.com

Andrew Agno
SRI International
333 Ravenswood Ave
Menlo Park, CA 94403
agno@ai.sri.com

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*coherence and coordination, intelligent agents, languages and structures, multiagent systems.complexity measures, performance measures*

Keywords

multi-agent simulation, air space management, robotics

1. UAV COLLISION AVOIDANCE

As technology advances, unmanned aerial vehicles (UAVs) are capable of more complex missions. Miniaturization results in a migration of capability from larger to smaller UAV platforms and in cost reduction. According to the DoDs Unmanned Aircraft Systems Roadmap 2005-2030, military UAVs are now required to execute intelligence, surveillance, reconnaissance, strike, suppression of enemy air defense, electronic attack, communications, aerial delivery, and resupply missions. It is therefore evident that UAV developments require a more intelligent and autonomous airspace and mission management approach then ever.

The modern battlespace may be crowded with urban buildings, manned and unmanned aircraft, restricted operational zones, and direct and indirect fire. For UAVs to avoid colliding in such a crowded environment is challenging. We suggest an intelligent way to detect potential collisions and find alternative routes. Our suggested solution is based on centralized and distributed coordination approaches. The autonomous airspace management system will choose, in real time, the suitable solution that will assure collision avoidance.

The airspace management system will consider the following approaches:

1. Centralized, in which an airspace management center detects possible collisions and applies alternative routes for them
2. UAV deconfliction, in which the airspace management center detects potential collisions but allows the UAVs to find alternatives
3. UAV detection, in which the airspace management center informs each UAV of its neighbors, thus allowing the UAV to detect and adjust its own flight plan
4. Distributed, in which the airspace management center is completely incommunicable, and the UAV must become aware of its neighbors, detecting and deconflicting prospective collisions
5. Isolated, in which there is no communication between the airspace management center and the UAVs or between UAVs

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

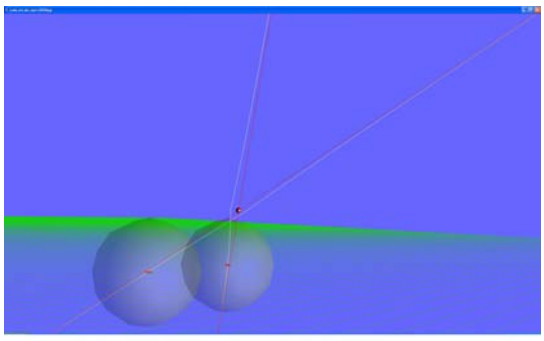
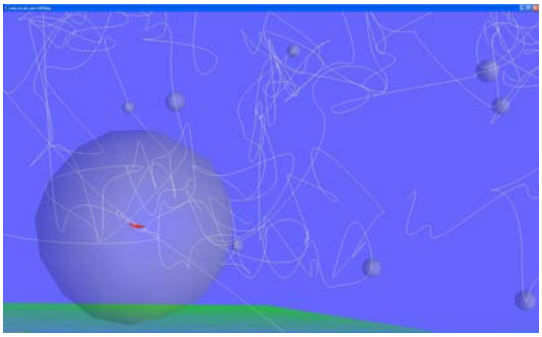
2. SYSTEM APPROACH AND PATH PLANNING

Our UAV path planning is defined by complex Bezier curves that express the physical constraints of a UAV, such as maximum speed, minimum turning radius, velocity, and acceleration. Two points and two vectors representing the entrance and exiting speed of the UAV define a Bezier curve. In this demonstration, each UAV has its own mission defined as a tuple of destination (longitude, latitude, altitude), deadline, and utility. The utility is the dimension we sum across all the UAVs to estimate the overall success. A mission is achieved if the UAV reaches its destination before the deadline. There is no partial utility, because each mission is achieved or not achieved. At the beginning of the simulation, all UAVs have a valid path (not necessarily conflict free) that meets the deadline. The first step of the algorithm is to detect if there is a possible conflict in any UAV paths. We use an octree algorithm to quickly find all conflicts. A conflict is composed of a 4D point in space (longitude, latitude, altitude) and time and all UAV paths that intersect at this 4D point. The list of conflicts is then passed to a solver that finds new conflict-free paths. This process is obviously iterative, because choosing a different path may result in one or more conflicts.

To add complexity, we have introduced a manned aircraft (controlled by a joystick connected to the system) that each UAV must avoid while maintaining the maximum number of deadlines and missions. The manned aircraft broadcasts only its position and heading, and all the UAVs must adapt their path to not collide with it. To handle the manned aircraft, each UAV considers a set of possible paths and checks all of them for possible conflicts.

Our demonstration is composed of two laptops. One laptop runs the manned aircraft user interface with the joystick; the other laptop runs the simulation and our deconfliction algorithm. The accompanying screenshots illustrate the simulation and user interface

Table 1: default

	
Conflict detected and proposed path changes hline	Many paths, many conflicts. The sphere is the communication range.
	
Realistic simulation in downtown San Francisco	

3. ACKNOWLEDGMENTS

Effort partially sponsored by a Government Prime Contract, under agreement number W911W6-05-C-0009. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government.

Multi-Agent Coordination and Control Testbed for Planning and Scheduling Strategies

Paul Verstraete, Paul Valckenaers, Hendrik Van Brussel, Karuna Hadeli, Bart Saint Germain
Department of Mechanical Engineering
K.U.Leuven University
Leuven, Belgium
{paul.verstraete, paul.valkenaers, hendrik.vanbrussel, hadeli.karuna, bart.saintgermain}@mech.kuleuven.be

ABSTRACT

The Multi-Agent Coordination and Control (MACC) testbed is a modelling and simulation environment for manufacturing control. It provides benefits to both the research community and industry. It uses a multi-agent manufacturing execution system, developed at PMA.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence-Multiagent systems

General Terms

design, experimentation

1. INTRODUCTION

The Multi-Agent Coordination and Control (MACC) testbed is a modelling and simulation environment for manufacturing control.

Manufacturing control is the operational level of production planning and control. It is the decision making activity concerned with the short-term and detailed assignment of operations to production resources.

The testbed uses a multi-agent Manufacturing Execution System (MES), developed at K.U.Leuven PMA [1]. A MES is a manufacturing control information system that supports the execution on the shop floor. This support ranges from tracing the execution to supervising the process control systems, the routes that products follow and the assignment of operations to production resources.

Current MES implementations do not support handling of disturbances. Disturbances are handled by operators on the shop floor today. If deliveries are late, clients change orders or machines break down, they try to deal with the problem, respecting the given requirements as good as possible (e.g. due date).

Operators on the shop floor however only have limited insight on how the rest of the system will react as a consequence of their decisions. Therefore they mainly rely on their experience in dealing with the disturbances they encounter.

Using multi-agent technology, the PMA multi-agent MES supports operators in taking these decisions, improves their insight in the consequences these decisions will have and allows for self-organization.

The testbed consists out of several tools:

- a modelling tool
- a web-based simulation service
- a simulation log browser

Results of the simulation can be twofold: studying the dynamics of the modelled systems or calculating performance criteria.

The testbed tools are written in Java and run on any platform supporting Java.The tools can be started from the website without seperate software installation (http://www.mech.kuleuven.be/benchmarking).

2. TESTBED ARCHITECTURE

The architecture of the testbed is structured into three main software components:

2.1 Test-Bench Assistant

The Test-Bench Assistant is a visual environment for assisting the designer of a test-bench case in inputting all the main data of the industrial case he/she wants to propose to the scientific and industrial community.

The data, describing an industrial case, comprise [2]:

- The production system taken as a reference (production system model):
  - the structural features: the physical configuration of the production system.
    - the physical components (workstations, transporters, storage) and their capabilities (sawing, milling, ...).
    - the layout of the plant
  - the technological features: the process plans of the products being processed in the plant
  - optionally, a planning indicating start, end times and the machine to be used for the various operations.
- Dynamic information about the industrial case :

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

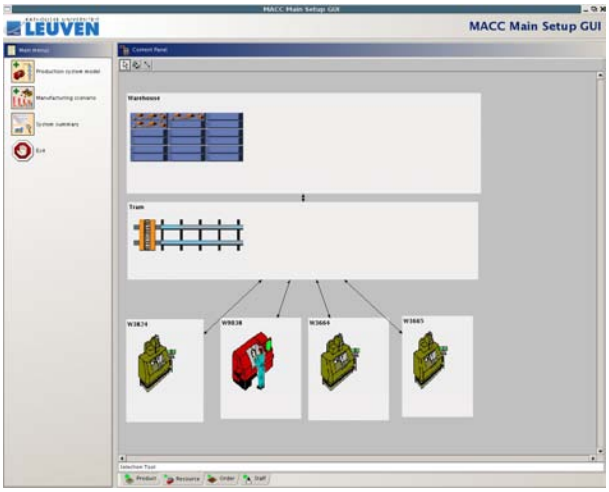


Figure 1: The test bench assistant

– The plant scenario:

events or activities related to the functioning of manufacturing physical components (e.g. machine breakdowns, stochastic variations on setup times and operation processing times)

– The operational scenario:

* Production orders
* Expected due dates and release dates
* Disturbances like stock breackage, or changes in orders, due dates or release dates.

2.2 TestBed Simulator

A web-based simulation tool for the experimentation, testing and performance analysis of submitted scheduling strategies (without visualisation). The simulation tool takes the input of the test-bench assistant and generates a simulation log, describing the state of the factory and the short term forecasts through the simulation run. Any performance measures can be calculated, based on this log.

2.3 The log browser

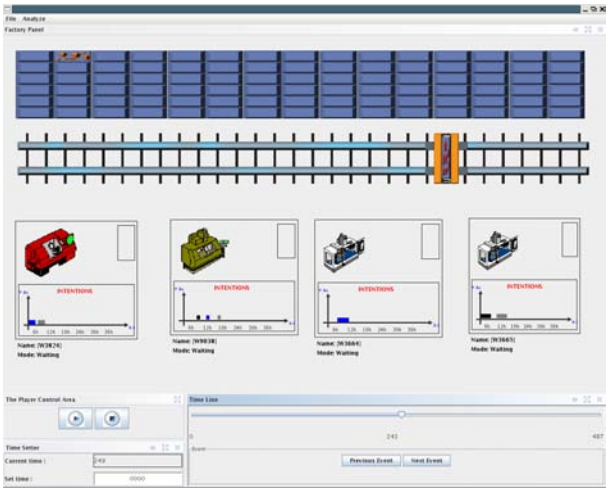


Figure 2: The log browser

The log browser visualizes the simulated factory, the production execution and the short-term forecasts.

A short term forecast indicates the work that has to be carried out in the next shift(s). This forecast automatically adapts in case of changing circumstances (new orders, delays in execution, missing parts, ...).

3. BENEFITS

The testbed provides the following benefits:

- Providing a common base for integrating manufacturing control and scheduling strategies into existing systems.

  The case model has to be modelled only once and can be combined with different control strategies. If new strategies are available, they can be tested on the case.

  The control strategy does not have to solve the whole problem. If no solution is given for part of the problem, the agents in the MES self-organize.

- Provide more insight to operators on the shop floor when taking decisions.

  The short term forecast described earlier, supports operators by providing the possible short-term future states of the underlying system. When changing jobs sequences or the starting time of a job (e.g. for maintenance), every entity involved gets an updated forecast.

- Provide more insight to developers in the applicability of their algorithm in various industrial cases.

  The testbed allows to collect industrial cases and to make them available to the research community. This shortens the time for conducting experiments significantly and increases the industrial relevance of the results.

4. FUTURE WORK

Future efforts aim to expand the possibilities for integrating planning strategies into the MES. For now, the MES follows the planning if it is feasible. If not, it searches a feasible alternative. Subject of further work is to allow the MES to search the feasible alternative that corresponds the most to the original planning [3].

Furthermore, the design of the MES allows to provide plug-ins, e.g. a custom scheduler for a physical component or a custom strategy for chosing between alternatives for a production order. In the future, the test-bed assistant will provide a graphical interface for this task.

This manuscript presents work supported by the K.U.Leuven Research Fund (Concerted Research Action on Agents for Coordination and Control).

5. REFERENCES

[1] P Valckenaers and H Van Brussel, “Holonic manufacturing execution systems,” *CIRP ANNALS-MANUFACTURING TECHNOLOGY*, vol. 54, no. 1, pp. 427 – 432, 2005.

[2] S. Cavalieri, L. Bongaerts, M. Taisch, M. Macchi, and J. Wyns, “A benchmark framework for manufacturing control,” in *Proc. of the Second International Workshop on Intelligent Manufacturing Systems*, September 1999.

[3] P. Verstraete and P. Valckenaers, “Cooperating planning and manufacturing execution systems,” in *INCOM*, 2006, submitted.

A Texas Hold'em poker player based on automated abstraction and real-time equilibrium computation*

Andrew Gilpin
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
gilpin@cs.cmu.edu

Tuomas Sandholm
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
sandholm@cs.cmu.edu

ABSTRACT

We demonstrate our game theory-based Texas Hold'em poker player. To overcome the computational difficulties stemming from Texas Hold'em's gigantic game tree, our player uses automated abstraction and real-time equilibrium approximation. Our player solves the first two rounds of the game in a large off-line computation, and solves the last two rounds in a real-time equilibrium approximation. Participants in the demonstration will be able to compete against our opponent and experience first-hand the cognitive abilities of our player. Some of the techniques used by our player, which does not directly incorporate any poker-specific expert knowledge, include such poker techniques as bluffing, slow-playing, check-raising, and semi-bluffing, all techniques normally associated with human play.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: General

General Terms

Algorithms, Economics

Keywords

Keywords: game theory, equilibrium computation, game playing

1. INTRODUCTION

In environments with multiple self-interested agents, an agent's outcome is affected by actions of the other agents. Consequently, the optimal action of one agent generally depends on the actions of others. Game theory provides a normative framework for analyzing such strategic situations. In particular, game theory provides the notion of an *equilibrium*, a strategy profile in which no agent has incentive to deviate to a different strategy. Thus, it is in an agent's interest to compute equilibria of games in order to play as well as possible.

Games can be classified as either games of *perfect information* or *imperfect information*. Chess and Go are examples of the former, and, until recently, most game playing work in AI has been on games of this type. To compute an optimal strategy in a perfect information game, an agent traverses the game tree and evaluates individual nodes. If the agent is able to traverse the entire game tree, she simply computes an optimal strategy from the bottom-up, using the principle of *backward induction*. This is the main approach behind minimax with α-β-pruning. These algorithms have limits, of course, particularly when the game tree is huge, but extremely effective game-playing agents can be developed, even when the size of the game tree prohibits complete search.

Current algorithms for solving perfect information games do not apply to games of incomplete information. The distinguishing difference is that the latter are not fully observable: when it is an agent's turn to move, she does not have access to all of the information about the world. In such games, the decision of what to do at a node cannot generally be optimally made without considering decisions at all other nodes (including ones on other paths of play).

The *sequence form* is a compact representation [7, 5, 10] of a sequential game. For two-person zero-sum games, there is a natural linear programming formulation based on the sequence form that is polynomial in the size of the game tree. Thus, reasonable-sized two-person games can be solved using this method [10, 5, 6]. However, this approach still yields enormous (unsolvable) optimization problems for many real-world games, most notably poker. In this research we apply automated abstraction techniques for finding smaller, strategically similar games for which the equilibrium computation is faster. The resulting strategies can then be used as approximate solutions to the original game. We have chosen poker as the first application of our equilibrium approximation techniques.

2. POKER

Poker is an enormously popular card game played around the world. The 2005 World Series of Poker featured more than $100 million dollars in prize money in several tournaments. Increasingly, poker players compete in online poker rooms, and television stations regularly broadcast poker tournaments.

*This material is based upon work supported by the National Science Foundation under ITR grants IIS-0121678 and IIS-0427858, and a Sloan Fellowship.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

Due to the uncertainty stemming from opponents' cards, opponents' future actions, and chance moves, poker has been identified as an important research area in AI [2]. Poker has been a popular subject in the game theory literature since the field's founding, but manual equilibrium analysis has been limited to extremely small games. Very recently, there has been considerable progress in tacking larger games. In a recent paper [4], we developed automated abstraction techniques, and applied them in computing *optimal* strategies for Rhode Island Hold'em poker [9], a smaller version of Texas Hold'em that is still over four orders of magnitude larger than previously solved poker games.

2.1 Texas Hold'em

Texas Hold'em is perhaps the most popular version of poker. It is the game that is used to determine the world champion at the annual World Series of Poker. In the demonstration we will be playing *heads-up*, in which there are just 2 players (in this case, a human player versus our player). The players alternate turns being player 1 and player 2. Player 1 is considered the *small blind*, and player 2 is the *large blind*. Before any cards are dealt, the small blind contributes one chip to the pot, and the large blind contributes two chips to the pot. Both players then receive two cards each, face down; these are known as the *hole cards*.

After receiving the hole cards, the players take part in one betting round. The small blind goes first. Each player may check or bet if no bets have been placed. If a bet has been placed, then the player may *fold* (thus forfeiting the game), *call* (adding chips to the pot equal to the last player's bet), or *raise* (calling the current bet and making an additional bet). In Texas Hold'em, the players are usually limited to four raises each per betting round. In this betting round, the bets are in increments of two chips.

After the betting round, three community cards are dealt face up. These cards are called the *flop*. Another betting round take places at this point, with bets equal to two chips.

Another community card is dealt face up. This is called the *turn* card. Another betting round takes place at this point, with bets equal to four chips.

A final community card is dealt face up. This is called the *river* card. Another betting round takes place at this point, with bets equal to four chips.

If neither player folds, then the *showdown* takes place. Using the seven available cards (the two hole cards and five community cards), the players form their best 5-card poker hands. The player who has the best 5-card poker hand takes the pot. In the event of a draw, the pot is split evenly.

3. TECHNICAL OVERVIEW

The main contribution of our work is the application of automated abstraction techniques to a real-world game. Previous work has been limited to much smaller games. In this section we give a brief overview of our development of a Texas Hold'em poker player. A detailed description of our player is available in a separate paper [3]. There are two types of abstraction employed in our approach: state-space abstraction and round-based abstraction.

In our previous work [4] we developed techniques for *automatically* reducing the size of a game tree (a form of state-space abstraction) in order to make equilibrium-finding algorithms practical. We apply our algorithm, *GameShrink*, to the various game trees we encounter in the computation of strategies.

In addition to state-space abstraction, we also employ round-based abstraction. In our approach, we first solve for an approximate equilibrium for a truncated game involving only the first two rounds. We do this by solving a large linear program in an off-line computation. After the turn card appears, our player computes updated card probabilities based on observed behavior, and then computes an equilibrium approximation for the third and fourth rounds in *real-time*. The abstractions we employ in computing this equilibrium approximation are *dynamically* determined based on the information (*i.e.* community cards) revealed so far in the game. This allows our computation to focus on the specific portion of the game tree relevant to the current hand.

Round-based abstraction has been used in previous poker work [9, 1]. The primary difference with our approach is the fact that strategies are computed dynamically, using observed information to achieve a closer approximation. Furthermore, the sizes of the individual models are larger. For example, optimal strategies for pre-flop Texas Hold'em have been computed [8]. This approach requires modelling 169 distinct hands. Our model not only considers 169 hands in the first round, but also 2465 hands in the second round. Solving this model requires 18.8 GB of RAM and takes 7.1 days. In addition, our abstractions are automatically computed, rather than manually designed by an expert.

Some features of our computed strategies include poker techniques such as bluffing, slow-playing, check-raising, and semi-bluffing, all techniques normally associated with human play. In this demonstration, participants will compete with our opponent and will experience these strategies first-hand.

4. REFERENCES

[1] D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg, and D. Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

[2] D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. The challenge of poker. *Artificial Intelligence*, 134(1-2):201–240, 2002.

[3] A. Gilpin and T. Sandholm. A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation. Mimeo, 2006.

[4] A. Gilpin and T. Sandholm. Finding equilibria in large sequential games of imperfect information. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, Ann Arbor, MI, 2006.

[5] D. Koller, N. Megiddo, and B. von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, 14(2):247–259, 1996.

[6] D. Koller and A. Pfeffer. Representations and solutions for game-theoretic problems. *Artificial Intelligence*, 94(1):167–215, July 1997.

[7] I. Romanovskii. Reduction of a game with complete memory to a matrix game. *Soviet Mathematics*, 3:678–681, 1962.

[8] A. Selby. Optimal heads-up preflop poker, 1999. http://www.archduke.demon.co.uk/simplex/.

[9] J. Shi and M. Littman. Abstraction methods for game theoretic poker. In *Computers and Games*, pages 333–345. Springer-Verlag, 2001.

[10] B. von Stengel. Efficient computation of behavior strategies. *Games and Economic Behavior*, 14(2):220–246, 1996.

Decision-Making Support for Maritime Business

Jonathan Himoff
CEO of Magenta Technology
130 Shaftesbury Ave, London W1D 5EU, UK
Telephone number: +44 207 149 3751
jon.himoff@magenta-technology.com

ABSTRACT
A near real time multi agent based system for scheduling ocean fleets has been developed. The ocean scheduling problem has relatively low numbers of cargos and vessels to deal with but it is highly complex in terms of constraints and numbers of options. It also has a high rate of events affecting the schedule and different fleet operators have differing goals for the parameters of their schedule. This variability is dealt with by storing knowledge of this in ontology.

Categories and Subject Descriptors
H.4.2 [**Information Systems Applications**]: Types of Systems – *decision-support, logistics.*

General Terms
Management, Decision-Making, Measurement, Performance, Design, Economics.

Keywords
Multi-agents, semantic web, decision making logic, ontology, agent negotiations, virtual market, event-driven continuous planning, near real time incremental planning.

1.INTRODUCTION
Based on the multi-agent and semantic web technologies, Magenta's Ocean i-Scheduler exhibits behavior that is fundamentally different from behavior of conventional schedulers. For example, it: refers to business domain knowledge while generating schedule options; reviews and if necessary modifies previously made decisions when a problem situation changes; plans users' activities (sequences, options, iterations) taking into account user defined goals, strategies and constraints; designs and conducts experiments to improve decision making; constructs models of problem situations; explains its decisions to users and works on scheduling continuously improving the specified enterprise value without a need to be switched on by users.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8–12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005…$5.00.

2.DEMONSTRATED FEATURES
The Ocean planning and scheduling domain is characterized by:

- Typically a few (<10) orders per day to be planned
- One vessel carries one cargo
- One voyage has one collect and one drop operation
- Refueling (bunkering) is a critical operation
- Canals (Suez, Panama) have specific logic of operation
- All vessels have their own, unique parameters
- Voyages times measured in a few to over 100 days
- Many options to consider in planning, with scope to make changes to plans
- Multiple objectives for fleet schedule (minimize idle days, minimize days when ship carries ballast rather than cargo, and maximize voyage profits etc.)

The Ocean i-Scheduler uses agents operating in a virtual market place to generate options for assigning cargos to ships and options for dealing with events that affect the schedule. Events that can be handled include:

- arrival of new cargo
- cargo parameters changed
- vessel late
- vessel early
- cancel cargo
- vessel breakdown
- re-schedule cargo
- owner events such as Off-hire or Dry-dock

The system has agents representing vessels, ports, cargoes, and customers. They operate in a virtual market and use ontology to understand parameters of the fleet, ports and canals, and to understand the objectives of the fleet as a whole. Events are planned by creating a new agent to represent that event. Vessels which are potentially affected by the event are selected. For each selected vessel a new variant of the schedule is built. The variants are built using a journey time matrix and a set of scheduling algorithms. The scheduling algorithms are hierarchical in nature. First an Open Slot' search is carried out. Is there a vessel available to take the cargo at the desired time? If not is it possible to 'Shift' an existing cargo in time so that the vessel can then take the new cargo? If not is it possible to 'Swap' the cargo with another cargo and then reschedule the replaced cargo? This can involve a deep combinatorial analysis of swap options. The variants are ranked according to the decision making logic (DML) in the ontology. The DML is user adjustable and can be weighted towards optimizing several objectives as described above.

The agent negotiations are an auction type competition for resources. The market has a concept of virtual money that the

agents use in their negotiations. Demand agents bid for resources using their money. Decisions are made based on all agents trying to achieve the best return for themselves. The agents are created and destroyed on demand as events occur and new schedules accepted.

This incremental planning approach to new events allows near real time responses to events (typically options are generated in a few tens of seconds) and presents a human scheduler with a set of ranked options to choose from.

2.1System Features Based on the Multi-Agent and Semantic Web Technologies

- Near real time incremental planning
- Decision making logic for defining user-specific preferences. This would typically be a trade of between minimizing costs, maximizing revenue, and maximizing customer service or service level agreements.
- Support of ocean-specific constraints (e.g. sizes of vessels versus port capabilities and canal sizes)
- Key Performance Indicator (KPI) calculation mechanism (fleet, vessel and voyage KPIs) and possibility to define, store and add new KPIs in ontology
- Event-driven continuous planning of fleet resources as demands (new orders) and fleet events occur
- Scheduling/rescheduling vessels according to "maximize profit of the fleet as a whole" criteria
- Support of scheduling of majority of ocean fleet events.

2.2User Interaction with the System

- Rich visual and text based reports
- Support of main users: executive team, scheduler team, chartering team, IT team;
- Decision making audit trail that keeps records;
- Decision making queue that provides strong visualizations, simulations and analytics to help users compare schedule options, drill into details, consider how to overcome potential constraints and then make decisions that will be immediately be updated into fleet schedule;
- Web-based UI with multi-user environment.

3.RELATED PROJECT

The first commercial customer for Magenta's Ocean i-Scheduler is Tankers International (TI), one of the largest VLCC (Very Large Crude Carriers) oil tanker pools in the world which is using Magenta's multi-agent technology to schedule its fleet in real time. The oil transportation market is extremely volatile with transportation costs fluctuating on a daily basis. Therefore, TI has to be able to deal with unexpected circumstances and operate from an optimized schedule. The Magenta's Intelligent Fleet Scheduler allows TI to assess potential cargoes in the marketplace, match these against its vessels' specifications and calculate voyage profitability. Within TI scheduling system, Magenta's multi-agent technology works by allocating a single agent to each element of entire fleet. Unlike other traditional systems, agents are also able to resolve conflicts as they are not bound by rigid rules and work around problems. Furthermore, having presented the best solutions to the operations team, the agents continue to work to provide alternative solutions to ensure that any unforeseen factors, such as changing freight rates or vessel positions, are reacted to dynamically.

The multi-agent Intelligent Fleet Scheduler allows Tankers International to

- Dynamically schedule ships in a fleet to respond to new cargo opportunities in real time; plan/re-plan to find options and suggest a decision, support decision-making process for planning and implementing of these plans; react to unforeseen changes in contracts, coming of new contracts or canceling of existing ones; enable cooperation of several vessels for executing of one order;
- Capture scheduling knowledge that was previously kept in the head of one scheduling specialist;
- Simulate complex dynamic solutions on the market of oil transportation for the estimation of effectiveness of made decisions: "what-if"?
- Simulate individual behavior models of tankers, ports, routes and orders: agents find knowledge about these objects in their ontologies and flexibly adapt themselves to the current situation.

4.CONCLUSIONS

The following features provided by using the multi-agent technology will be demonstrated during the demo session:

- Near real time scheduling;
- Capturing of business domain knowledge and decision-making logic in the ontology;
- Scheduling engine supports complex planning features in a flexible way: canal passage, bunkering operation;
- Support of wide range of fleet events;
- Powerful mechanisms for KPIs calculation;
- Easy to play "what-if" games;
- Current workflow allows analyzing/comparing options and market opportunities;
- Web-based user-specific reports and dashboards
- Teamwork support.

Participatory, Embodied, Multi-Agent Simulation

Paulo Blikstein William Rand Uri Wilensky
Northwestern University
2120 N. Campus Dr.
Evanston, IL 60208 USA
+1 (847) 467-7329
{paulo, wrand, uri}@northwestern.edu

ABSTRACT
We will demonstrate the integration of a software-based multi-agent modeling platform with a participatory simulation environment and real-time control over a physical agent (robot). Both real and virtual participants will be able to act as agents in a simulation that will control a physical agent. The backbone of this demonstration is a widely used, freely available, mature modeling platform known as NetLogo.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *coherence and coordination, intelligent agents, languages and structures, multiagent systems.*

General Terms
Design, Experimentation, Human Factors, Languages.

Keywords
multi-agent simulation, participatory simulation, embedded agents, robotics, sensors

1. WHY IS INTEGRATION GOOD?
Multi-agent simulation is a powerful technique used to encode real-world simple rules in virtual agents and then simulate their interactions [1]. Participatory simulations are similar to multi-agent simulation except individuals play the role of the virtual agents, sometimes in combination with the virtual agents [2]. In addition, an extensive amount of multi-agent research has gone into examining how to embody agent control within a physical entity [3]. All three of these areas are concerned with the creation, manipulation, and development of agents in one form or another. Thus combining these three disparate systems in to one unified platform would be useful. Of course, multi-agent simulation platforms, participatory simulation systems, and robotic systems have all been demonstrated to a large degree in the past. However many of these systems were not designed to be integrated, and thus combining these existing platforms is difficult if not impossible.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS '06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005…$5.00.

We will demonstrate a single platform (NetLogo) that supports simulated agents, participatory agents and physical agents [4]. Moreover this platform is freely available, widely used, and easy to learn. By enabling all three types of interaction to be combined in one modeling environment NetLogo facilitates development of systems and simulations that combine these aspects without the need for learning multiple platforms and then attempting to connect them to each other.

2. A MULTI-AGENT MODELING PLATFORM (NETLOGO)
NetLogo is a mature multi-agent modeling platform. It has been developed for over 8 years at the Center for Connected Learning and Computer Based Modeling which is under the leadership of director Uri Wilensky and is currently located at Northwestern University [4]. NetLogo has an estimated twenty thousand current users. With NetLogo, one can model complex systems with thousands of interacting agents, and study the connection between the micro-level rules and the macro-level "emergent" patterns. Agents can be either moving entities or stationary cells as in a cellular automaton. Thousands of these agents follow rules in the simulated system, acting in parallel and affecting other agents, both moving and stationary. NetLogo has found use by many natural and social scientists as a research tool and has also been adopted as a successful element of curriculum by hundreds of educational institutions. The emergent phenomena modeled span a wide range of domains including ecosystems, economies, organizational change, and molecular interactions and reactions. Figure 1 illustrates a typical NetLogo model of flocking behavior.

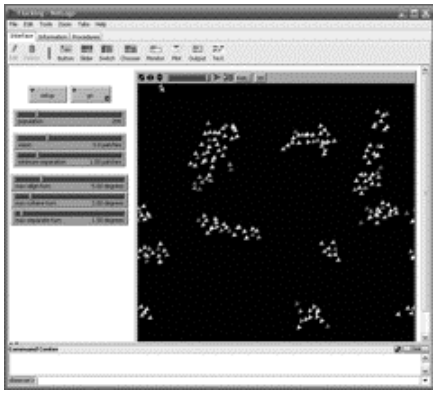


Figure 1: A NetLogo Model of Flocking Behavior.

3. PARTICIPATORY SIMULATION (HUBNET)

In a participatory simulation individuals act out the roles of agents in a complex system [2]. The behavior of the whole system is not defined ahead of time but instead emerges from the participation of various individuals in the simulation. This emergent behavior can then be displayed back to the group as a whole through a central server with the results usually projected on a screen at the front of the room. **HubNet** is the name we have given to the NetLogo architecture designed to enable individuals to participate in a simulation of a complex dynamic system. HubNet enables many users at individual computers or other devices to control the behavior of agents and to view the aggregated results on a joint display. The individual devices can be a mix of different hardware systems (programmable calculators, computers, or handheld devices) that are able to run the Java-based NetLogo platform. The network layer implements flexible communication protocols that include the ability to upload and download data sets, monitor key-presses at the individual level, support real-time interaction as in network computer games, and form collaborative groups of various sizes (e.g., peer to peer, small groups, and whole-group modes). Figure 2 illustrates a HubNet setup just before it was utilized in a classroom.

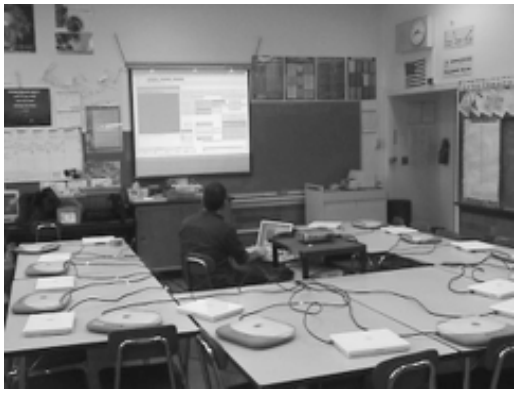


Figure 2: A HubNet Setup About To Be Used.

4. INPUT/OUTPUT TO THE REAL WORLD (NETLOGOLAB)

The **NetLogoLab** module enables users to perform real-time sensing and control, using a standard serial port robotics interface [3]. With various kinds of sensors (temperature, chemical composition, pH, humidity, light, distance, etc.), modelers can feed their models with environmental data, process it, and then control various kinds of actuators (motors, light bulbs, relays, LEDs). Being completely integrated into NetLogo, this module can make use of all the features of the modeling environment. Applications range from model calibration and validation to building robots, interactive simulations and autonomous vehicles. For this demonstration, we will connect a robot to the NetLogo model, and then let it be controlled in three scenarios: a self-contained pre-programmed agent-based model, a sensor-enable apparatus activated by environmental variables (such as human presence, light or temperature) and by humans interacting in a networked simulation, using HubNet. There will also be some basic construction and electronics materials for participants to modify the physical agents and try out different ideas for sensing and control. Figure 3 illustrates a typical NetLogoLab experiment.

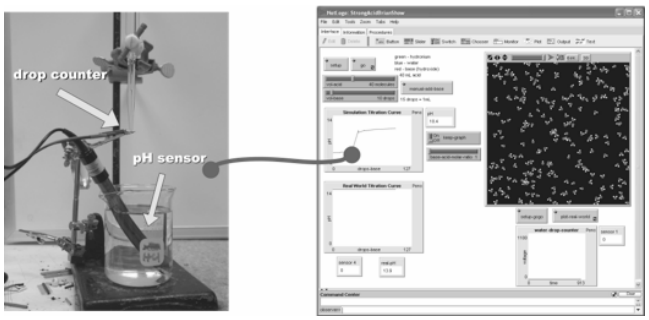


Figure 3: A Typical NetLogoLab Experiment.

5. INTEGRATION AND SUMMARY

The **NetLogo** platform provides the backbone for the integration and the place where the virtual agents are located. **HubNet** then provides a distributed interface into that platform for users to participate with the simulation. **NetLogoLab** then provides an output from the NetLogo platform to control robotic agents, as well as sensor input. In our demonstration we will begin by showing off NetLogo and a few of the hundreds of models that are built in to the NetLogo models library. From there we will progress to demonstrating both the HubNet and the NetLogoLab technologies independently. Finally, we will combine all of these aspects in a participatory simulation where people will work with virtual agents to control a robotic device.

To recapitulate, we plan to demonstrate all three aspects of the NetLogo modeling platform separately, then combine them in a simple proof-of-concept simulation to demonstrate the power behind a platform that seamlessly integrates virtual, participatory, and robotic agents in one package. We feel that this demonstration will excite and entertain a wide variety of researchers who are looking for an easy to learn, powerful multi-agent platform to utilize for experimentation within one of these three fields or any combination.

6. REFERENCES

[1] Epstein, J. and R. Axtell, *Growing Artificial Societies: Social Science from the Bottom Up*. 1996, Washington: Brookings Institution Press.

[2] Wilensky, U., & Stroup, W. *Participatory Simulations: Envisioning the networked classroom as a way to support systems learning for all*. in *Presented at the Annual meeting of the American Research Education Association*. 2002. New Orleans, LA, April 2002.

[3] Blikstein, P. and U. Wilensky. *'Hybrid Modeling': Advanced Scientific Investigation Linking Computer Models and Real-World Sensing.* in *International Conference for the Learning Sciences*. 2006. Bloomington, IN.

[4] Wilensky, U., *NetLogo [Computer software]*. 1999, Center for Connected Learning and Computer-Based Modeling: Evanston, IL.

CONOISE-G: Agent-Based Virtual Organisations

Jigar Patel, W.T. Luke Teacy, Nicholas R. Jennings, Michael Luck
School of Electronics and Computer Science, University of Southampton, UK
{jp03r,wtlt03r,nrj,mml}@ecs.soton.ac.uk

Stuart Chalmers, Nir Oren, Timothy J. Norman, Alun Preece, Peter M. D. Gray
Department of Computing Science, University of Aberdeen, UK
{schalmer,noren,tnorman,apreece}@csd.abdn.ac.uk

Gareth Shercliff, Patrick J. Stockreisser, Jianhua Shao, W. Alex Gray, Nick J. Fiddian
School of Computer Science, Cardiff University, UK
{p.j.stockreisser,g.shercliff,j.shao,w.a.gray,n.j.fiddian}@cs.cardiff.ac.uk

Simon Thompson, BT Group, Adastral Park, United Kingdom
simon.2.thompson@bt.com

ABSTRACT

The ability to create reliable, scalable virtual organisations (VOs) on demand in a dynamic, open and competitive environment is one of the challenges that underlie Grid computing. In response, in the CONOISE-G project, we are developing an infrastructure to support robust and resilient virtual organisation formation and operation. Specifically, CONOISE-G provides mechanisms to assure effective operation of agent-based VOs in the face of disruptive and potentially malicious entities in dynamic, open and competitive environments. In this paper, we describe the CONOISE-G system, outline its use in VO formation and perturbation, and review current work on dealing with unreliable information sources.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*multiagent systems*

General Terms

Design

Keywords

Virtual organisations, trust, Grid, quality of service, policing

1. INTRODUCTION

The engineering of systems using approaches that establish a fixed organisational structure is not sufficient to handle many of the issues inherent in open multi-agent systems (in particular, heterogeneity of agents, trust and accountability, failure handling and recovery, and societal change [1, 2]). Such issues are becoming increasingly important in the context of Grid computing, which aims to enable resource sharing and coordinated problem-solving in dynamic, multi-institutional virtual organisations (VOs) [1]. VOs provide a way of abstracting the complexity of open systems to make them amenable to application development. The organisational structure, participant responsibilities, synchronisation concerns and economic mechanics of the VO are hidden from the VO user.

While the notion of VOs underpins the vision of Grid computing, the conditions under which a new VO should be formed, and the procedures for its formation, operation and dissolution, are still not well-defined. This automated formation and ongoing management of VOs in open environments thus constitutes a major research challenge, a key objective of which is to ensure that they are both agile (can adapt to changing circumstances) and resilient (can achieve their aims in a dynamic and uncertain environment). In addition to traditional constraints that relate to issues such as resource management and bidding strategies, we must also consider softer constraints relating to contract management, trust between VO participants and policing of contracts.

The CONOISE-G project (Grid-enabled Constraint-Oriented Negotiation in an Open Information Services Environment, website at *http://www.conoise.org*) is directed at addressing just these issues. It seeks to support robust and resilient VO formation and operation, and aims to provide mechanisms to assure effective operation of agent-based VOs in the face of disruptive and potentially malicious entities in dynamic, open and competitive environments.

In particular, to operate an effective VO in open, dynamic and competitive environments, it is essential that we also consider how to encourage good interactions, and cope effectively with bad ones. In our view, this requires that QoS levels are monitored, that uncertainty in participant behaviour, possibly arising from self-interest and strategic lying and collusion, is minimised, and that mechanisms for recognising and addressing contract violations once they have occured are established. Addressing these concerns is integral to the wide-scale acceptance of the Grid and agent-based VOs.

2. THE CONOISE-G ARCHITECTURE

In essence, the CONOISE-G architecture comprises several different agents, including *system agents* and *service providers* (SPs), as in Figure 1. The former are needed to achieve core system functionality for VO formation and operation, while the latter are involved in the VO itself.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

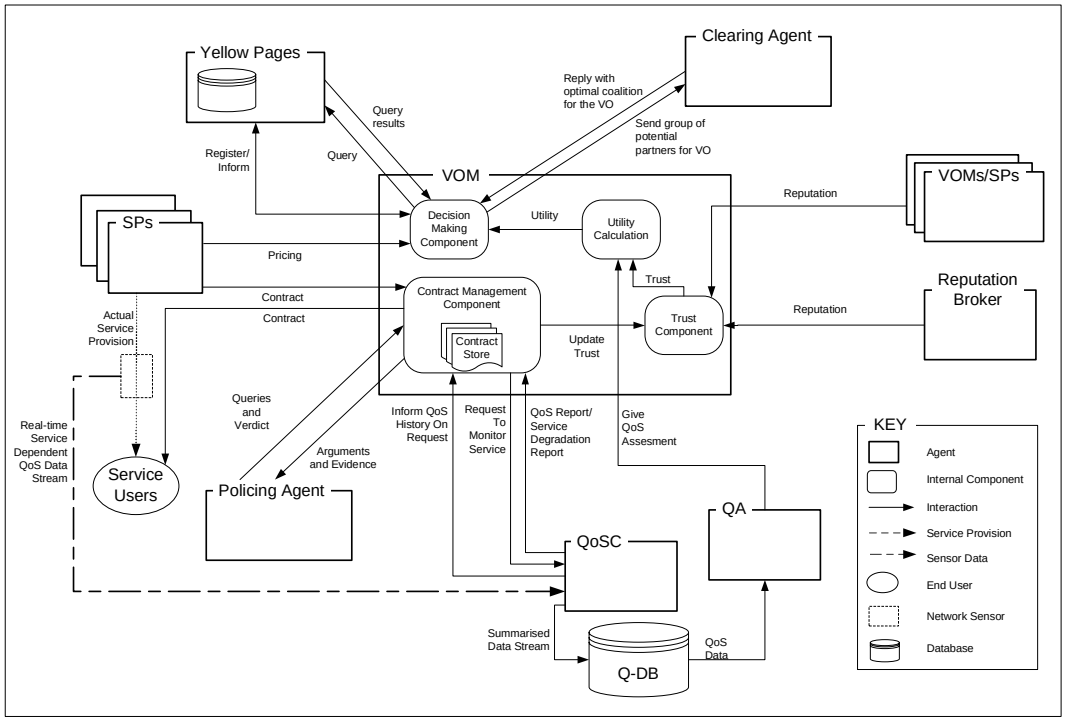


Figure 1: The CONOISE-G architecture

Assuming that service providers have already advertised their services to a Yellow Pages agent (YP), the VO formation process starts with a particular SP acting on behalf of a user, the Requester Agent (RA), which analyses the requester's service requirements, locates the relevant providers through the YP, and then invites the identified providers to bid for the requested services. The quality and trustworthiness of the received bids are assessed by the Quality Agent (QA) and the trust component respectively, and the outcome is combined with the price structure by a Clearing Agent (CA) [3] to determine which combination of the services/providers will form an optimal VO (in terms of price, quality and trust) for the requester. At this point, the VO is formed and the RA takes on the role of VO Manager (VOM), responsible for ensuring that each member of the VO provides its service according to contract.

During the operational phase of the VO, the VOM may request the QoS Consultant (QoSC) to monitor any services provided by any members of the VO, and any member of the VO may invoke the Policing agent to investigate any potential dispute regarding service provision. Ultimately, our aim is for monitoring to take place to inform the user when the actual service level diverges from the agreed service level. At present, however, this is achieved by configuring the levels of QoS for each service that will cause the QoSC to alert the VOM, using predetermined service provision and quality level simulations. When the QoS provision of a service in the VO falls below an acceptable level of service, or some breach of contract is observed, the QoSC alerts the VOM, which initiates a VO re-formation process; relevant information is fed into the trust component to ensure that the provider concerned is penalised to an appropriate level by updating its record of trust.

In this re-formation process, the VOM issues another message to the YP requesting a list of SPs that can provide the service previously provided by the failed SP. As before, the YP identifies possible SPs, bids are received and evaluated, resulting in the CA determining the best SP to replace the failed provider. At this point, the VOM re-forms the VO with the new SP replacing the old one, and instructs the QoSC to stop monitoring the old SP and to monitor the new one instead.

3. CONCLUSIONS

The work described in this paper takes an approach in which issues relating to the formation and operation of robust VOs in the dynamic environments with unreliable agents are considered. In contrast to the "brawn" of the Grid, we have concentrated on the "brains" [1] — on the development of techniques for autonomous problem-solving in VOs. Thus, we have described an agent architecture for re-forming VOs in the face of unreliable information, through the use of a range of techniques that support robust and resilient VO formation and operation for application to realistic electronic commerce scenarios.

Acknowledgements

CONOISE-G is funded by the DTI and EPSRC through the Welsh e-Science Centre, in collaboration with the Office of the Chief Technologist of BT.

4. REFERENCES

[1] I. Foster, N. R. Jennings, and C. Kesselman. Brain meets brawn: Why Grid and agents need each other. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 8–15, 2004.

[2] M. Luck, P. McBurney, and C. Preist. A manifesto for agent technology: Towards next generation computing. *Autonomous Agents and Multi-Agent Systems*, 9(3), 2004.

[3] T. J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian. Agent-based formation of virtual organisations. *Knowledge-Based Systems*, 17:103–111, 2004.

Automated Argument System based on Logic of Multiple-Valued Argumentation

Toshiko Wakaki
Takanori Mukai and
Hiromitsu Miyashita
Shibaura Institute of Technology, Saitama, Japan
{twakaki, p02141, p02139}@sic.shibaura-it.ac.jp

Hajime Sawamura
Kumiko Matsunaga and
Taro Fukumoto
Niigata University, Niigata, Japan
{sawamura, kumiko, fukumoto}@cs.ie.niigata-u.ac.jp

Katsumi Nitta
Tokyo Institute of Technology, Tokyo, Japan
nitta@dis.titech.ac.jp

ABSTRACT

We demonstrate the basic features and advantages of Automated Argument System based on Logic of Multiple-Valued Argumentation by applying it to three convincing arguments: (i) Argument-based recommender system, (ii) Schedule management system and (iii) An integrated system of semantic web reasoning and argument-based reasoning.

Categories and Subject Descriptors

I.2.11 [**Distributed AI**]: Intelligent agents

General Terms

Design

Keywords

Multiple-valued Argumentation, multi-agent system, learning, knowledge acquisition

1. INTRODUCTION

Arguing is a powerful tool we use individually and socially in the daily life. This is also true in agents' world. Computational argumentation has been accepted as a social computing mechanism or paradigm in the multi-agent systems community.

Argumentation is by nature a logic of reasoning under uncertain or changing information environment. Logic of Multiple-valued Argumentation (LMA) was built on top of Extended Annotated Logic Programming EALP [7]. It is a very generic logic of multiple-valued argumentation in the sense that it allows to specify various types of truth values depending on application domains and to deal with uncertain arguments under the speci ed truth values.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

We are going to demonstrate our system with three convincing arguments: (i) Argument-based recommender system, (ii) Schedule management system and (iii) An integrated system of semantic web reasoning and Argument-based reasoning. In doing so, we connect three sites on the Internet: Hakodate conference site, Niigata University in Niigata and Shibaura Institute of Technology in Saitama, where arguing agents and a mediator agent reside in.

2. OUTLINE OF EALP AND LMA

EALP (Extended Annotated Logic Programming) is an expressive logic programming language we formalized for argumentation. The basic language constituents are literals associated with annotations as truth-values or epistemic states of agents. LMA is A Logic of Multiple-valued Argumentation constructed on top of EALP. In this logic, three notions of negation: epistemic, ontological and default negation play a crucial role in argumentation since they turn to yield a momentum or driving force for argumentation. We have soundness and completeness for LMA under the x-point semantics based on the acceptability of arguments and the dialectical proof theory [7]. LMA was implemented in C++/Java and Prolog, where each agent has its own knowledge base in EALP, and a complete lattice of truth values and argument engine are shared among agents concerned (see Figure 1).

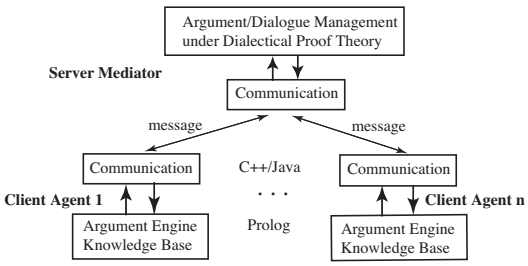


Figure 1: Argument system architecture

3. ARGUMENTS IN DEMONSTRATION

3.1 Argument-based recommender system

Chesñevar and Maguitman propose ArgNet, a recommender system that integrates a traditional web search engine with a defeasible argumentation framework [2]. We demonstrate an improved version of ArgNet by using our LMA in place of their two-valued argumentation framework, so that LMA with its truth-values specialized to $\mathcal{FOUR} = (\{\bot, \mathbf{t}, \mathbf{f}, \top\},$) allows for uncertain argumentation.

3.2 Schedule management system

Schedule management problems are typical targets for which agent systems have been developing their capabilities such as interaction, negotiation, cooperation and so on. We demonstrate a new approach to realizing a schedule management system as LMA-based argumentation system. Then participating agents use not only calendar information but also preferential knowledge base of their own in EALP. The truth values used is a complete lattice of the unit interval of reals $\Re[0, 1]$.

3.3 An integrated system of semantic web reasoning and argument-based reasoning

We will demonstrate an integrated system of semantic web reasoning and argument-based reasoning.

Semantic Web has started to attract much attention as the next-generation web technology. In 2004, the W3C recommended the OWL Web Ontology Language as standard ontology languages for the Semantic Web, which has three increasing expressive sublanguages, namely OWL lite, OWL DL, and OWL full. On the other hand, as shown by Horrocks and Patel-Schneider [6], ontology entailment in OWL Lite and OWL DL reduces to knowledge base (un) satis - ability in the description logics such as the DL $\mathcal{SHIF}$(D) and the DL $\mathcal{SHOIN}$(D) respectively. Though tableaux theorem proving techniques have been commonly used with respect to reasoning ontologies expressed by description logics (or DL, for short), quite recently, many logic programming approaches have been proposed such as Heymans et al.'s method [5] , Baral's method [1] and so on which make use of *answer set programming* (or ASP, for short) [4].

In such progress of the Semantic Web technology, we have implemented a prototype system for the Semantic Web reasoning which can handle ontologies expressed by the DL $\mathcal{SHOIN}$(D) or OWL DL, and can be accessible to and from LMA-based argument system (specialized to two values $\{f, t\}$) on the Web (see Figure 2). It consists of three components, that is, ontology translations done in two steps and ontology reasoning via ASP as follows:

1. Translation of ontology expressed by OWL DL into the one expressed by the DL $\mathcal{SHOIN}$(D) (or vice versa) though the current prototype does not support XML schema primitive datatypes as well as datatype roles.
2. Translation of ontology expressed by DL into *Disjunctive Logic Programs* (or DLPs, for short) based on Heymans et al.'s method or Baral's method,
3. Query evaluation w.r.t. knowledge base (i.e. ontologies) expressed by a DLP using the ASP solver *dlv* [3].

The translation of the rst component is established by using XSLT. With respect to the second component, let $KB = Tbox \cup Abox$ be a knowledge base which is a nite set of axioms expressed by $\mathcal{SHOIN}$(D). Then an axiom F is a *logical consequence*of KB, denoted $KB \models_{DL} F$, i every model of KB satis es F.

Heymans et al. considers the DL $\mathcal{ALCHODQ}(\sqcup, \sqcap)$ which di ers from the DL $\mathcal{SHOIN}$(D) by its lack of transitive roles, inverse roles, and data types. In their method, $\mathcal{ALCHODQ}(\sqcup, \sqcap)$ knowledge base are expressed by *conceptual logic programs* (or CLPs, for short) whose syntax is the restricted DLP. CLPs extend Answer Set Programming with, possibly in nite, open domains, which leads, in general, to undecidable reasoning. Thanks to restricting the syntax of DLPs, reasoning with the resulting CLPs can be reduced to nite, normal Answer Set Programming, for which e cient ASP solvers (e.g. *dlv*)[3] are available.

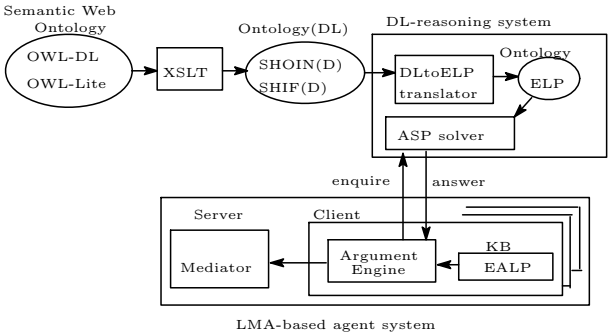


Figure 2: WebArg system architecture

In the demonstration, we show how the semantic web reasoning brie y described above interacts with LMA-based argumentation system. It then attempts to construct arguments with both the usual knowledge base and the ontology knowledge provided by the semantic web reasoner.

4. REFERENCES

[1] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

[2] C. I. Chesñevar and A. G. Maguitman. `ArgNet`: An argument-based recommender system for solving web search queries. In *Procs. of the 2nd. International IEEE Conference on Intelligent Systems (IS 2004)*. IEEE Press, 2004.

[3] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. A deductive system for nonmonotonic reasoning. In *Proc. of LPNMR 1997*, volume 1265, pages 364–375. Springer-Verlag, http://www.dbai.tuwien.ac.at/proj/dlv/, 1997.

[4] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing 9*, pages 365–385, 1991.

[5] S. Heymans, D. V. Nieuwenborgh, and D. Vermeir. Nonmonotonic ontological and rule-based reasoning with extended conceptual logic programs. In *Proc. of ESWC'05*, volume 3532, pages 392–407. Springer-Verlag, 2005.

[6] I. Horrocs and P. F. Patel-Schneider. Reducing owl entailment to description logic satis ability. In *Proc. ISWC'2003*, volume LNCS 2870, pages 17–29. Springer-Verlag, 2003.

[7] T. Takahashi and H. Sawamura. A logic of multiple-valued argumentation. In *Proceedings of the third international joint conference on Autonomous Agents and Multi Agent Systems (AAMAS'2004)*, pages 800–807. ACM, 2004.

Computational Mechanism Design for Multi-Sensor Information Fusion

A. Rogers, R.K. Dash and N.R. Jennings
School of Electronics and Computer Science
University of Southampton
Southampton, UK.
{acr,rkd02r,nrj}@ecs.soton.ac.uk

S. Reece and S. Roberts
Department of Engineering Science
University of Oxford
Oxford, UK.
{reece,sjrob}@robots.ox.ac.uk

ABSTRACT

Conventional information fusion architectures are challenged by developments in sensor networks that allow individually-owned (and thereby selfish) sensors to interact and share data. Given this, we advocate the use of tools and techniques from computational mechanism design, a field at the intersection of computer science, game theory and economics, to address the challenges posed by these networks. These techniques allow us to engineer networks with desirable system-wide properties, in which sensors are represented as selfish rational agents, each attempting to fulfill their own individuals goals. In this paper, we present the work of the Argus II DARP project (`http://www.robots.ox.ac.uk/~argus/`) and we demonstrate the application of mechanism design within a real world information fusion sensor network scenario.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Design, Theory

Keywords

computational mechanism design, sensor networks

1. INTRODUCTION

Multi-sensor networks (MSN) are being deployed in a wide variety of application areas ranging from military sensing to environmental monitoring and traffic control. Such networks consist of a number of sensors connected via a communication network. Each sensor is able to sense their local environment, but can also make use of data transmitted from neighbouring sensors in order to improve the accuracy of their own measurements (e.g. by combining their own noisy observation, with the observations from a number of other sensors, in order to reduce the final uncertainty).

To date, research in sensor networks has mainly concentrated on using cooperation among distributed sensors [3]. At its core, this approach involves determining the exchanges of observed data between the sensors that results in the maximum gain in information across the whole sensor network. However, this approach overlooks the fact that in some applications each sensor may be individually-owned by different stakeholders. In such cases, the sensors are operating in a competitive rather than cooperative environment, and thus, they may attempt to optimise their own gain from the network, at a cost to the performance of the entire system.

To address the challenges of building sensor networks which exhibit desirable system-wide properties, despite the competitive behaviour of the constituent sensors, the Argus II DARP project (a collaboration between two academic partners: the University of Southampton and the University of Oxford, and three industrial partners: BAE SYSTEMS, QinetiQ and Rolls-Royce — see `http://www.robots.ox.ac.uk/~argus/` for more details) is exploring the application of computational mechanism design within information fusion scenarios.

2. INFORMATION FUSION SCENARIO

In this work, we consider a real world aerial surveillance scenario such as that posed in disaster relief contexts, where multiple emergency response agencies, with aerial vehicles of different capabilities, must interact in order to locate casualties. The sensor network is formed from sensors that are mounted onboard these aerial vehicles. Each sensor is provided with an imprecise estimate of its own location by the navigation system of the aerial vehicles to which it is mounted, and it is tasked with detecting and tracking multiple targets within a *region of observation* immediately surrounding itself. Within its region of observation, the sensor is able to estimate the position of each target by making noisy or imprecise measurements of the range and bearing of the target from itself. However, in order to better resolve the uncertainty in these position estimates, the sensors must acquire target observations from neighbouring sensors and then fuse these observations with their own.

The sensors are connected together via a communication network that has a limited bandwidth, and thus, there must be some coordination to determine which data should be sent to which sensors in order to make best use of this limited global resource. However, since each sensor is owned by a different stakeholder, it is thus selfishly seeking to maximise the accuracy of its own target position estimates. As a result, each sensor has a positive disinclination to share its observations since in doing so, it will occupy valuable bandwidth which may be used to receive observations from other agents. Thus, we use techniques from mechanism design to engineer a protocol that incentivises the sharing of observations, whilst also ensuring that the global resource of communication bandwidth is used effectively.

However, in order to apply the tools of mechanism design we

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

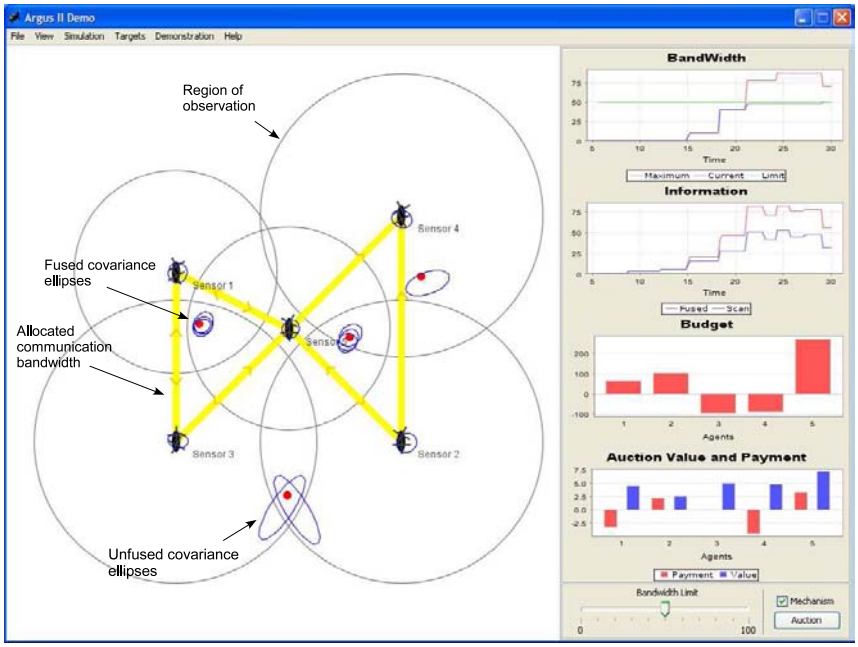


Figure 1: Example sensor network system showing the auction allocation in process and the resulting communication allocation.

require a metric that allows us to assign a value to the observations provided by each sensor. In our work, we have derived a metric based on the Fisher information of these observations [4]. This measure is related to the precision of the observations, and thus sensors that make range and bearing measurements with greater precision, generate observations with greater information content, and hence greater value. This metric is particularly attractive as when independent observations are fused, the information content of the fused observation, is simply equal to the sum of the two individual observations.

Adopting this measure allows us to develop a protocol, whereby an independent auctioneer allocates the communication bandwidth based on the valuations of observations supplied by the individual sensors. Thus far, we have focused our research onto a class of direct mechanisms that are said to be strategy-proof or incentive compatible [1, 2]. That is, the sensors have a dominant strategy to truthfully reveal the information content of their observations to the auctioneer (they thus have no incentive to engage in complex strategic behaviour). Such an outcome is achieved through the payments that the sensors must make (or collect) for receiving (or transmitting) observations. These payments are calculated by the auctioneer in order to align the individual goals of the sensors with the overall system-wide goals of the system designer. By incentivising truthful reporting of the information content of observations from the sensors, the auctioneer is then able to ensure that the bandwidth is allocated efficiently (i.e. to maximise the information gain of the entire network).

3. DEMONSTRATOR

To demonstrate the application of this mechanism, the information fusion scenario, the auction mechanism, and the individual sensors have been implemented within a fully asynchronous Java simulation environment. Figure 1 shows a screen shot of this simulator in operation. The main view shows the measurements and communication of the individual sensors that make up the sensor network. The graphs to the right record metrics of the entire system such as the total information gain of the network, the utilisation of the available bandwidth, and the outputs of the auction process.

4. CONCLUSIONS

In this work, we have demonstrated an effective and efficient control mechanism for sensor networks, in which the constituent sensors are owned by different stakeholders that have their own goals and objectives. Our system has a principled valuation metric based on the Fisher information of a sensor's observations and uses computational mechanism design to engineer a sensor network with a particular desirable system-wide property (in this case, the efficient use of limited bandwidth capacity), despite the selfish goals and actions of the individual sensors. Whilst this research is ongoing, and is currently focusing on how the role of the auctioneer may be distributed amongst the sensors within the network, it is clear that computational mechanism design offers an invaluable and powerful set of tools and techniques to address the challenges posed by these scenarios.

5. ACKNOWLEDGMENTS

The work reported on here undertaken as part of the ARGUS II DARP (Defence and Aerospace Research Partnership). This is a collaborative project involving BAE SYSTEMS, QinetiQ, Rolls-Royce, the University of Oxford and the University of Southampton, and is funded by the industrial partners together with the EPSRC, MoD and Dti.

6. REFERENCES

[1] R. Dash, A. Rogers, S. Reece, S. Roberts, and N. Jennings. Constrained bandwidth allocation in multi-sensor information fusion: A mechanism design approach. In *Proc. of the 8th Int. Conf. on Information Fusion*, Philadelphia, USA, 2005.

[2] R. K. Dash, D. C. Parkes, and N. R. Jennings. Computational-mechanism design: A call to arms. *IEEE Intelligent Systems*, 18(6):40–47, 2003.

[3] V. Lesser, C. Ortiz, and M. Tambe, editors. *Distributed sensor networks: A multiagent perspective*. Kluwer Publishing, 2003.

[4] J. Manyika and H. Durrant-Whyte. *Data fusion and sensor management: A decentralized information-theoretic approach*. Ellis Horwood, 1994.

Component Agent Framework for domain-Experts (CAFnE) Toolkit

Gaya Jayatilleke John Thangarajah Lin Padgham Michael Winikoff

RMIT University
Melbourne, AUSTRALIA
{gjayatil,johthan,linpa,winikoff}@cs.rmit.edu.au

ABSTRACT

The Component Agent Framework for domain-Experts (CAFnE) toolkit is an extension to the Prometheus Design Tool (PDT). It uses the detailed design produced by PDT with further annotations by domain experts to automatically generate executable code into a desired agent platform. The key feature of CAFnE is that it allows domain experts with limited programming skills to easily build and modify agent systems.

Categories and Subject Descriptors

D.2.6 [**Software Engineering**]: Programming Environments

General Terms

Design, Human Factors

Keywords

Agent Software Engineering, Methodologies, Tool Support

1. INTRODUCTION

The Prometheus Design Tool (PDT) [4] supports the design and development of multi-agent systems using the Prometheus methodology [5]. While the methodology supports all phases of development PDT in its current state only supports the system specification and design phases.

The Component Agent Framework for domain-Experts (CAFnE) [1] toolkit was built to allow domain experts with limited programming skills to easily build and modify agent systems. This is a result of the experience gained in working with a group of meteorologists on an agent-based weather alerting system [3]. It is often the case that domain experts are the ones who identify modifications in complex domain applications. However, they have less support for making these changes.

The CAFnE extension to PDT supports the design and implementation of BDI style agent applications. It uses a set of components that define BDI style agents such as event, goal, plan and belief. By adopting a strict structure for these components and visual modelling, CAFnE is able to provide an environment for domain experts to maintain agent systems. The toolkit is based on the concepts of model-driven development and component-based software engineering[2]. Figure 1-(a) shows the layered definition approach used in CAFnE. The toolkit works at the M1 layer and automatically generates the runtime code at the M0 layer.

In addition to using the design artifacts in PDT, CAFnE allows domain experts to use *Attributes* to define the environment and the internal beliefs of each agent of the system; and uses execution units called *Steps* within plans to specify plan behaviour.

CAFnE takes these components from the design and first creates a domain dependent but platform independent component model of the application. This model is then transformed by a *transformation module* into a specific agent platform that can be compiled and executed. Currently we have developed a transformation module for the JACK[1] agent platform. Similar modules for other platforms can be integrated into CAFnE. Figure 1-(b) provides an overview of the various modules in CAFnE.

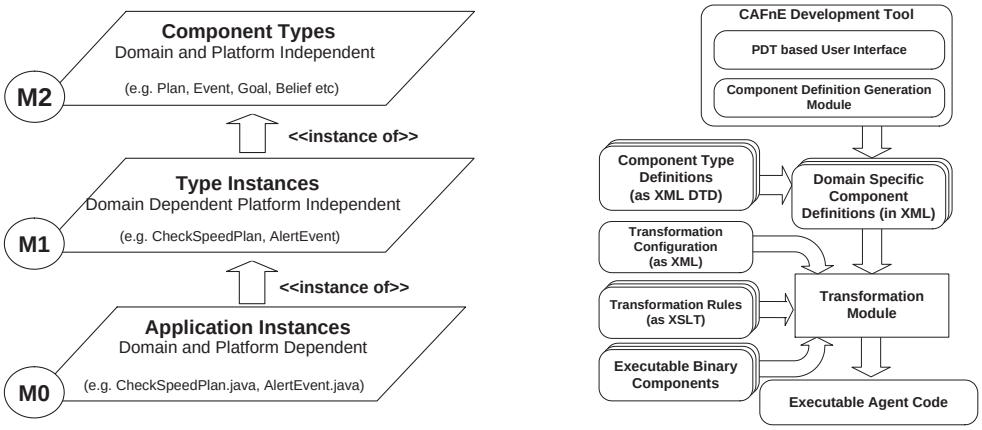
2. FEATURES OF THE TOOL

Integrated Development Environment: CAFnE provides a GUI based integrated development environment (IDE) for visual modelling, code generation, compilation and execution of agent applications. In addition to standard features such as design diagrams, CAFnE provides various features that help a domain expert in maintaining an application. The IDE includes an agent instance diagram where initial agent instances can be defined. Users can go into specific details of an agent instance as providing initial belief data values and start up plans to run when the agent instance is created at runtime.

Domain Expert Orientation: CAFnE incorporates various measures in order to allow domain experts with limited agent programming knowledge to build and modify agent applications. CAFnE uses a simplified BDI agent model that is easy to comprehend. By providing diagrammatic representations of the agent application at different levels, CAFnE helps the user in understanding the system. With its automated code generation, CAFnE has reduced programming tasks to editing diagrams and providing necessary parameter values for components. The Attribute based definition of the environment and the internal beliefs, allows the tool to prompt the user with possible parameter or attribute name lists at input boxes. This greatly helps the users in not having to remember parameter names and eliminating errors in input values. The toolkit also has tracing features to help track execution at runtime (see below).

[1]http://www.agent-software.com

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.



(a) Layered Components Definitions

(b) Overview of the Toolkit Modules

Figure 1: An Overview of CAFnE

Agent Platform Independent: In line with Model Driven Development principles [2], CAFnE uses a platform-independent model (see Figure 1-a). Platform specific code is generated using transformation rules (Figure 1-b). One consequence of this approach is that what the model which the user works with is not specific to a given platform. Changing CAFnE to target an alternative platform is simply a matter of providing alternative transformation rules. This is easiest if the alternative platform is also BDI-like.

Error Checking: CAFnE includes a consistency checker that validates the application model against a set of rules that define the components and relationships between them. In other words, these rules check whether the model created by the user is internally consistent with the meta model used by CAFnE. The errors detected range from warnings such as an agent not having incoming data, to more serious errors such as missing parameters in steps.

Tracing: Basic "Tracing" options provided in the tool help track the execution of the agent system. With event-plan based execution of BDI agents having a lot of asynchronous processes, tracing such events as plan firing, goal raising and belief updating, helps identify errors in the application.

Support for Component Reuse: CAFnE allows for easy reuse of components such as plans, events and beliefs used in the application. Users can copy and paste components across agents and within agents, with the toolkit checking for any inconsistencies in these actions. To support this process the toolkit has features such as automated linking of components when copies are made. Step components used in Plans go beyond the boundary of a single application and allow reuse across applications. It is easy to define additional Steps, and a Step defined in one application can be reused in another application as long as it is provided with the correct input and output parameters.

3. DISCUSSION

We recently conducted an evaluation of CAFnE using five senior forecasters from the Bureau of Meteorology in Melbourne, with varying levels of programming experience, none of whom had developed an agent system before. They were able to rapidly become familiar with the CAFnE concepts and make changes to an existing design for a weather alert system, such as adding the ability to alert on wind data, as well as temperature.

A number of other agent-oriented software engineering methodologies have tool support including Tropos[2] and MaSE[3]. Tropos tool support consists of a number of separate tools that cover different aspects of the software engineering process. None of these tools provide support for transforming design into executable agent code. The JACK Design Environment (JDE) allows design diagrams to be specified and supports automatic code generation and execution. The JDE however is tailored specifically for JACK and requires the user to understand JACK-specific concepts, as well as the JACK programming language.

Currently CAFnE is a stand-alone application developed with the same graphical model as PDT and takes a PDT design file as input. There are still some features that we would like to add to CAFnE, for example, backward propagation (that is, changes in the CAFnE descriptor propagated to the design diagrams) and visualization tools for users to understand the dynamics of a multi-agent system.

This work was done in collaboration with the Australian Bureau of Meteorology and Agent Oriented Software Pty. Ltd. and was also supported by the Australian Research Council (Linkage Grants LP0347025 and LP0453486).

4. REFERENCES

[1] G. Jayatilleke, L. Padgham, and M. Winikoff. Component agent framework for non-experts (CAFnE) toolkit. In R. Unland, M. Calisti, and M. Klusch, editors, *Software Agent-Based Applications and Prototypes*, pages 169–196. Birkhaeuser Publishing Company, Sept. 2005.

[2] A. Kleppe, J. Warmer, and W. Bast. *MDA Explained, The Model Driven Architecture: Practice and Promise*. Addison-Wesley Publishing Company, 2003.

[3] I. Mathieson, S. Dance, L. Padgham, M. Gorman, and M. Winikoff. An open meteorological alerting system: Issues and solutions. In *Proceedings of the 27th Australasian Computer Science Conference (to appear)*, Dunedin, New Zealand, Jan. 2004.

[4] L. Padgham, J. Thangarajah, and M. Winikoff. Tool support for agent development using the Prometheus methodology. In *First international workshop on Integration of Software Engineering and Agent Technology (ISEAT 2005)*, Melbourne, Australia, September 2005.

[5] L. Padgham and M. Winikoff. *Developing Intelligent Agent Systems: A practical guide*. Wiley Series in Agent Technology. John Wiley and Sons, 2004.

[2]http://www.troposproject.org

[3]http://macr.cis.ksu.edu/projects/mase.htm

Conflict Negotiation Among Personal Calendar Agents

Pauline M. Berry*, Cory Albright*, Emma Bowring**, Ken Conley*, Kenneth Nitz*, Jonathan P. Pearce**, Bart Peintner*, Shahin Saadati*, Milind Tambe**, Tomás Uribe*, and Neil Yorke-Smith*

*SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025, USA

{berry, conley, peintner}@ai.sri.com,
{saadati, uribe, nysmith}@ai.sri.com,
{kenneth.nitz, cory.albright} @sri.com

**University of Southern California
3737 Watt Way,
Los Angeles, CA 90089, USA

{bowring, jppearce, tambe}@usc.edu

ABSTRACT

We will demonstrate distributed conflict resolution in the context of personalized meeting scheduling. The demonstration will show how distributed constraint optimization can be used to facilitate interaction between cognitive agents and their users. The system is part of the CALO personal cognitive assistant that will also be explored during the demonstration.

Categories and Subject Descriptors

H.4.1 (**Information Systems Applications**): Office Automation – Time management; I.2.11 (**Artificial Intelligence**): Distributed Artificial Intelligence – Intelligent agents; K.6.3 (**Management of Computing and Information Systems**): Software Management.

General Terms

Design, Human Factors

Keywords

Scheduling, Personal Assistants, User Interfaces

DESCRIPTION OF DEMONSTRATION

The Cognitive Assistant that Learns and Organizes—CALO—is a personal assistive agent designed for a busy knowledge worker. The name was inspired by the Latin word *calonis*, "soldier's servant," because the goal of the project sponsor, the U.S. Defense Advanced Research Projects Agency, is to create a cognitive system that can reason, learn, and respond to surprise in order to assist in military situations.

The CALO project brings together leading computer scientists and researchers in artificial intelligence, perception, machine learning, natural language processing, knowledge representation, multimodal modal dialog, cyber-awareness, human-computer interaction, and flexible planning. The single research focus of all these experts is to create an integrated agent system that can "learn in the wild"—that is, adapt to changes in its environment and its users' goals and tasks without programming assistance or technical intervention.

An integral part of the work of any busy decision maker, whether military or civilian, is the management of her time. In particular, the management of interactions with other people, including colleagues and customers, involves careful meeting scheduling and calendar adaptation. As such, time management and scheduling capabilities are central to the CALO system. CALO exists in an open, unbounded environment, where issues of privacy, authority, adjustable autonomy, cross-organizational scheduling, and uncertain availability of participants abound. PTIME [1] is an autonomous agent integrated within the larger CALO agent [2]. The goal of PTIME is to carry partial responsibility for managing the user's calendar and the scheduling of meetings, thus decreasing the user's need for complex interactions with other humans to plan when to meet or how to reorganize their calendars. PTIME absorbs much of the responsibility for handling negotiations and decreasing the communication overload while keeping the user informed about the state of the process.

PTIME agents in collaboration with their users are autonomous, self-interested entities that manage their users' own calendars. They are, in effect, single-calendar schedulers dependent on other agents to coordinate shared calendar entities. The functional architecture of a single agent is shown in Figure 1. The scheduling task is viewed as a shared goal of the user and the agent. For a PTIME agent, global utility is thus secondary to maximizing the local utility of its user, which typically includes a component valuing cooperation with others [4]. This is in contrast to most distributed scheduling systems [5], which aim to maximize some notion of global schedule quality (e.g., [3]).

Hence, PTIME supports two forms of collaboration: between itself and its user and between itself and the PTIME agents of other users [1]. PTIME also extends the notion of collaboration from inter-agent negotiation to the collaborative dialog between agents and their human users; the collaborative scheduling process is separated from the scheduling algorithms, to enable interaction with the user and other PTIME agents. This interaction forms the framework for learning and adjustable autonomy. The collaborative interface is shown in Figure 2.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

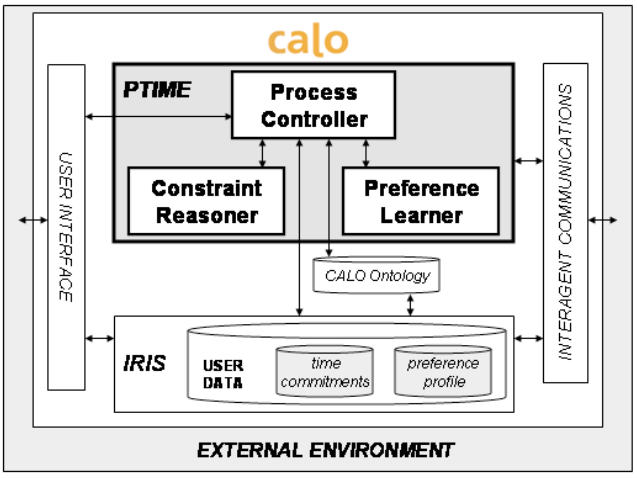


Figure 1: Architecture of a PTIME Agent

User studies confirm that other participants' constraints and preferences often influence the meeting organizer's decisions [4]. When PTIME decides which of many possible schedules to present to the user, it reasons about how each candidate schedule will affect others. This requires PTIME to retrieve from all affected parties their schedules, their preferences, and how the proposed schedule affects them. Once a user selects a schedule from those presented, PTIME must negotiate with the agents of the other parties to resolve conflicts within each schedule.

To determine the effect of a particular schedule on others, PTIME uses a distributed querying mechanism, based on a Distributed Constraint Optimization (DCOP) protocol that obtains a cost reflecting the rescheduling effort required as a result of a proposed schedule. The scope of negotiation within the DCOP algorithm is controlled heuristically. The cost for users that do not respond (i.e., communication with their CALO fails) is estimated using the most recent knowledge about their schedules and their negotiation history.

When the user organizing a new meeting selects a candidate schedule, PTIME invites all other meeting participants, with a justification for the request. All conflicts are then resolved by other users who, in turn, will either resolve the conflicts with other CALOs or choose to not attend.

Our demonstration will take the user through a scheduling task involving the resolution of conflicts with other participants. It will illustrate the PTIME process including the specification of a scheduling problem, inter-agent negotiation, the dialog with the user, and a subset of CALO system functionality.

ACKNOWLEDGMENTS

We thank all the members of the CALO project, especially our collaborators at SRI, University of Michigan, and USC. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010.



Figure 2: PTIME Collaborative Interface

REFERENCES

[1] P. M. Berry, K. Conley, M. Gervasio, B. Peintner, T. Uribe, and N. Yorke-Smith (2006). Deploying a personalized time management agent. In *Proc. of AAMAS-06 Industrial Track.*

[2] W. Mark and R. Perrault (2005). *CALO: Cognitive Assistant the Learns and Organizes*. www.ai.sri.com/project/CALO.

[3] P.J. Modi, W. Shen, M. Tambe, and M. Yokoo (2005). ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2).

[4] L. Palen (1999). Social, individual and technological issues for groupware calendar systems. In *Proc. of CHI-99*.

[5] S. Sandip and E. Durfee (1998). A formal study of distributed meeting scheduling. *J. Group Decision and Negotiation*, 7.

Synthetic Humans in Emergency Response Drills

Daniel Massaguer, Vidhya Balasubramanian, Sharad Mehrotra, and Nalini Venkatasubramanian

Donald Bren School of Information and Computer Science
University of California, Irvine
Irvine, CA 92697, USA
{dmassagu, vbalasub, sharad, nalini}@ics.uci.edu

ABSTRACT

Being able to realistically simulate humans in emergency response activities is a key challenge for advancing the field of information technology for emergency response. Building good human models enables testing information technology solutions in any emergency response simulated scenario. This paper presents an agent behavior model for mimicking peoples's behavior on emergency response activities. The accompanying demo demonstrates a concrete model for evacuations integrated with an emergency drill simulation environment.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence. Distributed Artificial Intelligence[Intelligent agents, Multiagent systems]; H.1.2 [**Information Systems**]: Models and Principles. User/Machine Systems[Human information processing]; I.6.3 [**Computing Methodologies**]: Simulation and Modeling Applications; I.6.4 [**Computing Methodologies**]: Simulation and Modeling. Model Validation and Analysis

General Terms

Design, Algorithms, Experimentation

Keywords

Agents and cognitive models, agent-based simulation and modeling, applications of autonomous agents and multi-agent systems, artificial social systems, perception, action and planning in agents, synthetic, embodied, emotional and believable agents

1. INTRODUCTION

Information Technology (IT) enables better information flow from source to consumers. It changes the quality and amount of information humans have during emergency response and, thus, influences the decisions humans make. In order to evaluate the impact of IT solutions in emergency response, special attention on human models that emphasize the role of information in the decision-making process is needed.

We model the decision-making process as an information processing system [4]. In our approach, we determine the relevant information variables for each decision. The information abstraction process converts agent observations of the scenario (obtained via raw sensory data) into relevant information variables. Our agents use this information to make decisions. The decision-making model is tested and calibrated with an emergency drill simulation environment– DrillSim. The rest of the paper is organized as follows. Section 2 sets the context by introducing DrillSim. Section 3 presents our agent model. Section 4 describes how our model integrates and calibrates with DrillSim. The paper then concludes with the conclusions and future work.

2. DRILLSIM

DrillSim [1] is a multi-agent emergency response simulation system. The system consists of a multi-agent simulator and a pervasive space. The multi-agent simulator models an emergency response activity where each agent represents a person. The pervasive space captures a drill of the activity in the real space. The system allows both a real drill and a simulation of a response activity to be played out simultaneously. This way, the whole activity takes place in an hybrid world–the merging of the real drill and the simulation. DrillSim is a system for testing IT solutions for disaster response. This testing is enabled due to the plug-and-play capabilities of the system that allow plugging IT solutions for disaster response, thereby translating IT metrics (e.g., delay, QoS) to emergency response metrics (e.g., evacuation time, casualties).

3. AGENT BEHAVIOR MODEL

Each agent in DrillSim represents a person in the response activity. Agents are the key constituents of the activity and take decisions based on their awareness and their cognitive characteristics. Agent behavior (Figure 1) in our model is motivated by the information processing model of [4]. Agents take some action every t time units. For this purpose, an agent acquires awareness of the world around it (i.e. event coding), transforms the acquired data into information, and makes decisions based on this information. Then, based on the decisions, it (re)generates a set of action plans. These plans dictate the actions the agent attempts before going to sleep again. For example, hearing a fire

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

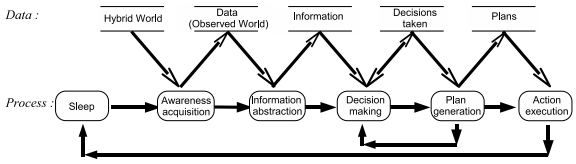


Figure 1: Agent Behavior process.

alarm results in the decision of exiting a floor, which results in a navigation plan to attempt to go from the current location to an exit location on the floor, which results in the agent trying to walk one step following the navigation plan.

Awareness acquisition: Awareness acquisition is the process of coding events into data, thus adding data to the agent's observed world. An agent's observed world is composed of the information it knows a priori (e.g., a map of the building) and the events it observes during its life.

Information abstraction: Based on its observed world, the agent translates the acquired awareness into information. Information is a set of variables that take values between 0 and 1. Note that, information abstraction is a cognitive process and varies between individuals. It depends on factors such as underlaying social networks [5] and the agent's cognitive characteristics.

Decision making: The decision making step is based on a probabilistic model over an artificial neural network [3] (i.e. a time-discrete recurrent artificial neural network). This allows for explicitly modeling the importance of each piece of information on each decision, setting the emphasis on the impact of information on decision making rather than on the reasoning process itself. The characteristics of the neural network parameters (i.e. activation function, weights matrix) are set according to the agent's role and calibrated through running real emergency response drills within the DrillSim environment.

Planning of actions and execution: For every decision an agent decides to take, the agent generates a plan of actions if no previous plan exists or if the previous plan is invalid. Planning may also involve taking further decisions. These decisions are also decided at the decision-making neural network above presented. After plan generation, the agent attempts an action from each plan. The most basic actions are walk and speak. For example, given a navigation plan, the agent walks one step more of the computed path.

4. INTEGRATION WITH DRILLSIM

An initial prototype of DrillSim with the agent behavior here described has been implemented in Java and JADE [2]. This prototype provides a GUI (shown during the demo) that allows real humans to observe and interact with the drill simulation. Figure 2 shows a snapshot of the GUI. In particular, it allows a user to start the simulation, pull the fire alarm, input a hazard, get an arbitrary view of the simulation in 2D or 3D, get a 3D view of what an agent is viewing, send messages to other agents, control an agent, and get statistics of the evacuation.

5. CONCLUSIONS AND FUTURE WORK

Providing realistic agent behavior for simulating emergency response activities will definitely catalyze information technology research in emergency response. Realistic agent behavior enables simulating emergency response activities for evaluating different information technology solutions. We propose to model agent behavior as an information processing system based on five steps: Awareness acquisition, information abstraction, decision making, plan generation, and action execution. This five-step process translates raw events observed by the agent to concrete actions. The agent behavior is integrated and demonstrated with DrillSim. The instrumented environment of DrillSim allows calibrating the decision-making model.

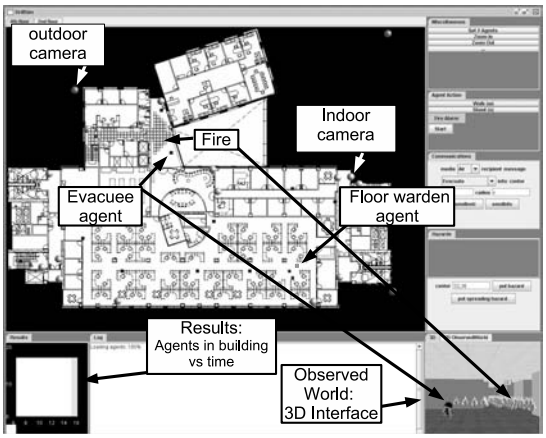


Figure 2: Snapshot of the prototype.

Thoroughly modeling agent behavior is a complex task and calibrating it using real data is indeed challenging. Our future work involves exhaustively calibrating our decision-making model and making it more comprehensive.

6. ACKNOWLEDGMENTS

We would like to thank the rest of the DrillSim team for their dedication to the DrillSim project. This research has been supported by the National Science Foundation under award numbers 0331707 and 0331690.

7. REFERENCES

[1] Drillsim: Multi-agent simulator for crisis response. http://www.ics.uci.edu/ projects/drillsim/, 2005.

[2] F. Bellifemine, A. Poggi, G. Rimassa, and P. Turci". "an object oriented framework to realize agent systems". In *WOA 2000*, May "2000".

[3] S. Haykin. *"Neural Networks - A Comprehensive Foundation"*. Prentice Hall, 1999.

[4] L. Thow-Yick. The basic entity model: a fundamental theoretical model of information and information processing. *Information Processing and Management*, 30(5):647–661, 1994.

[5] S. Wasserman and K. Faust. *Social Network Analysis: Methods and applications*. Cambridge University Press, 1994.

ITSUMO†: an Intelligent Transportation System for Urban Mobility

Bruno Castro da Silva†, Robert Junges†, Denise de Oliveira‡, and Ana L.C. Bazzan†*
Instituto de Informática, UFRGS
Caixa Postal 15064, CEP 91.501-970 Porto Alegre, RS, Brazil
{bcs,junges,edenise,bazzan}@inf.ufrgs.br

ABSTRACT

This paper presents an overview of ITSUMO, a microscopic traffic simulator based on cellular automata. The implementation uses agent technologies with a bottom-up philosophy in mind. We give an overview of the system and some details of its modules (data, simulation, driver and information/visualization).

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence
Multiagent systems, Intelligent agents

General Terms

Human Factors

1. INTRODUCTION AND MOTIVATION

The increasing urban mobility challenges traffic engineers, urban planning experts, and researchers involved with optimization and information technology. Urban mobility both produces information and increases the demand for it. In order to produce the information to be broadcasted, a fast simulation model is necessary. We use the Nagel–Schreckenberg approach [3] which is a microscopic model for traffic simulation based on cellular–automata (CA).

The motivation for the development of the simulation module is threefold. First, although several similar tools exist, these are built normally with the objective to tackle isolated problems. Second, in general, one has no access to the source code regarding both commercial and academic tools. Finally, we see this domain as a very rich area for experimenting some agent-based techniques, especially modelling and simulation of drivers (agents).

Therefore, driven by the incentives given by Brazilian funding agencies regarding free software, we have developed ITSUMO – Intelligent Transportation System for Urban Mobility [4]. One of the aims of the project is to create an information system for urban mobility capable of integrating different functionalities, such as a simple traffic control (mostly based on control of traffic lights), traffic management and real-time information providing via internet and/or mobile phone. The idea is to distribute the system under free-software license. The system has been used to investigate route choice scenarios [1], as well as control (refer to `http://www.inf.ufrgs.br/~mas/traffic/itsumo/demo` for screen shots and other publications).

To address these objetives, it is possible to define drivers as intelligent agents and to plug each driver model as a module in the simulation system. This approach is in contrast with the current models, which are purely reactive and ignore drivers' mental states (informational and motivational data). The kernel of the simulator was programmed using the object-oriented paradigm. The idea is that anyone can aggregate further code when necessary. A XML-based database, used to define the network objects, and a visualization module are also part of the package.

2. MICROSCOPIC SIMULATION MODEL

Basically, there are two approaches to model traffic movement: the macroscopic and the microscopic. The former is mainly concerned with the movement of platoons of vehicles, focusing on the aggregate level. It considers only averaged vehicle densities but not individual traffic participants. In the microscopic model of simulation, each object can be described as detailed as desired, thus permitting a more realistic modeling of drivers' behavior. In the microscopic approach both travel and/or route choices may be considered, which is a key issue in simulating traffic since those choices are becoming increasingly more complex. Also, individual traffic lights can be modelled according to several approaches, from classical off-line coordination to recently proposed ones (negotiation, communication-free, via game theory, reinforcement learning, swarm intelligence, etc). In fact we have been using this model and simulator to investigate different aspects of the problem: route choice, traffic lights coordination and control, etc.

In order to achieve the necessary simplicity and performance, we use the Nagel–Schreckenberg model [3]. In short, each road is divided in cells with a fixed length. This allows the representation of a road as an array where vehicles occupy discrete positions. Each vehicle travels with a speed based on the number of cells it currently may advance without hitting another vehicle. The vehicle behavior is expressed by rules that represent a special form of car–following behavior. This simple, yet valid microscopic traffic model, can be implemented in such an efficient way that is good enough for real-time simulation and control of traffic. Given that every vehicle has a nonnegative integer speed (v, limited to $vmax$), the following four rules are verified simultaneously for all vehicles in a CA way: movement (each vehicle advances v cells at each time step); acceleration (each vehicle's speed is increased by one

*corresponding author
†project and authors supported by CNPq

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8-12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

unit, up to $vmax$ or the gap – the number of empty cells in front of a vehicle); interaction (if the vehicle ahead is too close, v is decreased by one unit; and randomization (vehicles decelerate with probability p in order to simulate the nondeterministic dynamics of vehicle movement). Although these rules might seem too simplistic, investigations reported in the literature showed that the cellular automaton model is capable of reproducing macroscopic traffic flow features including realistic lane changing behavior.

3. DESCRIPTION OF THE SIMULATOR

ITSUMO can use both off-line information (maintained by different providers), and on-line (e.g. traffic flow). The information regarding the network is stored in a XML file. In addition to the CA approach, one can also define other driver decision-making procedures via a special, optional, module. If the user does not define particular classes of drivers, then the simulation is performed using the standard driver model described in Section 2. A visualization module retrieves data originated from the microscopic simulation and exhibits a graphical representation of the traffic simulation. The ITSUMO system is thus composed by four distinct modules: the data module, the simulation kernel, the driver definition module, and the visualization module.

3.1 Data Module

This module creates, updates, and stores (XML file) both the static and the dynamic objects to be used in the simulation, as for instance the cartesian coordinates of the intersections. The main attributes are described below:

- **General Settings**: topology name, traffic system orientation (right-handed, left-handed), cell size, frequency of sensor measurements, deceleration probability, etc.;
- **Network**: network name and its settings;
- **Node**: cartesian coordinates of the intersection;
- **Street, Section, Laneset, Lane**: street name, section name, whether it is preferential, delimiting nodes, laneset length, maximum speed, and width for the lane;
- **Turning Probabilities**: set of allowed vehicle movements and their probabilities;
- **Signal Plan**: set of lane-to-laneset allowed movements in a specific order, cycle time and offset;
- **Sink and Source**: nodes for removal and insertion of vehicle according to given values;

3.2 Simulation Module

This module was developed using C++ over the microscopic model presented before. However it also solves an issue inherent to the basic microscopic model described in Section 2, namely the difficulty to represent urban scenarios.

The simulation occurs in discrete steps and is implemented as a series of updates in the drivers' decisions of movement, followed by simultaneous updates in the vehicles' positions in the network. Each update in a node or traffic light may modify its current behavior. The simulation output can be formatted according to the user needs. The most usual formats are the "cell map" and the "laneset occupation map". The former indicates which portions of the lane are occupied by which vehicle, providing the most detailed output possible. On the other hand, the "laneset occupation map" is a high-level output which specifies the rate of occupation (density) for each laneset in the network.

3.3 Driver Modelling Module

Modeling drivers' behavior can be approached in different ways, depending on the purpose of the simulation. In most cases, the objective is to simulate the collective or macroscopic behavior. However, this behavior emerges out of individual ones. Simple algorithms, like the Nagel–Schreckenberg model, can be used to describe vehicles movement without loosing significant simulation fidelity with reality. However, other behaviors can be implemented, normally out of two different components: decision about movement and planning. The former is responsible for a vehicle's short-time movement, while the planning decision is related to a more sophisticated decision-making, e.g. which direction to turn and what to do when facing a jam.

3.4 Visualization Module

This is the module that allows the graphical visualization – either in a macroscopic or microscopic level – of the simulation results. At a macroscopic level, the visualization considers only data which reflect the overall behavior of the network, providing an useful tool to capture the big picture of what is happening in a specific scenario. It is useful to provide this kind of information via the internet, as it has been the case in [2]. The microscopic level provides an interface through which one can see, both in 2D and 3D manner, individual vehicles movement. In order to obtain a more realistic and detailed visualization, these modules are developed using OpenGL, enabling features such as walk-through navigation and detail-focused interfaces.

The visualization of data at microscopic level is the kernel of the information system for the traffic engineer and/or the urban planning experts. At this level, the whole system can be used to perform *what-if* simulations.

4. FINAL REMARKS

The ITSUMO system is capable of dealing with several aspects of the simulation of a traffic scenario, such as the driver behavior, traffic lights coordination, traffic jam prediction, among others. The ITSUMO system provides an useful tool to simulate traffic conditions and to support urban planning tasks.

We plan to extend ITSUMO to consider other kinds of information such as those related to weather forecast and information providing via internet and/or mobile phone.

5. REFERENCES

[1] Ana L. C. Bazzan and Franziska Klügl. Case studies on the Braess paradox: simulating route recommendation and learning in abstract and microscopic models. *Transportation Research C*, 13(4):299—319, August 2005.

[2] S. F. Hafstein, R. Chrobok, A. Pottmeier, M. Schreckenberg, and F. C. Mazur. A high-resolution cellular automata traffic simulation model with application in a freeway traffic information system. *Comp-aided Civil Eng*, 19(5):338–338, 2004.

[3] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *Journal de Physique I*, 2:2221, 1992.

[4] B. C. Silva, A. L. C. Bazzan, G. K. Andriotti, and D. Oliveira. ITSUMO: an intelligent transportation system for urban mobility. In *Proceedings of the Optimization of Urban Traffic Systems*, Lecture Notes in Computer Science – Innovative Internet Computing Systems 2004, pages 224–235, Guadalajara, Mexico, 2005. Berlin, Springer-Verlag.

Industry Track

Towards a Reference Model for Agent-Based Systems

Pragnesh Jay Modi, Spiros Mancoridis,
William M. Mongan, William Regli
Department of Computer Science
Drexel University
Philadelphia, PA 19104
{pmodi, spiros, wmm24,
regli}@cs.drexel.edu

Israel Mayk
US Army CERDEC C2D
AMSEL-RD-C2-SS-T
Fort Monmouth, NJ 07703
Israel.Mayk@us.army.mil

ABSTRACT

The current state of the art in agent technology sees that several implementations of agent frameworks exist. However, there is little agreement on the terms and concepts used to describe such systems, which is a significant barrier towards adoption of these technologies by industry, military and commercial entities. A clear definition of terms and concepts at an appropriate level of abstraction is needed to facilitate discussion, evaluation and adoption of these emerging agent technologies. In this paper, we argue that a *reference model* for agent-based systems can fill this need. We discuss what a reference model is, why one is needed for agent-based systems, and our proposed methodology for creating such a reference model. While the complete model is a work in progress, we present a preliminary version to motivate further discussion from the agents community at large. It is our hope that ultimately a wider community of practice will assume responsibility for the standardization similar to the way that the well-known seven-layer Open Systems Interconnection (OSI) reference model was a driving force underlying communications standards.

Categories and Subject Descriptors

H.1 [**Information Systems**]: Models and Principles

General Terms

Standardization

Keywords

Intelligent Agents, Reference Model

1. INTRODUCTION

The ultimate goal of the Agent-Based Systems Reference Model (ABSRM) is to provide a technical recommendation for a reference model for those who develop and deploy systems based on agent technology. The ABSRM should allow for existing and future agent frameworks to be compared and contrasted as well as to provide a basis for identifying areas that require standardization within the agents community. As such, the aim of the ABSRM is to

- establish a taxonomy of terms, concepts and definitions needed to compare agent-based systems;
- identify functional elements that are common in agent-based systems;
- capture data flow and dependencies among the functional elements in agent-based systems;
- specify assumptions and requirements regarding the dependencies among these elements.

As a reference model, the ABSRM will make no prescriptive recommendations about how to best implement an agent-based system; nor is the objective to advocate for any particular approach, architecture or framework. In its broadest sense, an agent-based system for the purposes of the ABSRM simply describes a software platform for building agents and supporting their communications and collaboration. An agent-based system may consist of many different kinds of agents operating across a heterogeneous set of platforms and hosts.

One novel aspect of our approach is to create the reference model based on a forensic analysis of existing agent-based systems. The reference model developed in this document is based on static and dynamic software analysis of existing agent frameworks. Examined frameworks include Cougaar, JADE, RETSINA, and others. Anyone building an agent framework would have to recreate or reproduce some portion of the functionality or components in these existing frameworks (i.e., to enable communications, to enable agent startup and shutdown, etc). By analyzing existing frameworks and the agent-based systems they can be used to build, we can avoid the debate concerning "what is an agent" and simply document the existing state-of-the-art systems that are called agent-based. The model aims to document a superset of the features and functional concepts in the set of existing agent frameworks. Given that there is significant variation between existing frameworks and the functions they may provide, the reference model should describe at an abstract layer the complete set of functional components across all examined agent frameworks.

It is important to note however that the reference model is not confined to being a description of capabilities of existing systems —it serves as a basis for situating the complete set of functions that anyone may want or need to have in an agent-based system. For example, security for mobile agent code is currently a vastly challenging problem that lacks a satisfactory solution. However,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

the lack of any established, uniform and generally accepted security system for mobile agents does not prevent the reference model for including a description of the security functions and facilities that an agent-based system may provide.

2. WHAT IS A REFERENCE MODEL?

A *reference model* provides appropriate abstractions for facilitating adoption, adaptation and integration of evolving technologies [18]. Any generic model that has specific examples can be considered to be a reference model. Reference models are known to play a key role in understanding a given domain, establishing the domain as a scientific discipline, facilitating collaboration and promoting competition towards maturing technology relevant to the domain. Reference models emerged since the 1980s as a result of the success of the Open Systems Interconnection seven-layer reference model (ISO/IEC 7498-1:1994: Open Systems Interconnection Basic Reference Model) [3] [4] [15] that revolutionized the way communications systems developed. This model was developed as an International Standards Organization (ISO) effort. Another successful example of a reference model that has been developed by the ISO is one for archiving systems and is known as the Reference Model for an Open Archival Information Systems (OAIS) (ISO 14721:2003). In motivating and developing a reference model for agent systems, we draw heavily on these examples of existing successful reference models.

As these existing reference models demonstrate, reference models do not prescribe how functions and systems should be implemented. Instead, reference models provide the patterns of the solution for transforming vague notions into real-world implementation. Reference models simplify problem solving, to enable others to practice their discipline with a solid foundation. Software professionals, in particular use reference models to better understand abstractions and their potential for reuse.

3. WHY A REFERENCE MODEL?

Reference models are a necessity in a confusing, rapidly changing technology environment. As noted in [18], reference models are becoming more commonplace in fields of various human endeavors. The power of compelling reference models of knowledge can not be underestimated as a tool for technical leadership facilitating conclusions from a sound understanding of the problem space and solution domain. Thus, we argue that a reference model would be very useful tool to identify, assess and facilitate R&D and acquisition of agent technology for a wide variety of applications.

One particular application of agent-based systems is in military domains. Agent-based systems are being proposed to enhance and automate applications for collecting, presenting, storing, producing and sharing domain information. A future force is envisioned to be highly autonomous, modular, scalable, and flexible through the agentization of applications and services. This is especially true for complex system of systems (SoS). SoSs are large-scale, net-centric and include a variable mix of multi-department heterogeneous intelligent agents, humans-in-the-loop, and unmanned autonomous components and subsystems. Examples of complex SoSs in the US Army are the Army Battle Command System of the Current Force [1] and the Future Combat System of the Future Force [2]. Systems that would be found in a Joint Service (Army, Navy, Marine and Air Force) SoS are depicted in Figure 1.

Heterogeneous intelligent agents promise to enable conflict resolution between and among applications and services engaged in competition, negotiation, mediation and arbitration, to assist humans-in-the-loop, and to control unmanned autonomous systems and robots.

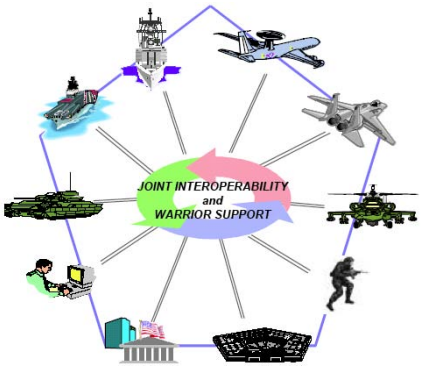


Figure 1: A Joint Service Battle Command is an intended future application for intelligent agent systems.

Agents are anticipated to play an important role in realizing dynamically varying mixed initiative capabilities to command and control manned as well as unmanned assets. As systems become increasingly complex, modularity as promoted by agent technology will become the key to reuse, scalability, and an open architecture. In addition, these design goals are a key to a manageable and affordable transformation from current to future force capabilities.

A reference model for intelligent agents would motivate the benefits of the technology by formalizing key concepts of both behavior and structure essential to enable agent technologies to reduce information collection, storage and sharing latency, workload, presentation overload and clutter for Battle Commanders and their staff. The recognition of the need and the investment to develop a unifying ontology for intelligent agent software across the domains of the systems in an enterprise-wide System of Systems (SoS) are shown to be crucial if not pivotal to the success of such SoS engineering efforts which are inherently multi-disciplinary and collaborative.

4. HOW TO CONSTRUCT A REFERENCE MODEL?

We take a quantitative and evidentiary approach to create a reference model based on a forensic analysis of existing agent-based systems. There are many agent frameworks available today: from companies, from academia and from the open source community. These frameworks have emerged from several large governmental and private research and development programs and have been used in the creation of many successful military and commercial systems. This suggests that a sensible approach towards building a reference model begins with examination of these existing systems and attempts to construct the model in a bottom-up fashion.

Reverse engineering (RE) is an ideal technique for examining and understanding existing systems. In RE, one obtains the software modules that comprise the subject systems by performing software analysis. By grouping, abstracting and querying data in different ways, information can be gleaned that simple observation of code may not find. Conventionally, RE is used to produce data that can allow for documentation and understanding of the software systems and for verification of any existing software documentation. For our purposes, we hypothesize that this data can also be analyzed and abstracted to obtain a reference model. In this section, we describe a specific RE technique we are using called *static analysis*. We also provide an overview of an additional RE technique that we will use in our future work: *dynamic analysis*.

4.1 Static Analysis

Static analysis is the analysis of software using its source code as the primary artifact. The system needs not be executing in order to obtain the appropriate data. Instead, source code or intermediate code is inspected to find the software modules, data structures, data flow, methods and metrics appropriate to the system.

This type of analysis yields many benefits, such as code-rewriting and vulnerability detection, but for purposes of our reference model, the primary goal is to use static analysis to produce a data repository from code that can be queried to find the primary software subsystems. This facilitates the transition from analyzing subject systems to identifying software modules that might fit the overall abstract system defined by the reference model. We describe two of the software analysis tools that we used: Bunch and JadeSniffer.

Bunch[1] is a clustering tool intended to aid the software developer and maintainer in understanding, verifying and maintaining a source code base, especially when when documentation of is nonexistent or outdated. The process proceeds in two steps. First, information from the source code is used to construct a directed graph that represents the key components, and the dependencies between these components. This directed graph, which we refer to as the Module Dependency Graph (MDG) of the system, has nodes as program units such as files or classes, and edges between the nodes as relationships between those modules such as function calls or inheritance relationships.

In the second step, Bunch uses the MDG to find a "good" clustering for the system. In particular, Bunch processes the MDG and produces as output a clustered MDG which includes all of the original source code and dependencies clustered into logical subsystems. Bunch can also use pre-defined clusters to measure or improve the quality of the system's clustering. The output from Bunch can be visualized by several popular graph visualization tools.

The Jade Sniffer[2] is a tool developed specifically to analyze Jade multi-agent systems. Jade is a FIPA-compliant agent framework, so a tool of this type is helpful to analyze behavior patterns and relate them to FIPA-compliant systems. This is accomplished by tracking messages between agents at the Virtual Machine level. The Sniffer itself claims to be a FIPA-complaint agent, like the Jade framework it analyzes. Using this tool together with Jade's logging services, we hope to produce sequence diagrams and message history that occurs between the agents; this provides a detailed view of the agent system behavior from a dynamic analysis perspective.

4.2 Dynamic Analysis

Dynamic analysis also collects data on software systems, but it does so by inspecting the system during execution. This analysis can vary widely by implementation, but one approach is to build a data repository of program behavior. This repository holds information on data flow, object instantiation, the call graph, interprocess communication, network or filesystem I/O activity, and so on. This analysis assists the production of the reference model by providing more sophisticated justification than static analysis alone. For example, dynamic analysis inspects the system as it is running and often breaks the system down into "features", e.g., code components that are actually used to provide some behavior. Moreover, dynamic analysis can obtain data on behavior-specific aspects of the system such as threading and I/O, that could not be found using static analysis. Finally, dynamic analysis can assist in cases where source code is not available for static analysis to be performed.

[1]http://serg.cs.drexel.edu/bunch

[2]http://jade.tilab.com/doc/tools/sniffer/

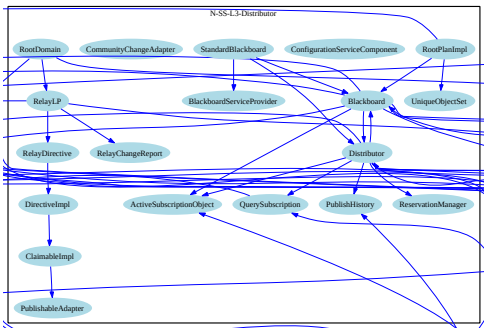


Figure 2: Diagram of Cougaar's Blackboard Distributor obtained using the Bunch reverse engineering tool.

5. EXISTING AGENT FRAMEWORKS

We describe three of the agent frameworks we have investigated to date: Cougaar, Jade and RETSINA. For each framework, we describe several of their functional concepts which were discovered and verified through a combination of static analysis of source code and accompanying documentation.

5.1 Cougaar

Cougaar (Cognitive Agent Architecture)[3] is a Java-based architecture for the construction of large-scale distributed agent-based applications. It was developed over two DARPA-sponsored research programs spanning eight years of effort. The goal of Cougaar is to provide developers with a framework to implement large-scale distributed multiagent applications with modular agent features and requiring minimal consideration for the underlying architecture and infrastructure. Below we describe a few of the key functions of Cougaar.

- *Directory Services:* Each agent is provided a unique name. This name is often similar to a network domain name based on the category and type of role that agent will have (though this is simply by convention). Cougaar provides a White Pages function that maps agent names to their locations. A Yellow Pages was also introduced to handle searches for agents and their locations based on their attributes or capabilities. Yellow Pages is built on top of UDDI, and a number of servers can provide the distributed Yellow Pages service.

- *Messaging:* Cougaar features a distributed blackboard, in which agents in a particular domain communicate with each other. Figure 2 shows a diagram obtained from applying the Bunch reverse engineering tool to the source code of Cougaar's blackboard component. In Cougaar, a software developer creates plugins that use a common language and protocol. These plugins publish objects to the blackboard, and receive objects from the blackboard according to an agent's subscriptions. To facilitate transmitting data to and from other agents, relay attributes are stored to show what information is desired from and appropriate for which agents. Alternatively, instead of storing actual recipient addresses, an agent can address a message based on attributes of the agent within the community.

- *Conflict Management:* To facilitate conflict management, an agent within the Cougaar architecture can form a group with other agents that share a similar goal or function. These

[3]http://cougaar.org

communities together form additional communities or a society. When the Cougaar society is active, new nodes may dynamically join the society, and may leave as well. To coordinate activities, nodes create functional relationships such as producer/consumer (service-based) relationships or superior/subordinate (delegation based) relationships. In order for agent communication to occur, there must exist established relationships between them.

5.2 Jade

Jade[4] is a Java based agent platform that is built upon a number of FIPA standards, including Agent Management Specification [9], Agent Communication Language (ACL), and the ACL Message Structure. Thus, Jade is structured as a FIPA compliant agent platform in which agents, the agent management system and the directory facilitator interact independently with a common message transport system.

- *Directory Services:* Agents are registered, named, and looked up via FIPA standard implementations. An Agent Management System (AMS) is provided for name lookups, and a Directory Facilitator is provided for agent lookups. Based on the location of the agent (local or remote), an agent name lookup will find the physical location of the agent and the Jade subsystem will forward the message over the appropriate I/O channels.

- *Messaging:* A peer-to-peer conversation occurs between agents. This is implemented via a FIPA standard, in which an Initiator and Responder agent communicate with each other through message passing. Feasibility Preconditions are met before the message is sent, and a Rational Effect are defined as to what the Initiator agent should expect (but not assume) upon receipt and processing of the message by the responder. Agents can inform each other or make requests, and they can be responded to with a not-understood, refusal or agreement response, and so on. In addition, agents may subscribe to messages provided by other agents based on their content.

- *Mobility:* Jade supports migration of agents to other hosts. This is done by copying or serializing the agent across the network. The jade-mobility-ontology is provided to ensure that agents are always aware of their physical location. At a minimum, agent mobility is described by an ontology that contains the agent's name and destination. It can also contain an agent profile which contains the OS that the agent resides / will reside within. It can also specify the system and language of the agent.

5.3 RETSINA

RETSINA[5] is developed by the Robotics Institute of Carnegie Mellon University. RETSINA is developed in C++ but is platform independent. Agent systems can be run on a number of different platforms and hosts (including handheld systems) and can be implemented in a number of languages.

- *Directory Services:* The ANS (Agent Name Service) is provided to map an agent by name to a physical location. This location is not bound to a particular physical medium such as TCP/IP, but instead can be any location that physically defines a method to contact the agent. In addition to a naming service, attributes and services of each agent are cataloged when it joins the society by *middle agents* called "Matchmakers." These matchmakers provide agent lookup based on a service query, and can also send notifications when new agents join the society that meet the specified criteria.

- *Messaging:* Messages are either direct or multicast to agents based on their services or attributes. Message passing can be synchronous or asynchronous, and can occur between multiple agents simultaneously via a threaded architecture. RETSINA also separates the physical communications model from the protocol, language and ontology. KQML is used as the agent communication language in RETSINA. As part of the language, each message passed between agents contains both the content and the structure (ontology, language, sender, conversation, etc.) of that message.

- *Semantic Interoperability:* A shared dictionary defines the meanings of content of KQML messages.

6. TOWARDS A REFERENCE MODEL

We describe a preliminary, partial reference model for agent-based systems. The portions of the model we present include a high-level view organized as a set of layers and a functional decomposition of an agent-based system.

6.1 Layered View of an Agent-Based System

An **Agent-Based System** is comprised of agents and their supporting framework and infrastructure which provide fundamental services and operating context to the agents. Our model defines framework, platform and host layers, which mediate between the agents and the external environment. This layered model can be organized vertically as shown in Figure 3. Each layer is described as follows:

- The **Agents** layer consists of agents that perform computation, share knowledge, interact and generally execute behaviors in order to achieve application level functionality. We make few assumptions about the Agent layer except to state that agents are *situated* computational processes—instantiated programs that sense and effect an environment in which they exist. We make no assumptions about the internal processing structures of an agent. An agent could be built with a complex cognitive model or it could be a simple rule-based system. Given the vast array of tasks envisioned for agent systems, it is not the role of a reference model to limit or define what an agent is.

- The **Framework** layer provides standardized functionality specific to supporting agents. A framework typically provides support services such as conflict management, directory and naming services, security, and agent administration services such as monitoring and allocating resources to the executing agents. The major benefit of employing an agent framework is to provide standardization of services and functionality to agents that exist within the framework. The end result is that agents written within a particular framework are easily interoperable with one another. In other agent-based systems, the framework may be trivial or merely conceptual, for example if the services are merely a collection of system calls or are compiled into the agents themselves.

- The **Platform** layer provides more generic computing infrastructure. The platform contains the software components that

[4]http://jade.tilab.com/

[5]http://www.cs.cmu.edu/ softagents/retsina.html

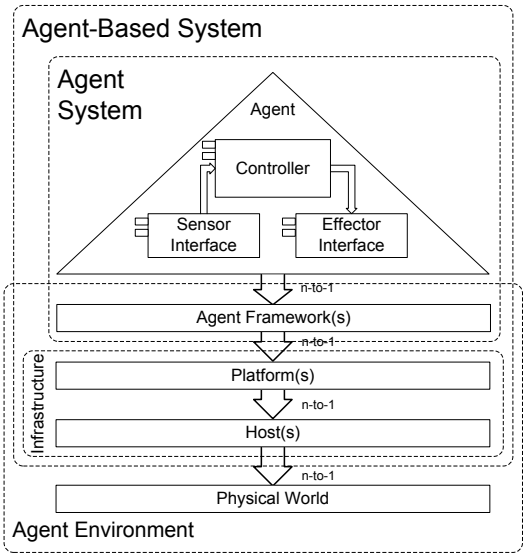


Figure 3: An Agent-Based System is made up of layers.

are available to the agent framework, but are not packaged along with it. Elements such as operating systems, user interface libraries, database software, device drivers, and message transport or socket libraries are in this layer. These services are often provided by third parties and it is unlikely that an agent-based system will provide its own implementation of the platform functions.

- The **Host** layer contains the hardware devices on which the above layers operate. This layer includes not only the physical computing devices such as a desktop computer or handheld device, but also the hardware that provides interaction with the environment such as robot sensors and effectors, cameras, displays, GPS receivers, etc.

- **Environment** is the physical world in which the agent-based system exists and operates.

Each layer can support many entities from the layers above it—many agents may execute on a single framework, many frameworks may execute on a single platform, and so on.

6.2 A Functional View of an Agent-Based System

This section presents a view of an agent-based system as a set of abstract functional concepts that support overall system execution. For example, *administration* and *security* are two abstract functional concepts we define. Our use of the rather abstract term "concept" here is deliberate. The more concrete (and perhaps more familiar) UML term "component" could be used, but we wished to reflect the idea that a function often does not correspond directly to what we might think of as a component, i.e., a clearly delineated piece of the system. Similar to the aspect-oriented software engineering view, we view a functional concept as something that emerges out of complex interactions between pieces of software and hardware located in different layers of the agent-based system.

It is important to note that we make few prescriptions about whether and how each functional concept is implemented. The way in which functional concepts are instantiated may vary significantly in structure, complexity and sophistication across different agent-based system implementations. Indeed, some agent-based systems may not even possess some of the functional concepts we will describe. Our aim here is to describe what the function is in abstract terms so that one can determine if the function exists in a given system, or to verify its existence if it is claimed to exist within a given system.

A functional concept is described through a brief definition and a set of primitive operations that implement the function which we call a Process Model. We define Administration, Security, Conflict Management, and Messaging concepts. Other functional concepts that are part of the model but not described here are Mobility, Semantic Interoperability, Directory Services, and Logging.

6.2.1 Administration

Definition: *Administration functionality a) facilitates and enables supervisory command and control of agents and/or agent populations and b) allocates system resources to agents. Command and control involves instantiating agents, terminating agents, and inspecting agent state. Allocating system resources includes providing access control to CPUs, User Interfaces, bandwidth resources, etc.*

Administration functionality may be implemented in various ways. For example, the framework may perform all the administration functions directly, or there may be (multiple) agent(s) in the agent layer that perform agent administration functions by commanding and controlling other agents, or there may be elements of both approaches in a given system. For convenience of exposition below, we will use the term "administrator" to encapsulate all the administration functions although administration functions may not necessarily implemented with a single administrator.

To further facilitate the exposition of the following process model, consider as an example a hypothetical system that uses agents to monitor message traffic on a communication network. The number of agents required to perform adequate monitoring may be contingent upon the complexity of the network topology or the priority of monitoring relative to other system goals. As both the network topology and priority changes, an administrator is employed to manage the network monitoring agents.

Process Model: Agent administration functionality is described by the following set of processes:

- *Agent Creation* The act of instantiating or causing the creation of agents. In our example, the administrator may determine that there are too few network monitoring agents to adequately maintain a minimum level of security. Therefore, new network monitoring agents will be created.

- *Agent Management* The process by which an agent is given an instruction or order. For example, if it is determined that the greatest security threat is over HTTP traffic, the administrator may request that the network monitoring agents focus their analysis on HTTP traffic.

- *Resource Control* The process by which an agent's access to system resources is controlled. For example, the administrator may determine that security is of less priority than CPU usage. Therefore, it can reduce the available CPU time of the network monitoring agents.

- *Agent Termination* The process by which agents are terminated (*i.e.*, their execution is permanently halted). For example, the administrator might determine that there are too many network monitoring agents and decide to remove those in saturated regions of the network.

6.2.2 Security

Definition: *The purpose of security functionality is to prevent execution of undesirable actions by entities from either within or outside the agent-based system while at the same time allowing execution of desirable actions. The goal is for the system to be useful while remaining dependable in the face of malice, error or accident.*

Process Model: Security functionality is described by the following processes.

- *Authentication* A process for identifying the entity requesting an action. Common examples include user name/password credentials and use of public/private keys for digital signatures.
- *Authorization* A process for deciding whether the entity should be granted permission to perform the requested action. A common example in file system security is maintenance of a permission list for each file, which specifies the allowable actions for a given user. Another example includes a web server denying requests when it gets overloaded.
- *Enforcement* A process or mechanism for preventing the entity from executing the requested action if authorization is denied, or for enabling such execution if authorization is granted. A common example for preventing access to information is to encrypt it. Permission to access the information is granted by providing the entity a decrypted copy or providing the entity the means to decrypt it, e.g., the encryption key.

Some general technologies for achieving security include authorization models and mechanisms; auditing and intrusion detection; cryptographic algorithms, protocols, services, and infrastructure; recovery and survivable operation; risk analysis; assurance including cryptanalysis and formal methods; penetration technologies including viruses, Trojan horses, spoofing, sniffing, cracking, and covert channels.

6.2.3 Conflict Management

Definition: *Conflict management functionality facilitates and enables the management of interdependencies between agents activities and decisions. The goal is to avoid incoherent and incompatible activities, and system states in which resource contention or deadlock occur.*

As an example, a framework may allow designation of superior/subordinate relationships between agents and provide generic conflict resolution services based on these relationships. The Cougaar framework does this. Similarly, a framework may provide a multiagent task planning language, such as TAEMS [17], that can be used to reason about the interactions between agent actions and to detect plan conflicts.

Process Model: Conflict management functionality is described by the following processes.

- *Conflict avoidance*. A process or mechanism for preventing conflicts. Examples of such processes include multiagent planning algorithms (both on-line and off-line) that take care to produce action plans that do not have conflicts.
- *Conflict detection*. The process by which it can be determined when a conflict is occurring or has occurred. One example includes a plan execution monitoring algorithm that is able to sense when the actions of agents are in conflict. Another example includes performing logical inference over different agents beliefs to determine when they are inconsistent with one another.
- *Conflict resolution*. The process through which conflicts between agent activities are resolved. Negotiation, mediation and arbitration are common mechanisms for facilitating conflict resolution.

Some general technologies for conflict management in agent systems include argumentation and negotiation, distributed constraint reasoning, game theory and mechanism design, multiagent planning, norms, social laws and teamwork models.

6.2.4 Messaging

Definition: *Messaging functionality facilitates and enables information transfer among agents in the system.*

This concept is associated specifically with the mechanisms and processes involved in exchanging information between *agents*. Although information exchange via messages can and often does occur between other parts of the system, for example between an agent and its framework, between frameworks, between a host and its platform, etc., such information transfer is not included because it is in a sense at a lower level. Our concept of messaging is at a higher level than that associated with network traffic or interprocess communications.

Messaging involves a source, a channel and a message. Optionally, we can designate a receiver but we admit models in which messages do not have a specific intended receiver. For example, signaling in the environment like smoke signaling, a light flashing morse code, etc, are examples of messaging where there is no designated receiver.

Process Model: The functionality is described by the following processes.

- *Message Construction* The process through which a message is created once a source agent has determined it wishes to deliver a particular message chosen from a finite or infinite set of messages. No commitments are made here in regard to the form, structure or content of a message. For the purposes of this model it is sufficient to discuss messages as an abstract object. The information to be delivered is simply the fact that a particular message was chosen among the set of all messages that could have been chosen.
- *Naming and Addressing* A mechanism for labeling the message with its intended destination or route. Directory white page services are a common mechanism to facilitate this function. Broadcast, multicast and group messaging also all fit within this model.
- *Transmission* The actual transport of the message over the channel. This may be a one-shot transmission or a continuous stream. One common model is messaging an agent on another host involves going through the platform to the host's network hardware, then out into the environment (via wire or air), and back in symmetrically to the receiver.
- *Receiving* The process for acquiring the transmitted information so that is usable by the receiver. This may be as simple as pulling the message off of a queue or more elaborate, e.g., going through a translator.

Some other areas of interest in messaging functionality include notions of best effort delivery, QoS and guaranteed delivery/timeliness.

7. RELATION TO EXISTING STANDARDS

Existing standards for agent-based systems can be used concurrently with a reference model. A reference model describes the abstract functional elements of a system. A reference model does not impose specific design decisions on a system designer. Existing standards such as the Knowledge Interchange Format (KIF), KQML, those developed by the Foundation for Intelligent Physical Agents (FIPA), and even some non-agent specific standards have a place in the reference model and in fact are needed by the reference model. The reference model states that a component needs to exist, the standards tell someone how to design it.

A reference model also does not define an architecture. A reference model can drive the implementation of multiple architectures in the same way that an architecture could drive multiple designs, or a design could drive multiple implementations. In the same way, this reference model is not an attempt to impose (or even define) standards for the implementation of an agent-based system. It is an attempt to describe the abstract layers and concepts that may exist in such a system.

We have classified existing standards for agent-based systems roughly into five categories:

- *General* standards that are higher-level, and less specific. Examples include the FIPA Abstract Architecture [5] and FIPA Agent Management [9] standards.
- *Communication* standards that deal with both the actual logistics of communications and the information being communicated. Examples include KIF [14], KQML [16], FIPA ACL Message Structure [8], and FIPA Message Transport Service.
- *Coordination protocol* standards for coordinating actions of agents. Examples include the FIPA Contract Net Interaction Protocol [10] and FIPA Recruiting Interaction Protocol [13].
- *Representation and Encoding* standards for encoding messages for transmission. Examples include FIPA Bit Efficient ACL [6] and FIPA XML ACL [7].
- *Ontologies* standards for knowledge representation. Examples include FIPA Device Ontology [11] and FIPA QoS Ontology [12].

8. FUTURE WORK

Our goal is to continue refinement of the reference model in consultation with the broader agents community. We will be making available an on-line interactive discussion forum, similar to a Wiki, for facilitating further development of the ABSRM.

Acknowledgments

We thank Christopher Dugan, Moshe Kam, Joseph Kopena, Robert Lass, Jeff Salvage, and Evan Sultanik for significant contributions to this work. We also thank Tedd Gimber, Bernard Goren, Michael Huhns, James Odell, Randy Reitmeyer and Todd Urness for useful discussions.

9. REFERENCES

[1] Army battle command systems (abcs). http://www.fas.org/man/dod-101/sys/land/abcs.htm.

[2] Future combat system (fcs). http://www.fas.org/man/dod-101/sys/land/fcs.htm.

[3] *ISO 7498(Draft): OSI Basic Reference Model*. 1984. Information Processing Systems.

[4] Uyless Black. *OSI: A Model for Computer Communication Standards*. Prentice Hall, 1991.

[5] Foundation for Intelligent Physical Agents. Abstract architecture, December 2002. http://www.fipa.org/specs/fipa00001/.

[6] Foundation for Intelligent Physical Agents. Acl message representation in bit-efficient encoding, December 2002. http://www.fipa.org/specs/fipa00069/.

[7] Foundation for Intelligent Physical Agents. Acl message representation in xml, December 2002. http://www.fipa.org/specs/fipa00071/.

[8] Foundation for Intelligent Physical Agents. Acl message structure specification, December 2002. http://www.fipa.org/specs/fipa00061/.

[9] Foundation for Intelligent Physical Agents. Agent management specification, December 2002. http://www.fipa.org/specs/fipa00023/.

[10] Foundation for Intelligent Physical Agents. Contract net interaction protocol specification, December 2002. http://www.fipa.org/specs/fipa00029/.

[11] Foundation for Intelligent Physical Agents. Device ontology specification, December 2002. http://www.fipa.org/specs/fipa00091/.

[12] Foundation for Intelligent Physical Agents. Quality of service specification, December 2002. http://www.fipa.org/specs/fipa00094/.

[13] Foundation for Intelligent Physical Agents. Recruiting interaction protocol specification, December 2002. http://www.fipa.org/specs/fipa00034/.

[14] M. R. Genesereth and R. E. Fikes. Knowledge Interchange Format, Version 3.0 Reference Manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, Stanford, CA, USA, June 1992.

[15] B. N. Jain and A.K. Agrawala. *Open Systems Interconnection: Its Architecture and Protocols*. McGraw-Hill, 1993.

[16] Yannis Labrou and Tim Finin. A Proposal for a new KQML Specification. Technical Report TR CS-97-03, CS and EE Department, U. of Maryland Baltimore County, Baltimore, MD, USA, February 1997.

[17] V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. NagendraPrasad, A. Raja, R. Vincent, P. Xuan, and X.Q. Zhang. Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. *Autonomous Agents and Multi-Agent Systems*, 9(1):87–143, July 2004.

[18] Raphael Malveau and Thomas J. Mowbray. *Software Architect Bootcamp*. Prentice Hall, 2001.

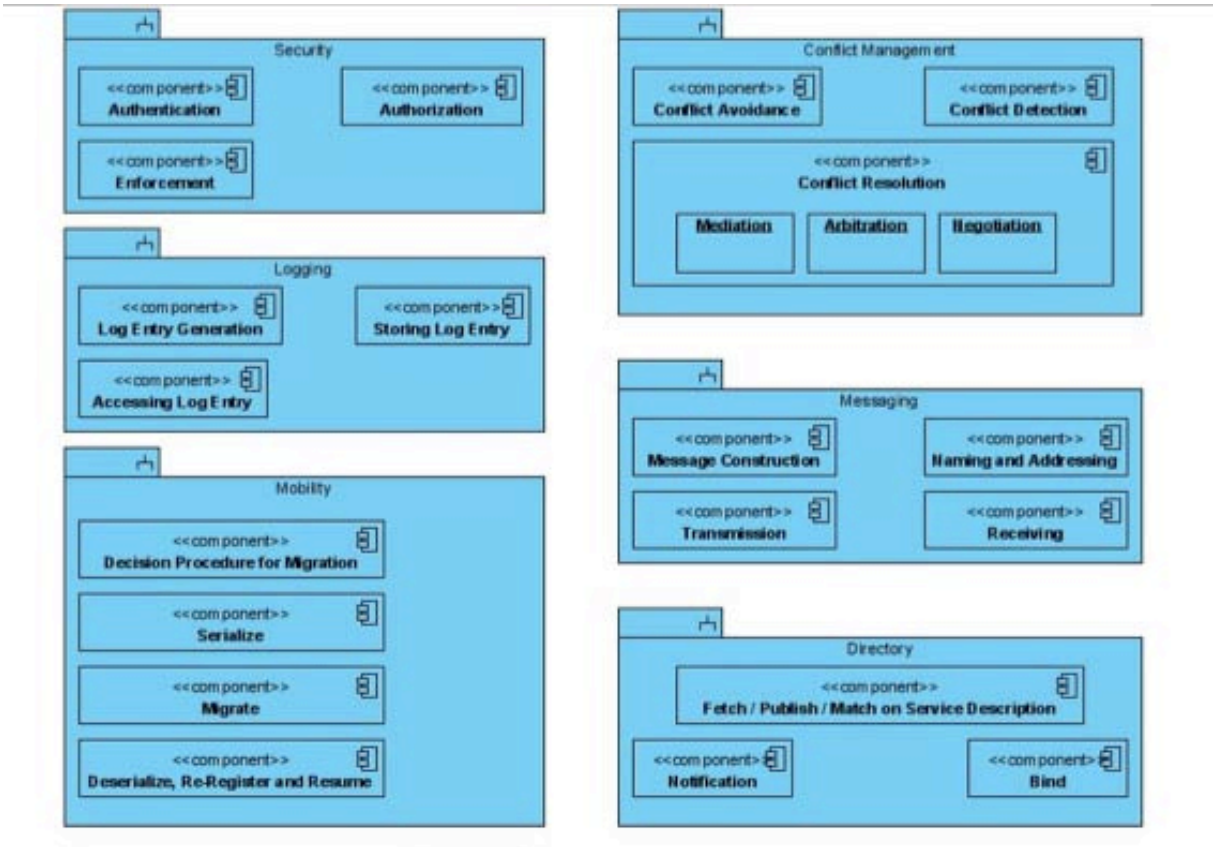


Figure 4: A Functional View of an Agent-Based System. selected functional concepts shown here in standard UML notation.

Automating Deep Space Network Scheduling and Conflict Resolution

Mark D. Johnston and Bradley J. Clement
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109 USA
{mark.johnston, bclement}@jpl.nasa.gov

ABSTRACT

The Deep Space Network (DSN) is a central part of NASA's infrastructure for communicating with active space missions, from earth orbit to beyond the solar system. Consisting of more than a dozen major ground antennas at three sites spaced around the globe, it must be carefully scheduled to satisfy the requirements of the various mission users, subject to many constraints. Scheduling the communication services for these missions is a fairly large scale negotiation problem since each mission cannot give up control of its spacecraft. While we will give details of this problem, this paper focuses more on describing an agent that provides scheduling advice to a single user within a collaborative scheduling system being developed. This agent can provide guidance to the user about feasible and optimal solutions to the problem they are working on. We describe our recent work in modeling the complexities of user requirements, and then scheduling and resolving conflicts on that basis.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Application and Expert Systems – *office automation*

General Terms

Design, Human Factors, Algorithms, Performance

Keywords

Keywords are your own designated keywords.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, *May 8-12, 2006, Hakodate, Hokkaido, Japan.*
Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

1. INTRODUCTION

The NASA Deep Space Network (DSN) is an international network of antennas that supports interplanetary spacecraft missions and radio and radar astronomy observations for the exploration of the solar system and the universe. The network also supports selected Earth-orbiting missions. The DSN consists of three deep-space communications facilities placed approximately 120 degrees apart around the world: at Goldstone, in California's Mojave Desert; near Madrid, Spain; and near Canberra, Australia. This geographic placement permits constant observation of spacecraft as the Earth rotates.

Each of the DSN sites supports antennas of sizes 26m, 34m, and 70m, along with a variety of receivers and other equipment that can be configured to support all the different kinds of communications requests that must be handled. Fig. 1 shows a picture of one of the largest antennas, the 70m antenna at Goldstone, California, along with one of the 34m antennas at this same complex.



Fig. 1: The 70m antenna at the Deep Space Network Goldstone complex in the California Mojave desert. At the right can be seen one of the 34m antennas.

The missions supported by the DSN include all of NASA's deep space missions, such as the planetary exploration spacecraft at Mars (the Mars Exploration Rovers Spirit and Opportunity, Mars Odyssey, Mars Global Surveyor) and Saturn (Cassini-Huygens); cometary explorers such as Deep Impact and Stardust; and missions such as Voyagers 1 and 2 that are leaving the solar system entirely. Other missions supported include observatories such as Chandra (X-ray) and Spitzer (Infrared), in Earth orbit or near Earth orbit. A total of about 20 spacecraft are currently allocated resources in a typical near-term scheduling period, while at the writing of this paper, 61 missions are being scheduled over the ~10 year horizon.

In this paper we will describe the nature of the DSN negotiation problems, i.e. the allocation of DSN resources to the missions that request them. We will discuss our progress in modeling the requirements of the missions, and in generating schedules that meet these requirements. We will discuss how an agent can automate the negotiation process, and we describe an agent that helps automate conflict resolution in what is currently a manual and iterative process. Key elements of our approach include tools for visualizing the schedule in various ways, and a search engine that looks for possible resolutions of conflicts to suggest to the user. We conclude with a discussion of future research and implementation directions.

2. THE DEEP SPACE NETWORK SCHEDULING PROBLEM

Currently, DSN schedules are manually generated out one year in the future. The nearer term portion of the schedule (up to 8 weeks out) considers the specific equipment and personnel to be allocated to a communications event. Beyond this interval, the schedule is concerned primarily with scheduling *tracks*, each of which is an allocation of an antenna to a mission over a specific time period. Tracks are typically one to eight hours in duration and must be placed within a *viewperiod* for that particular mission, i.e. when the spacecraft is visible (within specified geometric limits) from that antenna. A typical week will contain about 370 scheduled tracks. In this paper we are concerned with mid-range (8-week to 6-month) and long-term (beyond 6-months) schedules.

2.1 Constraints

Among the constraints that come into play when generating the DSN schedule are the following:

- A spacecraft can only be scheduled to use an antenna when it is within view: this is primarily a geometric constraint and can be pre-computed ahead of time into a set of usable viewperiods for each mission/antenna pair
- No two spacecraft can use the same antenna at the same time, with the exception that multiple spacecraft in view can downlink at the same time, but only one can uplink. Thus, for example, the multiple missions currently at Mars can share antennas for some activities.
- Tracks have *setup* and *teardown* durations that typically range from 10 minutes to an hour. (The combined setup, track and teardown is called an *activity*.) These setups and teardowns can be scheduled outside of the spacecraft viewperiod but generally cannot overlap other activities on the same antenna.

2.2 Requirements

The missions that use the DSN initially specify their requirements in high-level terms of hours/week per antenna, with more detailed specifications communicated informally (verbally or email) by the missions to the schedulers. One of the challenges faced in automating the DSN scheduling problem is that of defining a representation that can capture the details and subtleties of the individual mission's requirements. The representation must be highly expressive, since mission requirements can incorporate any of the following elements:

- *And/Or* groupings and alternatives. Missions must be able to specify conjunctions of requirements (such as multiple antennas at the same or different times). Such a conjunction means that all of the subrequirements must be satisfied at the same time in order for the requirement to be satisfied. In addition, disjunctive requirements may be specified, such that at most one requirement of a set of alternatives must be satisfied. In general, these requirements can be organized into an AND/OR tree, which is an ideal way to capture the combination of groupings (and) and alternatives (or) that arise when specifying a mission's requirements in detail.
- *Timing*. Tracks may have specified start times and durations, both of which may have allowable ranges.
- *Antennas*. Missions may specify one or more alternative antennas on which a track may be scheduled. They may specify groups of antennas, any one of which is acceptable for a particular type of track.
- *Repetitions*. Track repetitions may be specified in general as requirements, over some time period, on the number of tracks, their durations, overlap, and time gaps to neighboring tracks. All of these quantities may have ranges on them. So, for example, a mission might specify repetition requirements such as "seven tracks/week, of duration 5 to 8 hours with no gap longer than 20 hours between adjacent tracks".
- *Overrides*. In some cases, missions need to override a general requirement with a more specific one based on knowledge of ongoing activities. For example, a general requirement for a telemetry downlink every 10 hours may be superceded around the time of a maneuver with a requirement for continuous coverage.
- *Activity start and end times* for most missions must be scheduled on five-minute boundaries.

2.3 Negotiation

Missions are responsible for generating their own schedules, and although these may be automatically generated, conflicts will arise when two or more missions prefer the same antenna or equipment. While primary missions have priority over extended missions (those for which the primary mission has already been completed), there are no other precedence relations on missions or their particular requirements.
Thus, conflicts must be negotiated. If conflicts cannot be resolved, they are escalated, and a committee decides who wins. Currently, there are bi-weekly meetings up to four hours long where mission representatives and scheduling personnel try to resolve two weeks of conflicts scheduled about a half year into the future. General loading conflicts predicted beyond the half year are worked out twice a year in a meeting of more than 100 staff. Other meetings are held month-to-month. Outside of that mission representatives "horse trade" allocations to get what they need. While most missions get what they need, this process is obviously time-consuming and expensive.

There is an opportunity to significantly automate this negotiation. Using an automated scheduler to recommend solutions can greatly reduce human effort in finding resolutions to conflicts, but it does not relieve the communicative part of the negotiation process. One approach is to use an automated multi-objective scheduler to try to find solutions that favor all missions involved. This can reduce the number of rejections and counterproposals in the negotiation. Another complementary approach is for the mission to allow an agent to generate proposals and responses on its behalf. While the human user would not want an agent to make all decisions, in some cases the decision is obvious. For example, the agent should always refuse proposals to reschedule a critical track, and the agent should always accept proposals that fulfill unmet requirements. For proposals to which the agent may not respond, the agent can make suggestions, including counterproposals (based on multi-objective scheduling). As the user gains more confidence with the agent, it can define situations in which the agent is allowed to respond. Not only can this reduce the human involvement in negotiation, but also more negotiations can take place in order to create better overall schedules. If the use of such agents is adopted, distributed scheduling research could have great impact on this problem.

3. AUTOMATING DSN SCHEDULING

Our approach to automating the DSN scheduling problem is illustrated in the architecture diagram of Fig. 2. At the top are shown schematically the DSN data sources such as viewperiods and current schedules in various stages of completion and conflict resolution. This information is available via web services and network databases.

The Interactive User Environment, depicted in the lower left of Fig. 2, consists of a graphical requirements editor, a schedule timeline display function, and a local repository for requirements.

The Scheduling Agent, shown in the lower right of Fig. 2, incorporates three major approaches to schedule generation and repair: a local heuristic search technique and a set of systematic backtrack algorithms, and an intelligent assistant that suggests alternative schedule changes to the user. The API allows the scheduling computation to be done remotely on a grid while making it easier to switch out scheduling algorithms or intelligent assistants. In the following we describe each of these elements in more detail.

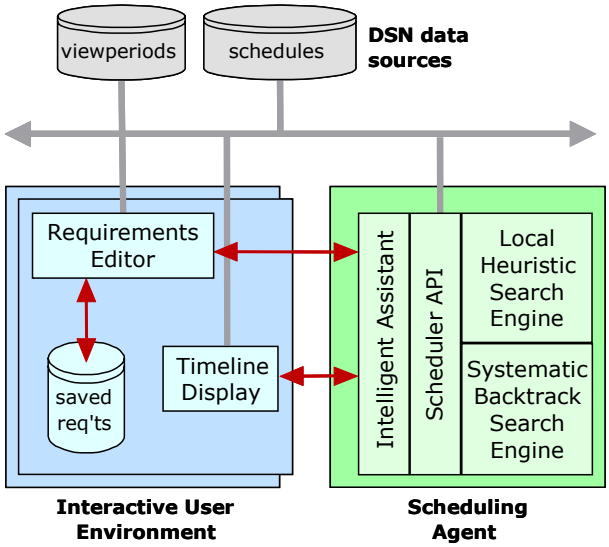


Fig. 2. Architectural overview of automated DSN scheduling

3.1 Interactive User Environment

The variety and complexity of the requirements described above provide a challenge for users. To help create and modify mission requirements, we have developed a graphical requirements editor that handles the full expressiveness of the requirements specification for the DSN. Fig. 3 shows a screen snapshot of this editor: on the left is a tree that represents the AND/OR tree of requirements and sub-requirements. As a node is selected, all of its attributes are displayed on the right side of the panel where they may be edited by the user, including the selection of antennas, timing, segmentation, and repetition. Requirements may be saved for further editing, or sent on to the scheduler as described below.

The timeline display provides the user with visual insight into the evolving schedule. We have adapted a standard Gantt-chart view and added additional features to make it easier to navigate and drill down for more information. These features displayed in Fig. 4, 5, and 6 include:

- activity start and end times, as well as setup and teardown times indicated by faint gray bars
- a schedule difference display that visually highlights tracks added, tracks deleted, and tracks changed. Changed tracks

can be clicked with the mouse to reveal a table display of the track attributes before and after, color encoded to show exactly which attributes have changed

- a metric-shaded background, to help visually spot regions of interest in the schedule. In the example shown, the shading is by gap duration in a 24h period, such that the bright shades of red indicate times with a scarcity of contacts.

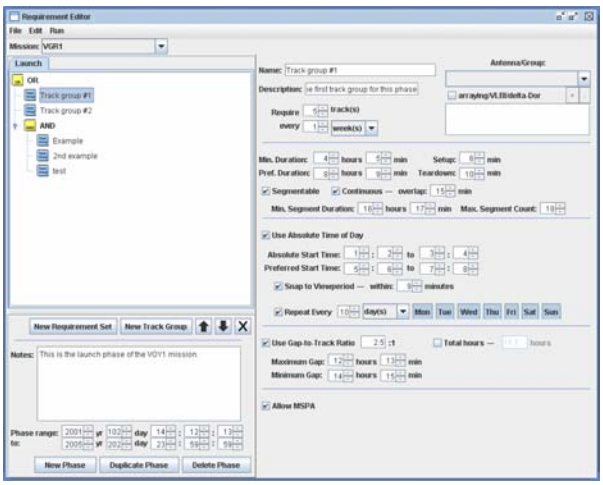


Fig. 3: The graphical requirements editor.

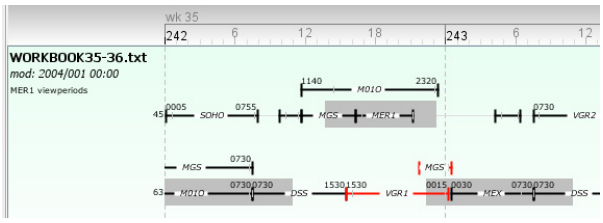


Fig. 4: Gantt view with annotations and conflict indicators

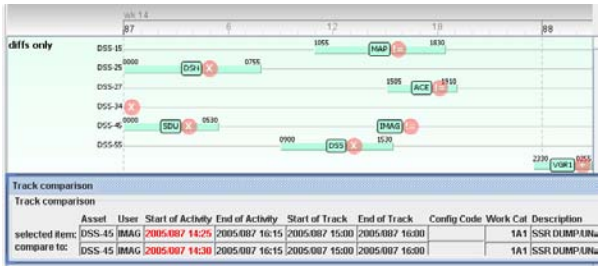


Fig. 5: Gantt view with schedule difference display

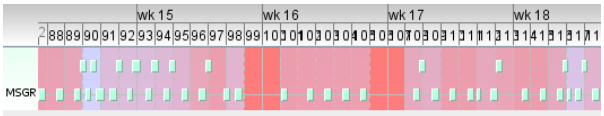


Fig. 6: Example of metric shaded background in the Gantt view

3.2 Local Heuristic Search

We have adapted the Aspen scheduler [10] to schedule all of the requirement types described above. This adaptation consists of two major components:

(1) a *mapping* from mission requirements (as described above) into Aspen's native modeling language [10]. A large portion of this mapping makes straightforward use of Aspen's built-in modeling features, including tasks and their hierarchical expansion into subtasks; temporal relationships; states; resources (consumable and nonconsumable); and parameters and their relationships. In some cases we have exploited Aspen's extension point mechanism to define external functions that can be invoked by Aspen. Details are provided in [11].

(2) a *set of heuristics* to drive Aspen's iterative repair-based search. These heuristics are based on "conflicts" in the schedule, such as state, resource, or temporal constraint violations, un-expanded hierarchical tasks, or un-propagated parameter dependencies. Part of the conflict selection and repair process incorporates randomness to help ensure a broad exploration of the search space. For a further description of the heuristics, see [11].

We have conducted experiments to gauge the performance of Aspen in this domain, with satisfactory results. In one experiment, we generated a full artificial schedule of 1861 tracks from periodic track requests in 39 CPU minutes. Modifications to this schedule were much faster: emergency antenna downtime or additional tracks were scheduled in 0.2 CPU seconds. Another experiment scheduled a week (274 tracks) of actual user requirements of the form of "*x* tracks of duration *y* on antennas (*a* & *b*) or (*c* & *d*)," resolving 77 out of 79 conflicts in 151 CPU seconds.

While the stochastic heuristic search technique embodied in Aspen has many advantages in this domain, it also has some disadvantages. The most important of these is that Aspen's search is not complete: it may fail to find a solution even if one exists. If solutions are sparse, the local search process may simply fail to find any in the time allotted (the "needle in a haystack" situation). To address this, we have complemented Aspen's search approach with several systematic search algorithms, as described in the next section.

3.3 Systematic Backtrack Search

We have implemented several systematic search algorithms based on constraint propagation and backtracking search. This approach, when appropriately limited in scope, can definitively answer the question of whether a solution exists and can additionally provide suggestions as to which requirements are overconstraining a problem, thus making it unsolvable.

The systematic search algorithms we have implemented are based on the following:

- search over time and resource assignments that start from the existing schedule (with or without constraint violations), so that alternatives "close" to the existing schedule are searched first
- tracks to assign are selected using a most constrained first heuristic
- backtracking is "fail fast", such that whenever any track has all domain values (times and resources) in conflict, the search immediately backtracks
- node and arc consistency is maintained based on track duration, view period windows, and resource availability

In addition to a basic backtracking search algorithm, we have also implemented an A* optimizing search, based on an objective that is a function of the time and resource assignments for each track. We have implemented several different objectives:

- Minimum total change: the smallest possible shifts in track times and antenna assignments for all tracks under consideration
- minimum number of missions affected by changes: this will allow more and larger changes, provided the number of missions impacted is smaller (thus limiting the number of parties who have to negotiate the changes)
- an overall preference function on track start times $\Sigma f_i(t_s)$ that can express mission-specific preferences on when tracks are scheduled

Systematic search can be exponentially costly even with these techniques, so we bound its application by considering the following:

- limit to only a subset of the fully expressive requirements language
- limit the time horizon of the schedule considered
- limit the activities to consider, by locking a substrate of activities on the schedule and search only over a subset which is allowed to move
- limit the CPU time expended in the search

Our experience to date indicates that frequently there are "pockets" of conflicts that occur together, which have limited impact on the schedule as a whole. In this situation, systematic search can be bounded and yet still provide valuable insight into solution possibilities.

To investigate the performance of the systematic search algorithms, we ran a number of experiments on mid-range and short-term schedules. These first of these was on a 14 week period after the initial conflict resolution process. We considered each day in the schedule separately, and ran three different systematic search algorithms on each:

- **BT1**: standard depth-first backtrack search
- **BT2**: same, but tracks were prohibited from changing their antenna assignments
- **A***: search for all "minimum change" schedules

The results are illustrated in Fig. 7. Of the 98 days examined, 28 had track resource/timing conflicts, ranging from one to four per day. Of those 28 days, one had no feasible solution and would have to have requirements relaxed in some manner by the respective missions. The other 27 had solutions that were quickly found with BT1, in an average time of 10ms. The typical change was relatively small: an average 1.9±1.4 tracks were changed, with a total time shift (summed over all tracks) of 6.8±11.5 hours. BT2 only found one improvement over the solutions found by BT1. Most of the solutions found by BT1 (16 out of 27) were optimal by the "minimum change" metric described above. In 5 cases, the A* search found better solutions, and in 6 cases A* was stopped at a 1000s run time limit before finding any optimal solutions. The computational cost of A* was both higher and more varied than BT1 or BT2: the mean run time was 18±52s.

As noted, this schedule represents the time after initial conflict resolution has been worked on by the schedulers and the mission users. We also selected a 2-week schedule snapshot taken before this conflict resolution process and ran the same experiments. The results are shown in Fig. 8. In this case 13 of the 14 days show conflicts, all of which were quickly resolved with BT1. The typical changes were much larger, however: an average 4.0±4.2 tracks were changed, with a total time shift of 19.7±26.9h. This reflects the smaller effort expended to resolve conflicts at this stage in the scheduling process. The A* algorithm found better solutions for 6 days, and no improvements for another 4 days. The A* runtime showed greater variability at 71±229s.

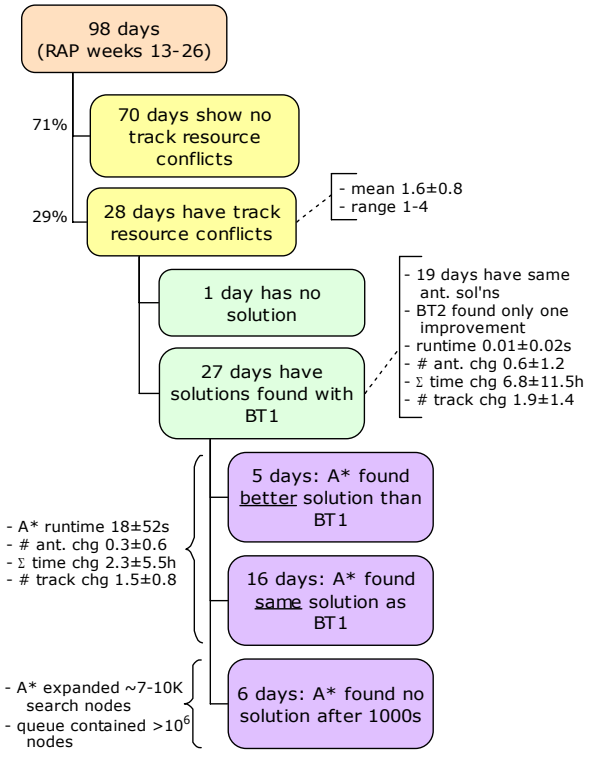


Fig. 7: Experimental results on a 98-day schedule span

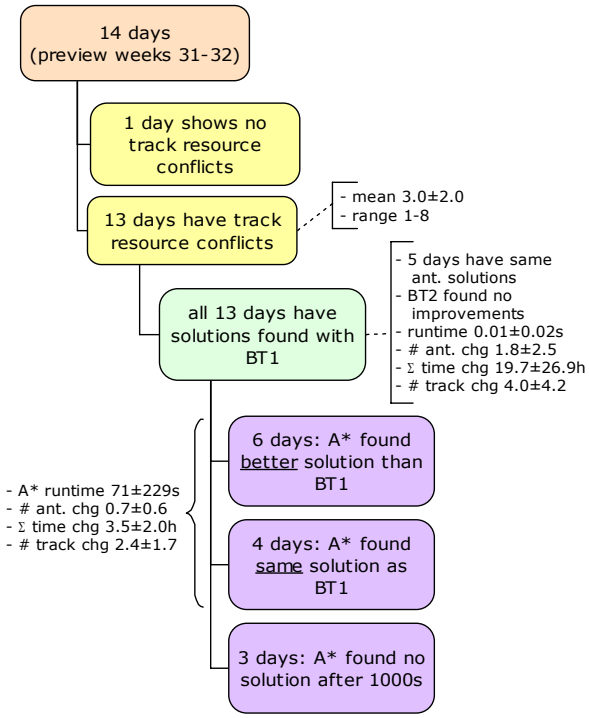


Fig. 8: Experimental results on a 14-day schedule before conflict resolution

3.4 Intelligent Assistant

The conclusion from these experiments is that systematic search can help identify solution possibilities when sufficiently bounded. We have integrated this functionality into the user interface elements described above by implementing an intelligent assistant that consists of

- a *filtering mechanism*, currently text-based, to allow the user to focus attention on a subset of the schedule: by time, mission, resource, etc.
- multiple *background asynchronous search processes* that run on the filtered problem and utilize the systematic search algorithms described above.

A snapshot of this interface is shown in Fig. 9. The intelligent assistant's background processes run automatically, starting whenever the filter criteria are changed by the user. They may be canceled by the user at any time, and they display current status as they run, both in a progress bar and in a textual status message. The BT1 and BT2 algorithms complete so quickly in our experiments (<10ms) as to be essentially instantaneous. The A* algorithm is much slower and more variable in duration: the status displayed in the progress bar shows how many of the total tracks have been successfully scheduled so far. In each case, when complete, the user can mouse over the status display to see a popup window with more detail, and can then click the "More" button to open a Gantt chart view of the schedule generated by the algorithm, along with highlighted differences from the original schedule.

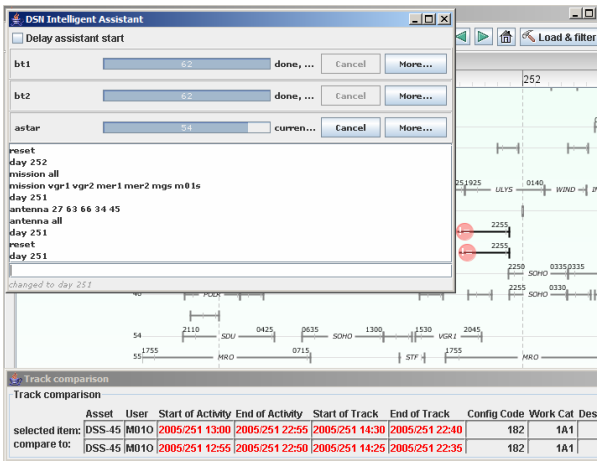


Fig. 9: Intelligent assistant dialog and display

By being unobtrusive, the intelligent assistant can provide the user with additional information at the precise time when the user is interested. Results are displayed immediately when they are calculated, and the user can cancel an ongoing search simply by revising the search criteria, which also automatically starts a new search using the new criteria. If the assistant generates an interesting schedule, the user can "drill down" into the details in a very straightforward manner.

4. RELATED WORK

There have been a number of previous investigations of DSN scheduling automation. The Operation Mission Planner (OMP-26) used heuristic search to allocate 26m antennas to missions and linear programming to adjust track time intervals [1]. Other automation projects were research efforts and were never deployed. LR-26 was a customisable heuristic scheduler that utilized Lagrangian relaxation [2], while the Demand Access Network Scheduler (DANS) employed an iterative repair technique [3]. Other graphical planning tools that have been used for forecasting, analysis, and interactive scheduling include TIGRAS [4] and FASTER [5]. Our approach combines some of the strengths of these systems, as well as incorporating innovative elements in both scheduling search and in schedule visualization.

Other scheduling domains pose problems that have similar characteristics to the DSN mid- and long-term scheduling problem. The Air Force Satellite Control Network (AFSCN) also schedules large numbers of satellites and ground stations but is limited to one day at a time. For this oversubscribed problem, there appear to be significant "plateaus" in the search space, that make strategies that take larger search steps more effective [6, 7]. Another related problem is that of fleets of Earth observing satellites with onboard resource limitations as well as tight viewperiod constraints [8]. In these problems, where the requests are prioritized, the goal of finding a best subset to fit on a

schedule has been addressed by a greedy approach with texture-based heuristics [9].

Automated sequence generation for spacecraft is different in that the horizon of interest is smaller (e.g. one week for an orbiter, one day for a rover) and the model of the spacecraft is very intricate. Safety is the highest priority in sequence scheduling, but efficiency is the goal of DSN resource allocation. Thus, the scheduling problem is very different.

5. DISCUSSION

We have developed and described a "hybrid" approach to automating DSN scheduling. An overall schedule for some time period can be generated based on the complex merged requirements from the different missions: for this phase, we make use of the ASPEN framework which implements an iterative repair-based search strategy. We have complemented this with a set of systematic search algorithms that work on an appropriately bounded region of the search space. We have embedded these algorithms in an agent that runs unobtrusively to provide timely information to the user engaged in rescheduling and resolving schedule conflicts. The agent can provide feedback immediately as it draws conclusions. For example, it may be possible for the scheduling agent to rapidly determine that there are no feasible solutions to a specific conflict situation, so a user can move on to consider changing the requirements or negotiating with another mission. Similarly, the agent may identify an optimal set of choices to resolve a conflict so that the user's time can be better spent in choosing among them rather than trying to find a solution by manual experimentation. We have found that, while systematic search does not always find timely solutions, it frequently does very quickly provide valuable help to the interactive user.

Future research and implementation directions include the following:

- further integration of the iterative repair and systematic search engines, to best exploit the advantages of each in a broader range of circumstances
- investigation of modelling and performance tradeoffs in mapping mission requirements into representational elements for the scheduler engines
- incorporating additional optimization criteria into the systematic search algorithm, to provide users with more insight into potential schedule modifications

6. ACKNOWLEDGEMENTS

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

REFERENCES

1. Kan, E.J., J. Rosas, and Q. Vu, *Operations Mission Planner - 26M User Guide Modified 1.0*. 1996, JPL Technical Document D-10092.

2. Bell, C., *Scheduling Deep Space Network Data Transmissions: A Lagrangian Relaxation Approach*. 1992, JPL Technical Report.

3. Chien, S.A., et al. *A hierarchical architecture for resource allocation, plan execution, and revision for operation of a network of communications antennas*. in *Proceedings IEEE International Conference on Robotics and Automation*. 1997. Albuquerque, NM.

4. Borden, C., Y. Yang, and G. Fox. *Planning and Scheduling User Services for NASA's Deep Space Network*. in *1997 International Conference on Planning and Scheduling for Space Exploration and Science*. 1997.

5. Werntz, D., S. Loyola, and S. Zendejas. *FASTER - A tool for DSN forecasting and scheduling*. in *Proceedings of 9th AIAA Computing in Aerospace Conference*. 1993. San Diego, CA.

6. Barbulescu, L., et al., *Scheduling Space-Ground Communications for the Air Force Satellite Control Network.* Journal of Scheduling, 2004. **7**(1): p. 7-34.

7. Barbulescu, L., L.D. Whitley, and A.E. Howe. *Leap Before You Look: An Effective Strategy in an Oversubscribed Scheduling Problem*. in *AAAI 2004*. 2004.

8. Frank, J., et al. *Planning and Scheduling for Fleets of Earth Observing Satellites*. in *I-SAIRAS 2001*. 2001.

9. Beck, J.C., et al. *Texture-Based Heuristics for Scheduling Revisited*. in *AAAI 1997*. 1997.

10. Chien, S., et al., *ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling*, in *SpaceOps 2000*. 2000: Toulouse, France.

11. Clement, B.J. and M.D. Johnston. *The Deep Space Network Scheduling Problem*. in *IAAI*. 2005. Pittsburgh, PA: AAAI Press.

Using Multiagent Teams to Improve the Training of Incident Commanders*

Nathan Schurr, Pratik Patil, Fred Pighin, Milind Tambe,
University of Southern California, Los Angeles, CA 90089, {schurr, pratiksp, pighin, tambe}@usc.edu

ABSTRACT

The DEFACTO system is a multiagent based tool for training incident commanders for large scale disasters. In this paper, we highlight some of the lessons that we have learned from our interaction with the Los Angeles Fire Department (LAFD) and how they have affected the way that we continued the design of our training system. These lessons were gleaned from LAFD feedback and initial training exercises and they include: system design, visualization, improving trainee situational awareness, adjusting training level of difficulty and situation scale. We have taken these lessons and used them to improve the DEFACTO system's training capabilities. We have conducted initial training exercises to illustrate the utility of the system in terms of providing useful feedback to the trainee.

1. INTRODUCTION

The recent hurricanes that have hit the gulf coast of the US have served to reaffirm the need for emergency response agencies to be better prepared for large scale disasters. Both natural and man-made (terrorism) disasters are growing in scale, however the response to these incidents continues to be managed by a single person, namely the incident commander. The incident commander must monitor and direct the entire event while maintaining complete responsibility. Because of this, incident commanders must start to be trained to handle these large scale events and assist in the coordination of the responding team.

In order to fulfill this need and leverage the advantages of multiagents, we have continued to develop the DEFACTO system (Demonstrating Effective Flexible Agent Coordination of Teams via Omnipresence). DEFACTO is a multiagent based tool for training incident commanders for large scale disasters (man-made or natural).

Our system combines a high fidelity simulator, a redesigned human interface, and a multiagent team driving all of the behaviors. Training incident commanders provides a dynamic scenario in which decisions must be made correctly and quickly because human safety is at risk. When using DEFACTO, incident commanders have the opportunity to see the disaster in simulation and the coordination and resource constraints unfold so that they can be better prepared when commanding over an actual disaster. Applying DEFACTO to disaster response aims to benefit the training of incident commanders in the fire department.

With DEFACTO, our objective is to both enable the human to have a clear idea of the team's state and improve agent-human team performance. We want DEFACTO agent-human teams to better prepare firefighters for current human-only teams. We believe that by leveraging multiagents, DEFACTO will result in better disaster response methods and better incident commanders.

Previously, we have discussed building our initial prototype system, DEFACTO [8]. Recently, the Los Angeles Fire Department (LAFD) have begun to evaluate the DEFACTO system. In this paper, we highlight some of the lessons that we have learned from our interaction with the LAFD and how they have affected the way that we continued to design of our training system. These lessons were gleaned from LAFD feedback and initial training exercises.

The lessons learned from the feedback from the LAFD include: system design, visualization, improving trainee situational awareness, adjusting training level of difficulty and situation scale. We have taken these lessons and used them to improve the DEFACTO system's training capabilities.

We have also performed initial training exercise experiments to illustrate the utility of the system in terms of providing useful feedback to the trainee. We ended up finding that allowing more fire engines to be at the disposal of the incident commander sometimes not only didn't improve, but rather worsened team performance. There were even some instances in which the agent team would have performed better had the team never listened to human advice at all. We also provide analysis of such behaviors, thereby illustrating the utility of DEFACTO resulting from the feedback given to trainees.

2. MOTIVATION

In this section, we will first start with an explanation of the current methods for training that the LAFD currently use. Then we explain some of the advantages that our multiagent approach has over these methods.

The incident commander's main duties during a fire shoulder all responsibility for the safety of the firefighters. In order to do this, the incident commander must have constant contact with the firefighters and have a complete picture of the entire situation. The incident commander must make certain that dangerous choices are

*This research was supported by the United States Department of Homeland Security through the Center for Risk and Economic Analysis of Terrorism Events (CREATE) under grant number N00014-05-0630. However, any opinions, findings, and conclusions or recommendations in this document are those of the authors and do not necessarily reflect views of the United States Department of Homeland Security.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.



(a) Current Incident Commander Training Exercise

(b) Fire Captain Roemer using the DEFACTO training system

Figure 1: Old vs. New training methods

avoided and the firefighters are informed and directed as needed.

We were allowed to observe a Command Post Exercise that simulated the place where the incident commander is stationed during a fire (see Figure 1(a)). The Incident commander has an assistant by his side who keeps track on a large sheet of paper where all of the resources (personnel and equipment) are located. A sketch of the fire is also made on this sheet, and the fire and fire engines' location is also managed.

The Command Post is currently simulated by projecting a single static image of a fire in an apartment. In the back of the room, several firefighters are taken off duty in order to play the role of firefighters on the scene. They each communicate on separate channels over walkie talkies in order to coordinate by sharing information and accepting orders. The fire spreading is simulated solely by having one of the off-duty firefighters in the back speaking over the walkie talkie and describing the fire spreading.

The LAFD's current approach, however, has several limitations. First, it requires a number of officers to be taken off duty, which decreases the number of resources available to the city for a disaster during training. Second, the disaster conditions created are not accurate in the way that they appear or progress. Since the image that the incident commander is seeing is static, there is no information about state or conditions of the fire that can be ascertained from watching it, which is contrary to the actual scene of a disaster response. Furthermore, the fire's behavior is determined by the reports of the acting fire fighters over the walkie talkie, which at times might not be a plausible progression of fire in reality. Third, this method of training restricts it to a smaller scale of fire because of the limited personnel and rigid fire representation.

Our system aims to enhance the training of the incident commanders (see Figure 1(b)). Our approach allows for training to not be so personnel heavy, because fire fighter actors will be replaced by agents. By doing this we can start to train incident commanders with a larger team. Through our simulation, we can also start to simulate larger events in order to push the greater number of available resources to their limit. Also, by simulating the fire progression, we can place the Incident commander in a more realistic situation and force them to react to realistic challenges that arise.

3. SYSTEM ARCHITECTURE

In this section, we will describe the technologies used in three major components of DEFACTO: the Omni-Viewer, proxy-based

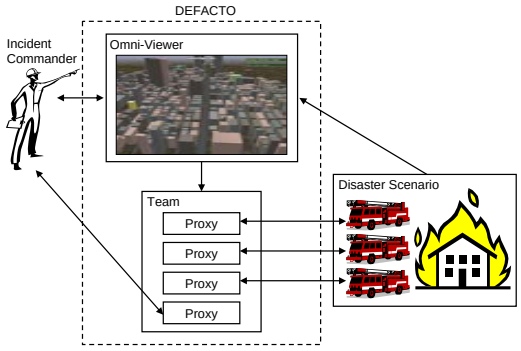


Figure 2: System Architecture

team coordination, and proxy-based adjustable autonomy. The Omni-Viewer is an advanced human interface for interacting with an agent-assisted response effort. The Omni-Viewer has been introduced before [8], however has since been redesigned after incorporating lessons learned from the LAFD. The Omni-Viewer now provides for both global and local views of an unfolding situation, allowing a human decision-maker to obtain precisely the information required for a particular decision. A team of completely distributed proxies, where each proxy encapsulates advanced coordination reasoning based on the theory of teamwork, controls and coordinates agents in a simulated environment. The use of the proxy-based team brings realistic coordination complexity to the training system and allows a more realistic assessment of the interactions between humans and agent-assisted response. These same proxies also enable us to implement the adjustable autonomy necessary to balance the decisions of the agents and human. This architecture has been described in a more extended fashion in [8]; we present a brief report here.

DEFACTO operates in a disaster response simulation environment. The simulation environment itself is provided by the RoboCup Rescue Simulator [3]. To interface with DEFACTO, each fire engine is controlled by a proxy in order to handle the coordination and execution of adjustable autonomy strategies. Consequently, the proxies can try to allocate fire engines to fires in a distributed manner, but can also transfer control to the more expert user (incident

commander). The user can then use the Omni-Viewer to allocate engines to the fires that he has control over. In our scenario, several buildings are initially on fire, and these fires spread to adjacent buildings if they are not quickly contained. The goal is to have a human interact with the team of fire engines in order to save the greatest number of buildings. Our overall system architecture applied to disaster response can be seen in Figure 2.

3.1 Omni-Viewer

Our goal of allowing fluid human interaction with agents requires a visualization system that provides the human with a global view of agent activity as well as shows the local view of a particular agent when needed. Hence, we have developed an omnipresent viewer, or Omni-Viewer, which will allow the human user diverse interaction with remote agent teams. While a global view is obtainable from a two-dimensional map, a local perspective is best obtained from a 3D viewer, since the 3D view incorporates the perspective and occlusion effects generated by a particular viewpoint.

To address our discrepant goals, the Omni-Viewer allows for both a conventional map-like top down 2D view and a detailed 3D viewer. The viewer shows the global overview as events are progressing and provides a list of tasks that the agents have transferred to the human, but also provides the freedom to move to desired locations and views. In particular, the user can drop to the virtual ground level, thereby obtaining the perspective (local view) of a particular agent. At this level, the user can fly freely around the scene, observing the local logistics involved as various entities are performing their duties. This can be helpful in evaluating the physical ground circumstances and altering the team's behavior accordingly. It also allows the user to feel immersed in the scene where various factors (psychological, etc.) may come into effect.

3.2 Proxy: Team Coordination

A key hypothesis in this work is that intelligent distributed agents will be a key element of a disaster response. Taking advantage of emerging robust, high bandwidth communication infrastructure, we believe that a critical role of these intelligent agents will be to manage coordination between all members of the response team. Specifically, we are using coordination algorithms inspired by theories of teamwork to manage the distributed response [6]. The general coordination algorithms are encapsulated in *proxies*, with each team member having its own proxy which represents it in the team. The current version of the proxies is called *Machinetta* [7] and extends the earlier Teamcore proxies [5]. Machinetta is implemented in Java and is freely available on the web. Notice that the concept of a reusable proxy differs from many other "multiagent toolkits" in that it provides the coordination *algorithms*, e.g., algorithms for allocating tasks, as opposed to the *infrastructure*, e.g., APIs for reliable communication. These proxies and their architecture have been discussed in detail in [8].

3.3 Proxy: Adjustable Autonomy

One key aspect of the proxy-based coordination is "adjustable autonomy." Adjustable autonomy refers to an agent's ability to dynamically change its own autonomy, possibly to transfer control over a decision to a human. Previous work on adjustable autonomy could be categorized as either involving a single person interacting with a single agent (the agent itself may interact with others) or a single person directly interacting with a team. In the single-agent single-human category, the concept of flexible transfer-of-control strategy has shown promise [6]. A transfer-of-control strategy is a preplanned sequence of actions to transfer control over a decision among multiple entities. For example, an AH_1H_2 strategy implies that an agent (A) attempts a decision and if the agent fails in the decision then the control over the decision is passed to a human H_1, and then if H_1 cannot reach a decision, then the control is passed to H_2. Since previous work focused on single-agent single-human interaction, strategies were individual agent strategies where only a single agent acted at a time.

An optimal transfer-of-control strategy optimally balances the risks of not getting a high quality decision against the risk of costs incurred due to a delay in getting that decision. Flexibility in such strategies implies that an agent dynamically chooses the one that is optimal, based on the situation, among multiple such strategies (H_1A, AH_1, AH_1A, etc.) rather than always rigidly choosing one strategy. The notion of flexible strategies, however, has not been applied in the context of humans interacting with agent-teams. Thus, a key question is whether such flexible transfer of control strategies are relevant in agent-teams, particularly in a large-scale application such as ours.

DEFACTO has introduced the notion of team-level adjustable autonomy strategies. For example, rather than transferring control from a human to a single agent, a team-level strategy could transfer control from a human to an agent-team. Concretely, each proxy is provided with all strategy options; the key is to select the right strategy given the situation. An example of a team level strategy would combine A_T Strategy and H Strategy in order to make A_TH Strategy. The default team strategy, A_T, keeps control over a decision with the agent team for the entire duration of the decision. The H strategy always immediately transfers control to the human. A_TH strategy is the conjunction of team level A_T strategy with H strategy. This strategy aims to significantly reduce the burden on the user by allowing the decision to first pass through all agents before finally going to the user, if the agent team fails to reach a decision.

4. LESSONS LEARNED FROM INITIAL DEPLOYMENT FEEDBACK

Through our communication with strategic training division of the LAFD (see Figure 1(b)), we have learned a lot of lessons that have influenced the continuing development of our system.

4.1 Adjustable Autonomy in Practice

Our most important lesson learned from talking with the LAFD and seeing their exercises is that adjustable autonomy correctly maps over to what happens in the actual disaster response. The adjusting of autonomy is easily seen as the event scales up and down in size and intensity. For a smaller scale response to, for example, a residential single story house fire, the incident commander will usually make all allocation decisions and thus practice the A strategy. For a larger scale event, a lot of the burden of most allocations are left to the team and other entities in the hierarchy, while the Incident Commander is left to concentrate on the bigger picture. In this case, the Incident commander is notified if a specifically problematic situation, for example not enough resources to attack a particular fire. This strategy is essentially what we refer to as A_TH in our experiments, in which, the team first tries to assign someone to the fire with the resources they have, and if not able to then pass it off to the Incident Commander for help. As the situation were to die down and the size of the team were to decrease, more autonomy would be shifted to the Incident Commander due to an increased ability to make allocations for the team.

It is very helpful to know that these strategies not only are capable of making our agent teams perform well and interface with the Incident Commander, but that they also reflect similar strategies that current firefighting teams are using.



Figure 3: Selecting for closer look at a Fire Engine.

4.2 Questioning the Incident Commander

Another lesson that relates to our agent design is that we learned how a team on the ground may possibly not agree with the command (allocation to a fire) given by the Incident Commander. This will usually be due to the fact that the Incident Commander has a broad global view of the disaster, whereas the agents each have a more detailed local view. This mismatch in information can, at times, lead to detrimental team allocations. In an actual disaster response, this is handled by the allocated team both questioning the order and providing the Incident Commander with the missing information.

This has led us to consider a team of agents that can disagree with human inputs. This issue has not been addressed in our implementation as of yet, but it is relevant given the results that will be presented later in the training exercise experiments. There are experimental settings in which the team performance would have been improved, had they rejected the Incident Commander's input.

4.3 Perspective

Just as in multiagent systems, the Incident commander must overcome the challenge of managing a team that each possess only a partial local view. This is highlighted in fighting a fire by incident commanders keeping in mind that there are five views to every fire (4 sides and the top). Only by taking into account what is happening on all five sides of the fire, can the fire company make an effective decision on how many people to send where. Because of this, a local view (see Figure 4(a)) can augment the global view (see Figure 4(b)) becomes helpful in determining the local perspectives of team members. For example, by taking the perspective of a fire company in the back of the building, the incident commander can be aware that they might not see the smoke from the second floor, which is only visible from the front of the building. The incident commander can then make a decision to communicate that to the fire company or make an allocation accordingly.

The 3D perspective of the Omni-Viewer was initially thought to be an example of a futuristic vision of the actual view given to the incident commander. But after allowing the fire fighters to look at the display, they remarked, that they have such views available to them already, especially in large scale fires (the very fires we are trying to simulate). At the scene of these fires are often a news helicopter is at the scene and the incident commander can patch into the feed and display it at his command post. Consequently our training simulation can already start to prepare the Incident Commander to incorporate a diverse array of information sources.

4.4 Fire Behavior

We also learned how important smoke and fire behavior is to the firefighters in order to affect their decisions. Upon our first showing of initial prototypes to the Incident Commanders, they looked at our simulation, with flames swirling up out of the roof (see Figure 5(a)). We artificially increased fire intensity in order to show off the fire behavior and this hampered their ability to evaluate the situation and allocations. They all agreed that every firefighter should be pulled out because that building is lost and might fall at any minute! In our efforts to put a challenging fire in front of them to fight, we had caused them to walk away from the training. Once we start to add training abilities, such as to watch the fire spread in 3D, we have to also start to be more aware of how to accurately show a fire that the Incident Commander would face. We have consequently altered the smoke and fire behavior (see Figure 5(b)). The smoke appears less "dramatic" to the a lay person than a towering inferno, but it provides a more effective training environment.

4.5 Gradual Training

Initially, we were primarily concerned with changes to the system that allowed for a more accurate simulation of what the Incident Commander would actually see. Alternatively, we have also added features, not because of their accuracy, but also to aid in training by isolating certain tasks. Very often in reality and in our simulations, dense urban areas obscure the ability to see where all of the resources (i.e. fire engines) are and prevent a quick view of the situation (see Figure 6(a)). To this aim, we have added a new mode using the 3D, but having the buildings each have no height, which we refer to as Flat World (see Figure 6(b)). By using this flat view, the trainee is allowed to concentrate on the allocation of resources, without the extra task of developing an accurate world view with obscuring high rise buildings.

4.6 User Intent

A very important lesson that we learned from the LAFD, was that the Incident Commander cannot be given all information for the team and thus the human does not know all about the status of the team members and vice versa. Consequently, this lack of complete awareness of the agent team's intentions can lead to some harmful allocations by the human (Incident Commander). In order for information to be selectively available to the Incident Commander, we have allowed the Incident Commander to query for the status of a particular agent. Figure 3 shows an arrow above the Fire Engine at the center of the screen that has been selected. On the left, the statistics are displayed. The incident commander is able to select a particular fire engine and find out the equipment status, personnel status, and the current tasks that are being performed by the fire fighters aboard that engine. This detailed information can be accessed if desired by the Incident Commander, but is not thrown to the screen by all agents, in order to not overwhelm the Incident Commander.

4.7 Scale

In addition, we have also learned of new challenges that we are currently attempting to tackle by enhancing the system. One of the biggest challenges in order to start simulating a large urban fire is the sheer scale of the resources that must be managed. According to the fire captains, in order to respond to a single high rise building with a few floors on fire, roughly 200 resources (fire engines, paramedics etc.) would need to be managed at the scene. Coordinating such a large number of agents on a team is a challenge. Also, as the incident scales to hundreds of resources, the Incident Commander ends up giving more autonomy to the team or else face



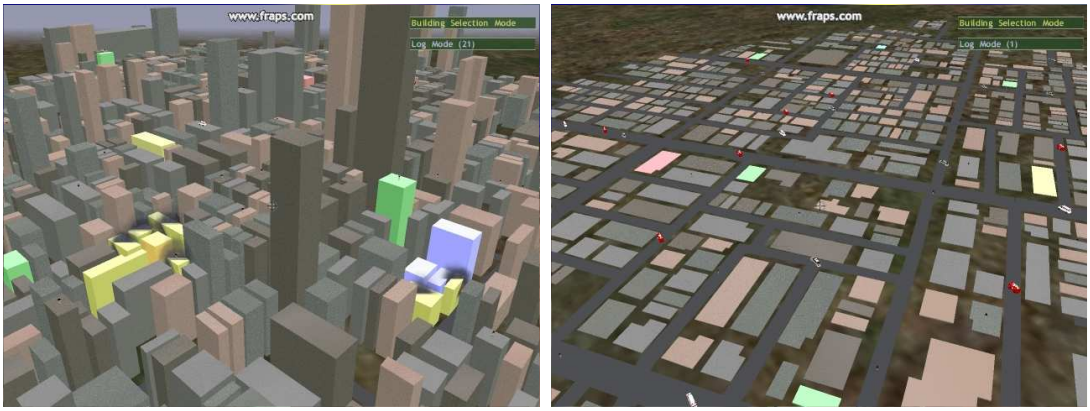
(a) Local Perspective (b) Global Perspective

Figure 4: Local vs. Global Perspectives in the Omni-Viewer



(a) Old Fire (b) New Smoke

Figure 5: Improvement in fire visualization



(a) Normal (b) Flat World

Figure 6: Improvement in locating resources (fire engines and ambulances)

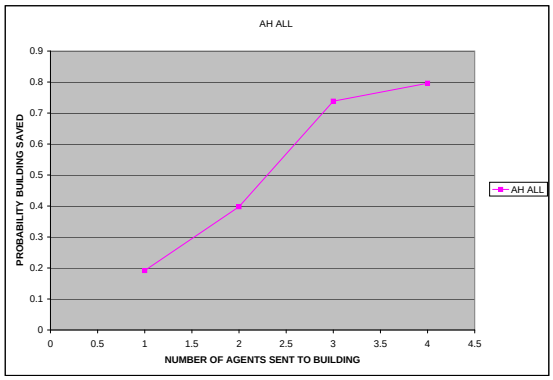


Figure 7: AH for all subjects.

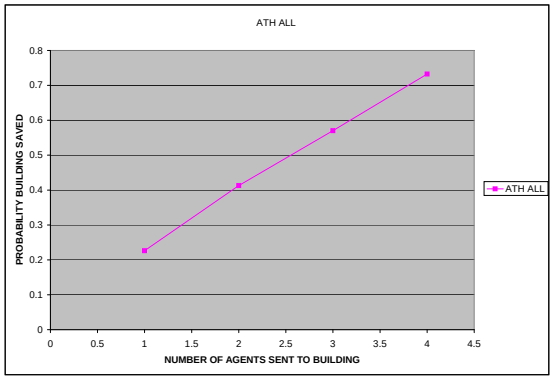


Figure 8: ATH for all subjects.

being overwhelmed. We believe that adjustable autonomy will start to play a bigger and more essential roll in allowing for the Incident Commander to monitor the larger situations.

5. LESSONS LEARNED FROM TRAINING EXERCISES

In this section, we will present results and analysis from a set of training exercises. Our initial experimental results have been published earlier [8], however the analysis presented here is new.

5.1 Training Exercises

In order to study the potential of DEFACTO, we performed some training exercises with volunteers. These initial experiments showed us that humans can both help and hurt the team performance. The key point is that DEFACTO allows such experiments with training exercises and more importantly allows for analysis and feedback regarding the exercises. Thus trainees can gain useful insight as to why their decisions led to problematic/beneficial situations.

The results of our training exercise experiments are shown in Figure 9, which shows the results of subjects 1, 2, and 3. Each subject was confronted with the task of aiding fire engines in saving a city hit by a disaster. For each subject, we tested three strategies, specifically, H, AH (individual agent, then human) and $A_T H$ (agent team, then human); their performance was compared with the completely autonomous A_T strategy. AH is an individual agent strategy, tested for comparison with $A_T H$, where agents act individually, and pass those tasks to a human user that they cannot immediately perform. Each experiment was conducted with the same initial locations of fires and building damage. For each strategy we tested, varied the number of fire engines between 4, 6 and 10. Each chart in Figure 9 shows the varying number of fire engines on the x-axis, and the team performance in terms of numbers of building saved on the y-axis. For instance, strategy A_T saves 50 building with 4 agents. Each data point on the graph is an average of three runs. Each run itself took 15 minutes, and each user was required to participate in 27 experiments, which together with 2 hours of getting oriented with the system, equates to about 9 hours of experiments per volunteer.

Figure 9 enables us to conclude the following:

- *Human involvement with agent teams does not necessarily lead to improvement in team performance.* Contrary to expectations and prior results, human involvement does not uniformly improve team performance, as seen by human-involving strategies performing worse than the A_T strategy in some cases. For instance, for subject 3 AH strategy provides higher team performance than A_T for 4 agents, yet at 10 agents human influence is clearly not beneficial.

- *Providing more agents at a human's command does not necessarily improve the agent team performance.* As seen for subject 2 and subject 3, increasing agents from 4 to 6 given AH and $A_T H$ strategies is seen to degrade performance. In contrast, for the A_T strategy, the performance of the fully autonomous agent team continues to improve with additions of agents, thus indicating that the reduction in AH and $A_T H$ performance is due to human involvement. As the number of agents increase to 10, the agent team does recover.

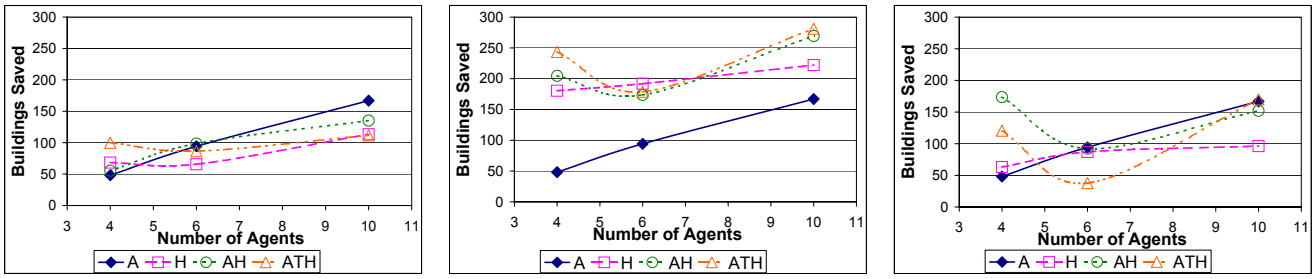
- *Complex team-level strategies are helpful in practice*: $A_T H$ leads to improvement over H with 4 agents for all subjects, although surprising domination of AH over $A_T H$ in some cases indicates that AH may also need a useful strategy to have available in a team setting.

Note that the phenomena described range over multiple users, multiple runs, and multiple strategies. Unfortunately, the strategies including the humans and agents (AH and $A_T H$) for 6 agents show a noticeable decrease in performance for subjects 2 and 3 (see Figure 9). It would be useful to understand which factors contributed to this phenomena from a trainee's perspective.

5.2 Analysis

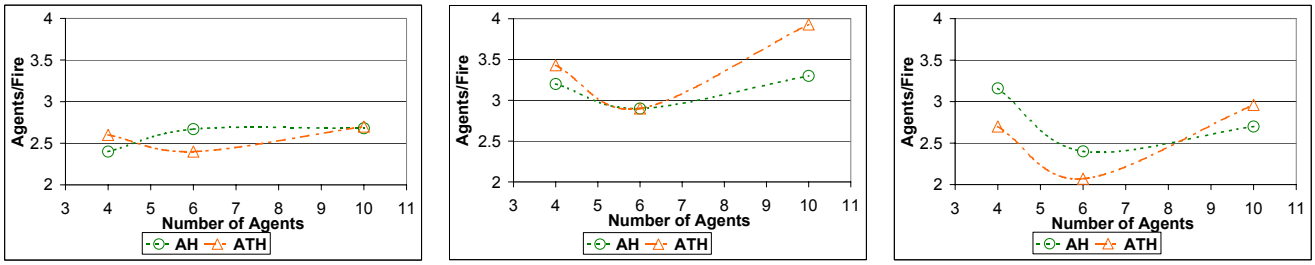
We decided to perform a more in depth analysis of what exactly was causing the degrading performance when 6 agents were at the disposal of the Incident Commander. Figure 10 shows the number agents on the x-axis and the average amount of fire engines allocated to each fire on the y-axis. AH and $A_T H$ for 6 agents result in significantly less average fire engines per task (fire) and therefore lower average. Another interesting thing that we found was that this lower average was not due to the fact that the Incident Commander was overwhelmed and making less decisions (allocations). Figures 11(a), 11(b), and 11(c) all show how the number of buildings attacked do not go down in the case of 6 agents, where poor performance is seen.

Figures 7 and 8 show the number of agents assigned to a building on the x-axis and the probability that the given building would be saved on the y-axis. The correlation between these values demonstrate the correlation between number of agents assigned to the quality of the decision.



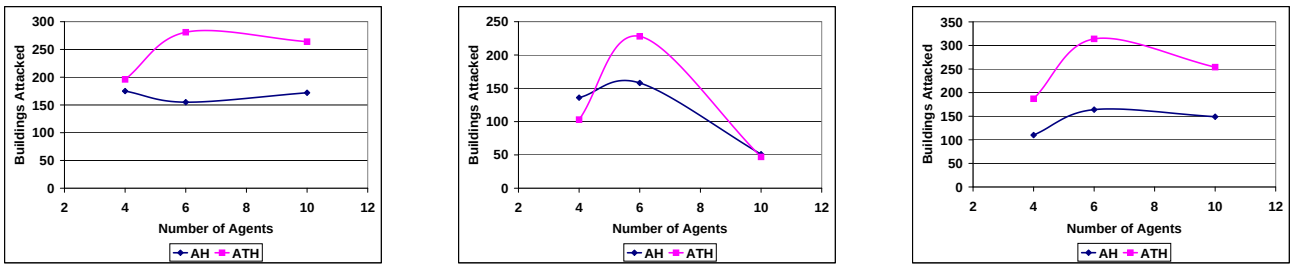
(a) Subject 1 (b) Subject 2 (c) Subject 3

Figure 9: Performance.



(a) Subject 1 (b) Subject 2 (c) Subject 3

Figure 10: Amount of agents assigned per fire.



(a) Subject 1 (b) Subject 2 (c) Subject 3

Figure 11: Number of buildings attacked.

We can conclude from this analysis that the degradation in performance occurred at 6 agents because fire engine teams were split up, leading to fewer fire-engines being allocated per building on average. Indeed, leaving fewer than 3 fire engines per fire leads to a significant reduction in fire extinguishing capability. We can provide such feedback of overall performance, showing the performance reduction at six fire engines, and our analysis to a trainee. The key point here is that DEFACTO is capable of allowing for such exercises, and their analyses, and providing feedback to potential trainees, so they improve their decision making, Thus, in this current set of exercises, trainees can understand that with six fire engines, they had managed to split up existing resources inappropriately.

6. RELATED WORK AND SUMMARY

In terms of related work, it is important to mention products like JCATS [9] and EPICS [4]. JCATS represents a self-contained, high-resolution joint simulation in use for entity-level training in open, urban and subterranean environments. Developed by Lawrence Livermore National Laboratory, JCATS gives users the capability to detail the replication of small group and individual activities during a simulated operation. At this point however, JCATS cannot simulate agents. Finally, EPICS is a computer-based, scenario-driven, high-resolution simulation. It is used by emergency response agencies to train for emergency situations that require multi-echelon and/or inter-agency communication and coordination. Developed by the U.S. Army Training and Doctrine Command Analysis Center, EPICS is also used for exercising communications and command and control procedures at multiple levels. Similar to JCATS however, EPICS does not currently allow agents to participate in the simulation. More recently multiagents have been successfully applied to training navy tactics [10] and teams of Uninhabited Air Vehicles [1, 2]. Our work is similar to these in spirit, however our focus and lessons learned are based on the train of Incident Commanders in disaster rescue environments.

In summary, in order to train Incident Commanders for large scale disasters, we have been working on the DEFACTO training system. This multiagent system tool has begun to be used by fire captains from the Los Angeles Fire Department. We have learned some valuable lessons from their feedback and the analysis of some initial training exercise experiments. These lessons were gleaned from LAFD feedback and initial training exercises. The lessons learned from the feedback from the LAFD include: system design, visualization, improving trainee situational awareness, adjusting training level of difficulty and situation scale. We have taken these lessons and used them to improve the DEFACTO system's training abilities. We have conducted initial training exercises to illustrate the utility of the system in terms of providing useful feedback to the trainee. Through DEFACTO, we hope to improve training tools for and consequently improve the preparedness of Incident Commanders.

7. ACKNOWLEDGMENTS

Thanks to CREATE center for their support. Also, thanks to Fire Captains of the LAFD: Ronald Roemer, David Perez, and Roland Sprewell for their time and invaluable input to this project.

8. REFERENCES

[1] J. W. Baxter and G. S. Horn. Controlling teams of uninhabited air vehicles. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS)*, 2005.

[2] S. Karim and C. Heinze. Experiences with the design and implementation of an agent-based autonomous uav controller. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS)*, 2005.

[3] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjoh, and S. Shimada. Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *IEEE SMC*, volume VI, pages 739–743, Tokyo, October 1999.

[4] L. L. N. Laboratory. Jcats - joint conflict and tactical simulation. In *http://www.jfcom.mil/about/fact_jcats.htm*, 2005.

[5] D. V. Pynadath and M. Tambe. Automated teamwork among heterogeneous software agents and humans. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, 7:71–100, 2003.

[6] P. Scerri, D. Pynadath, and M. Tambe. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 17:171–228, 2002.

[7] P. Scerri, D. V. Pynadath, L. Johnson, P. Rosenbloom, N. Schurr, M. Si, and M. Tambe. A prototype infrastructure for distributed robot-agent-person teams. In *AAMAS*, 2003.

[8] N. Schurr, J. Marecki, P. Scerri, J. P. Lewis, and M. Tambe. The defacto system: Training tool for incident commanders. In *The Seventeenth Innovative Applications of Artificial Intelligence Conference (IAAI)*, 2005.

[9] A. S. Technology. Epics - emergency preparedness incident commander simulation. In *http://epics.astcorp.com*, 2005.

[10] W. A. van Doesburg, A. Heuvelink, and E. L. van den Broek. Tacop: A cognitive agent for a naval training simulation environment. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS)*, 2005.

Autonomous Agents for Air-Traffic Deconfliction

Michal Pěchouček, David Šišlák, Dušan Pavlíček and Miroslav Uller

Gerstner Laboratory – Agent Technology Group
Department of Cybernetics, Czech Technical University
Technická 2, Prague, 166 27, Czech Republic

pechouc@labe.felk.cvut.cz

ABSTRACT

This contribution presents a deployment exercise of multi-agent technology in the domain of deconflicted air-traffic control among several autonomous aerial vehicles (manned as well as unmanned). Negotiation based deconfliction algorithm have been developed and integrated in the agent-based model of the individual flight. Operation of the underlying multi-agent system has been integrated with freely available, geographical and tactical data sources in order to demonstrate openness of the technology. An additional, web client visualization and access component has been developed in order to facilitate a multi-user, platform independent use of the system. The features and application design is illustrated in the demonstration video clip[1].

1. INTRODUCTION

In the future warfare and humanitarian relief operations (especially in the surveillance and monitoring domains) there will be a strong potential for integration of the technologies and mechanisms supporting coordinated flight among the collective of autonomous manned and unmanned aerial vehicles. In this paper we argue that agent technology, relying on collective decision making among several autonomous intelligent agents, can provide solution for such a challenge. Even though this is not a fundamental research paper, this contribution provides good evidence of practical applicability of agent technology by reporting on a technology deployment exercise in this specific domain.

According to [11] the current air traffic control methods based on rigidly structured airspace have shown to be inefficient even for the future coordination of the piloted aircraft. This is true mainly due to: (i) inefficiency of airspace utilisation that is based on fixed predefined flight corridors, (ii) increased air traffic workload given by ever increasing air traffic density and (iii) use of obsolete technologies, that are in many cases 30 years old.

[1] `http://agents.felk.cvut.cz/atg-videos/atc_video_divx_titles.avi`

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

In the future these will be important for both UAVs and manned aircraft:

- requirements to increase flexibility of the vehicles operations,
- requirements to fly higher number of vehicles in substantially smaller airspace and
- requirements for higher precision of the flight plan and its accomplishment.

Due to the first two reasons of inefficiency listed above, the classical air traffic control methods are not very suitable for coordination of higher numbers of dynamically tasked aerial vehicles. In our work, we have built on top of the concept of *free flight* [9], [4] – an approach to autonomous routing of the aircraft and continuous self-optimization of the flight trajectory complemented by the peer-to-peer deconfliction mechanism.

The presented work resulted in a demonstration prototype (ATC) of a multi-agent system modeling deconflicted free-flight of collective of fully autonomous aircraft.

ATC has not been primarily designed as flight planner producing a set of deconflicted route plans for a number of aerial vehicles. There is a central planning component that provides (during the *planning phase*) the flight plans without consideration of possible operation of other aircraft. This planning process can be easily distributed among the individual aircraft and thus no central planner would be necessary. The plans are likely to contain possible collisions. Working with no-collisions makes planning considerably faster and much more flexible. This also results in substantial scalability improvement affording to operate higher number of aerial vehicles in condensed airspace (can be used e.g. in situation where a higher number of unmanned aerial vehicles carries out rapid surveillance tasks).

Following the planning phase, ATC performs the *simulation phase*, where the aircraft implement the provided flight plans in a free-flight fashion. In a danger of possible collisions, the embedded deconfliction mechanism makes sure that aircraft negotiate deconfliction maneuvers, while still optimizing their operation.

2. DECONFLICTION BY NEGOTIATION

ATC is designed to operate in distributed manner, which is why the deconfliction technology (while developed within the multi-agent ATC model) is ready for deployment on autonomous vehicles without any central point of control.

The aircraft are modeled by agent containers hosting several agents. In this contribution we will be referring to

agents representing an aircraft auto-pilot. This agent is a self-interested entity that is in charge of (i) preparing a detailed flight plan for the airplane respecting time-specific waypoints for the airplane's mission and (ii) executing the flight plan by performing the flight.

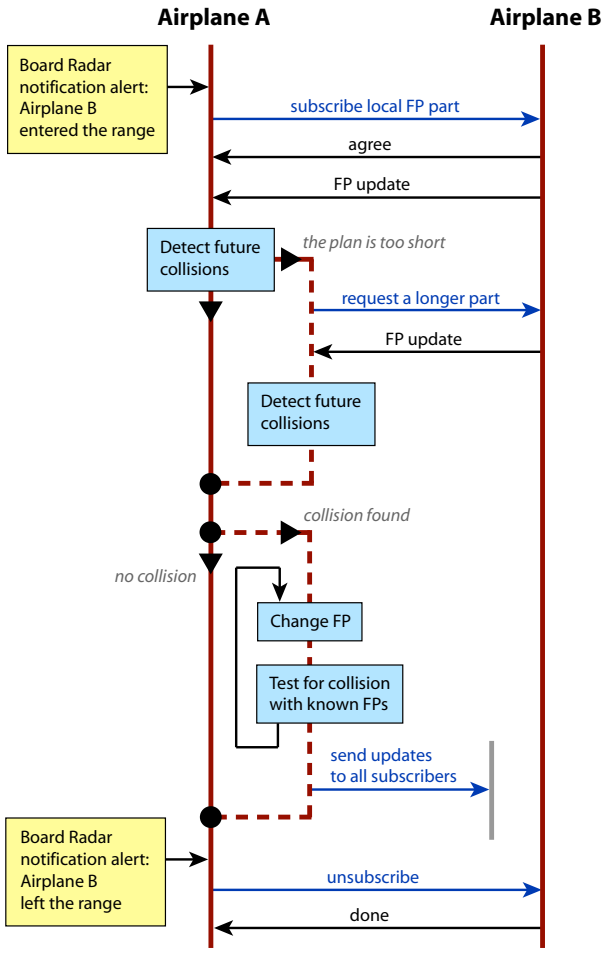


Figure 1: Negotiation protocol

Each simulated aircraft is surrounded by a number of concentric spherical zones: **Communication**, **Alert**, **Safety** and **Collision** zones. The **communication zone** is the outermost one. It represents the communication range of the data transmitter onboard the aircraft. Using this data, the aircraft can send data packets to other aircraft that are positioned within the specified spherical zone defined by its radius. The **alert zone** defines the operation range of the radar onboard the aircraft. If another aircraft is located within the alert zone, the aircraft are periodically notified about its relative position and its flight code. The **safety zone** encapsulates the area around an aircraft that other aircraft are not allowed to enter in order to minimize the mutual influence of the aircraft movements, e.g. airspace turbulence. If two aircraft do enter each other's safety range, they can still continue flying but their flight path may be influenced by e.g. turbulence. This is not the case when two or more airplanes fly together in a close formation. The **collision zone** is the innermost zone. It defines the critical contact area. When the mutual distance between two aircraft is smaller than the sum of their collision zone radiuses, the physical collision happens.

Cooperative deconfliction. In the simulation phase the ATC system solves collisions *cooperatively* by negotiation between the aircraft, see Figure 1. Let us suppose an aircraft `A` to fly along its planned optimal flight path through its mission waypoints. An airplane `B` enters the *alert zone* of the airplane `A`. The pilot agent of the aircraft `A` is notified about its position and flight code by the on board radar system. The pilot agent of the aircraft `A` tries to establish negotiation connection with the pilot agent of `B`. In the case when the connection cannot be established or the communication is not trusted, the pilot agents should use *non-cooperative approach*, described later in this section. If the connection was established successfully, the pilot agent `A` subscribes for a local area flight plan of the aircraft `B` (representation of the flight plan is described in Section 5.1). The pilot agent of aircraft `B` sends an update to the subscriber every time it changes its own flight plan. The update contains the part of the flight plan of the aircraft for the specified amount of time depending on the flight speed. The update is also sent when the time span of the previous update was not long enough and it needs to be updated again. When the pilot agent `A` receives an update from the pilot agent `B`, it executes the **collision detection process** on its own flight plan and the received one.

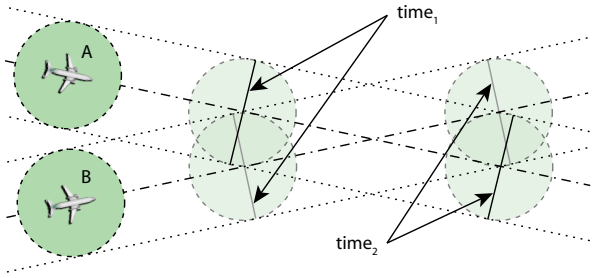


Figure 2: Flight plan collision interval

The collision detection process is a linear algorithm that analyses two flight paths and tests the condition of safe flights. If there is a specific point in time detected when the distance between the positions of the aircraft is smaller than the maximum of the safety ranges of the aircraft (see Figure 2), the detection process returns $time_1$ and $time_2$ which represent the first and the last collision point between the two flight plans. This information is needed by the pilot agent for better handling of the situation.

If the detection test is negative, both flight paths are safe for flying. If a collision is found, the airplanes `A` and `B` must modify their flight paths. The first prototype of the designed system uses a rule-based approach for modifying flight plans described in Section 2.1.

Non-cooperative deconfliction: The distributed deconfliction approach is open to be extended to *non-cooperative deconfliction*. The non-cooperative deconfliction is useful in situations when an airplane has a malfunctioning transmitter/receiver on its board or in the situation when there is an intruder/enemy airplane with adversarial behavior [7]

which intentionally sends incorrect future flight path parts to the others. The most suitable approach to the *non-cooperative deconfliction* is the game theory. In this case the pilot agent tries to change its own flight plan in a way that would guarantee a minimal collision risk for any future position of the other airplane. To determine all possible future positions of the other plane, information about its current position, direction and information about its type can be used. The monitored object's flight path is always continuous but there are also certain restrictions that depend on the airplane type – e.g. minimal/maximal flight speed, minimal radius of turning, etc. When the pilot agent wants to identify whether or not it should use the non-cooperative deconfliction for a particular airplane, it can integrate a special *detection module*. The detection module compares the notification received from the board radar with the known flight plan part of the aircraft in the radar range.

2.1 Collision Avoidance Mechanism

The specific collision type is determined based on the angle between the direction vectors of the concerned planes at $time_1$ projected to the ground plane (defined by X and Y axes), see Figure 3.

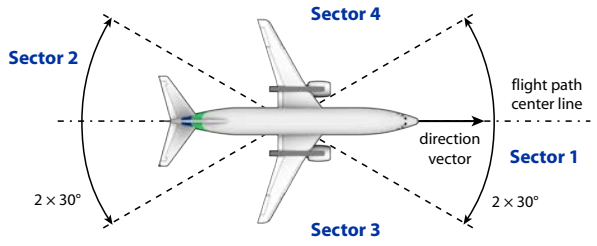


Figure 3: Identification of the collision type

Depending on the computed angle, the airplane `B` falls into one of four sectors surrounding the airplane `A`. Depending on that sector, one of the following rules is applied on the flight plan of the airplane `A` to avoid the collision:

- **Sector 1** – head-on collision, in this case the airplanes avoid each other by both of them turning to the right. The flight plan is changed as shown in Figure 4. The pilot agent shifts the plan points at $time_1$ and $time_2$ to the right, perpendicularly to the old direction vector. The length of the shift is equal to a minimum of safety ranges of both airplanes. Beyond $time_2$, the flight plan follows the shortest way to the next mission waypoint.
- **Sector 2** – rear collision, there are two subcases: i) the front aircraft is faster – airplanes do not change their current flight plans; ii) the rear airplane is faster – it has to change its flight plan so that it turns to the right and passes the front airplane without endangering it. The flight plan is similar to that in Figure 4. To achieve this, the rear airplane shifts its flight plan points at $time_1$ and $time_2$ to the right, perpendicularly to the old direction vector. The length of the shift is at least 1.1 times of the safety range.
- **Sector 3** – side collision, the airplane `B` has higher traffic priority. The aircraft `A` needs to slow down its speed so that it reaches the collision point at $time_1$ later than the airplane `B`. If this is not possible due to the minimal flight speed defined for each airplane type, the airplane `A` slows down as much as possible and shifts its flight plan point at $time_1$ to the right so that there is no collision between the two flight plans.
- **Sector 4** – side collision, the airplane `B` has lower traffic priority. The aircraft `A` changes its flight plan by increasing its flight speed so that it passes the collision point before the airplane `B`. The airplane `A` only accelerates as much as needed.

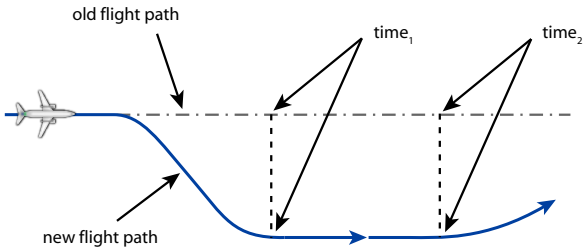


Figure 4: Change of the flight plan.

The above rule-based changes to the flight plan are carried out by both planes independently because each aircraft detects the possible collision with the other airplane from its own point of view. After applying the changes to the flight plan, the airplane sends an updated local flight plan part to all airplanes that subscribed for it. The change is also verified against all other known flight plans of all aircraft monitored by the board radar system. If another collision is detected, new changes are applied.

The pilot agent internally uses the *flight plan wrapper* interface for manipulation with its flight plan. The requests for each plan modification are handled as a special set of solver time-constrained waypoints. A special handling algorithm takes care of the execution of each modification that overrides the previous one.

3. A-GLOBE: MULTI-AGENT PLATFORM

The agent-based part of the designed prototype of the Air Traffic Control (ATC) system runs on the **$\mathcal{A}$-globe** Java multi-agent platform [10, 1]. **$\mathcal{A}$-globe** is a fast and lightweight platform with agent ***mobility*** and ***inaccessibility*** support. Besides the functions common to most of agent platforms it provides ***Geographical Information System***-like service to the user. Therefore, the platform is ideally suited for testing experimental scenarios featuring agents' position, position dependent environment simulation and communication inaccessibility. The platform provides support for permanent and mobile agents, such as communication infrastructure, storage, directory services, agent migration (including library migration with version handling), service deployment, etc.

$\mathcal{A}$-globe is optimized to consume just a limited amount of resources. **$\mathcal{A}$-globe** platform is not fully compliant with the FIPA [3] specifications, still it implements most protocols and respects the spirit of the specification. It does not support communication between different agent platforms (e.g. with JADE, JACK, etc.). For large scale scenarios, the problems with system performance that interoperability

brings (memory requirements, communication speed) outweigh any advantages, as heterogenous environment is of limited interest for simulations.

The $\mathcal{A}$-**globe** operation is based on several core components: The **Agent Platform** provides the basic components necessary to run one or more *agent containers*, the *container manager* and the *library manager*. The **Agent Container** is a skeleton entity that provides basic functions such as communication, storage and management for agents. The **services** provide common functions for all agents in a container. The agents have two means of communication with a service – either via standard messages or by using a service shell – a special proxy object that interfaces service function to a client so that they appear to be synchronous function calls, but the calls are actually handled in the service thread thereby preventing deadlocks and synchronization overhead.

The $\mathcal{A}$-**globe** platform is primarily aimed at large scale, real world simulations with fully fledged agents. To support this goal, it includes a special infrastructure for environmental simulation. *Actor agents* play roles in the simulated world, while *Environment Simulation (ES) agents* implement the simulated world itself. ES agents only rarely use messages to communicate with actor agents. Instead, they communicate via *topic messaging*.

Topic messaging implements container to container messaging reserved for easy environmental simulation. Topic messaging is built on top of standard messages and is managed by the Geographic Information System (GIS) Services – server and clients. GIS services provide distribution and subscription mechanism for the agents. ES agents can be responsible for nearly any simulation layer, depending on the wishes of the developers. Accessibility agent, which controls the availability of communication links between containers holding the actor agents, is one of the most important of ES agents. $\mathcal{A}$-**globe** messaging layers use the information provided by the accessibility agent to prevent sending messages between inaccessible nodes. Accessibility simulated by the system can depend on many factors, typically including the distance and simulated link reliability.

4. PROTOTYPE STRUCTURE OVERVIEW

Air Traffic Control (ATC) system is mainly written in Java except for the real-time visualization component which is written in C++. Agent-based part of the system runs on the $\mathcal{A}$-**globe** Java multi-agent platform. The system consists of several components, see Figure 5:

– **ATC core** is a mandatory component of the system responsible for aircraft simulation and airways planning. The component also provides interfaces for connecting a number of real-time visualization components and remote web clients. See Section 4 for details.

– **Real-time 2D/3D visualizer** is an optional component that provides the user with a real-time overview of the simulation state with all important information in a 3D/2D environment. See Section 7 for details.

– **Remote web client** is an optional component allowing a remote user to connect and interact with the ATC core system via a client application executed from an internet browser. If necessary, all communication data can be secured using asymmetric cryptography but it comes with higher processor load requirements. See Section 7 for details.

Agent-based core system encapsulates one *server component* (described in Section 4.1) and one or more *platform components*, Figure 5. The *platform component* is used as a registration unit for starting the simulated *aircraft container* (described in Section 4.2) inside Java Virtual Machine (JVM). When ATC system is used for planing/simulation of a huge number of aircraft, it is highly recommended to use several host computers with their own JVMs and *platform components*. The number of running aircraft is proportionally split between the registered platform components. This enables the ATC system to balance the overall load between all registered computers.

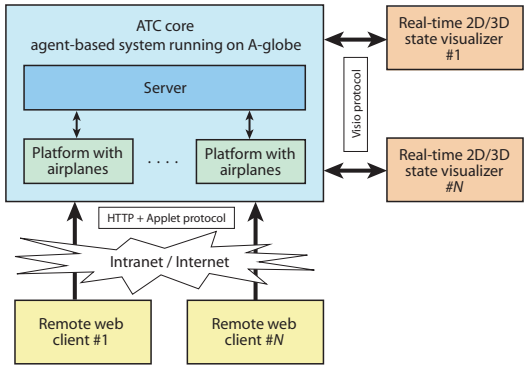


Figure 5: ATC System Structure Overview

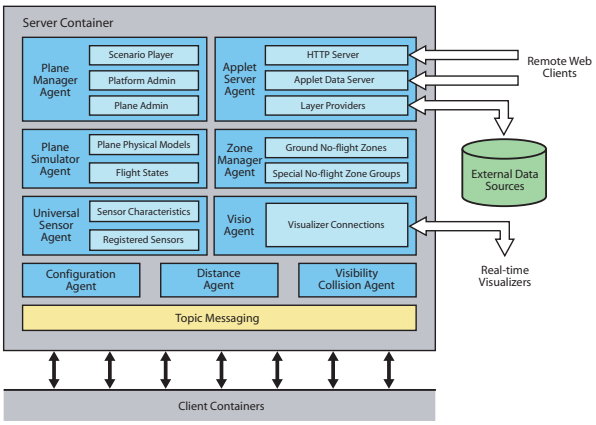


Figure 6: Server component of the ATC Core system

4.1 Server Component

The *server component* of the ATC core system is a sole central element of the system. It simulates positions of aircraft and other objects in the simulated world, aircraft hardware, weather conditions, communication ranges given by the ranges of board data transmitters, etc. It is also responsible for acquiring information about all airplanes and provides them to both types of visualizers. If the proposed distributed agent system for flying on deconflicted airways was used to control real aircraft, this *server component* could be removed from the system.

The *server component*, Figure 6, consists of several agents. **Configurator Agent** loads initial configurations from the specified configuration files and distributes them to other agents. **Plane Manager Agent** administrates connected

JVM platforms and running aircraft containers, it spawns new airplanes and removes the existing ones, and assigns initial flight missions to the airplanes. New airplanes can be started automatically as specified in the scenario script. The server also acts as a load balancer between connected platform components. **Plane Simulator Agent** computes the current position of the aircraft in the simulated world. It contains all physical models for all plane types and keeps all current flight plans and states of the running aircraft. When a plane *pilot agent* changes some part of the flight plan, the change is propagated via the *plane agent* to the *plane simulator agent* in the form of a difference flight plan update. The agent can be asked for the current airplane position by the *pilot agent*. **Distance Agent** calculates Euclidian distances between each pair of existing aircraft using their current positions. **Visibility Collision Agent** prepares **$\mathcal{A}$-globe** visibility updates [10] for controlling communication restrictions between airplanes. It also detects whether there has been a physical collision between flying aircraft. The airplanes that have collided with any other object are uncontrollable and they fall down to the ground. Falling aircraft can endanger any airplane that flies under it. **Universal Sensor Agent** represents all radar sensors on aircraft boards. **Zone Manager Agent** takes care of no-flight zones. It transforms any defined no-flight zones to a compressed octant tree. The ground surface is also represented as a special no-flight zone. This allows to use the planning mechanism even for generating flight plans that do not collide with the ground surface. No-flight zones can be dynamically changed during the planning/simulation. **Visio Agent** is an interface between the Core agent system and the C++ real-time visualizers. To ensure fast communication between them, a special binary communication protocol defined is used. **Applet Server Agent** runs the HTTP server, Applet Data server and all external data providers. It provides a communication interface between the agent system and the *remote web client* (described in Section 7).

All server agents communicate together using **$\mathcal{A}$-globe *topic messaging*** described in [10].

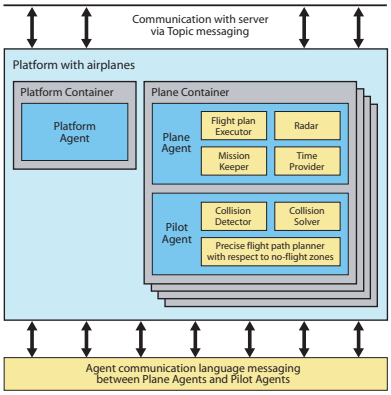


Figure 7: Platform Architecture

4.2 Platform Component

The *platform container* is used as a registration unit for starting containers with agents simulating aircraft behavior. Design of the platform component is shown in the Figure 7. The only agent running in the *platform container* is the *platform agent*. The agent acts as a control bridge between the *plane manager agent* and the local JVM on which it is running. Through this agent the *plane container* can be started or removed. Many *plane containers* can run in each *platform component* (inside one JVM) or there can be no *plane containers* at all.

Each running *plane container* hosts two agents. **Plane Agent** provides high-level airplane functions, such as flight plan execution in cooperation with the *plane simulator agent*, radar and detector readings, airplane configuration and time synchronization ticks. **Pilot Agent** is the main control unit of the simulated aircraft handling negotiation based deconfliction as described in Section 2. It processes notifications about every new visible objects on the radar and tries to communicate with the respective *pilot agent* if there is any. It generates a detailed flight plan based on the initial mission specification given by the *plane manager agent* with respect to the no-flight zones (Section 5), described in Section 5.2.

The airplane container agents communicate with the server container via **$\mathcal{A}$-globe *topic messaging***, while together they communicate using standard agent communication language (ACL) messages.

5. FLIGHT MODELING

The modeling of flights in our system proceeds in two main phases: the *planning* of the flight plans for each airplane and the *simulation* of these flight plans. In this section, we give a brief description of the planner component and other key concepts related to the flight modeling. We use a very simple physical model of the airplanes: they are modeled as mass points moving along a previously planned trajectories.

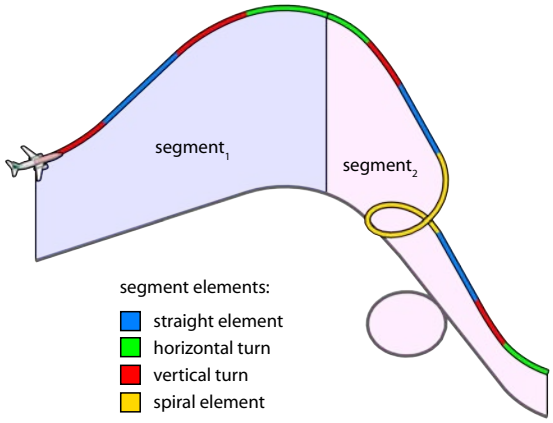


Figure 8: Flight plan, segments, elements

5.1 The Structure of Flight Plan

The flight plan describes a trajectory which the plane follows during its flight. Its principal building blocks are *waypoints*, *segments* and *elements*. A waypoint is an important navigational point used for rough definition of the flight route; it represents a certain location and also specifies the time interval when the airplane should fly through it. The waypoints serve as an input to the planner, which generates the precise flight path consisting of the segments and elements. A segment is a portion of a planned flight path

between two adjacent waypoints. The segments are composed of elements, which constitute the most basic parts of a flight plan with a simple geometry.

Our planner is able to generate flight paths from four types of elements: straight elements, turns (horizontal and vertical), spirals and a warp element. The warp element is a special kind of element allowing for an abrupt change of certain parameters of the flight route (e.g. position, direction or velocity). Without warp elements, the flight trajectory is always continuous and smooth and each element is connected smoothly to the adjacent ones. This is desirable in most cases; exceptions to this rule will be mentioned later. An example of a flight plan is shown in Figure 8.

5.2 Flight Path Planning

By the flight path planning we mean creating a smooth and continuous path passing through all input waypoints, which will also satisfy the time constraints associated to the waypoints. The planning proceeds in three phases: the computation of the actual path without regard to the time constraints, the adjustment of the flight plan to satisfy the time criteria and the replanning of the plan in the parts where it collides with no-flight zones (if defined).

During the **Path Planning**, the planner generates a detailed flight path which passes through all the waypoints. For each couple of successive waypoints, a segment is created, initially without any elements. A segment represents the smallest part of a flight plan which can be planned independently on the other parts of the plan. By planning, we mean filling of a segment with elements; the planned segment is typically composed of 2 to 8 elements.

After that, the **Time Planning** follows. For each segment, the actual flight plan is adjusted so that the time of flight through the segment matches the time intervals defined in the start and end waypoints belonging to the segment. The adjustment is done in two ways: through stretching the segment by altering some of its parameters (so that it becomes longer and thus the flight takes more time) and through setting the velocity of the plane at the beginning of the segment. For this reason, we place a warp element at the start of each segment, which allows for a step change of velocity at the segment boundaries. By using this simplification, our planning algorithm is very fast.

The flight path for an airplane can be planned with respect to predefined no-flight zones (such as areas around power plants, military facilities or natural terrain obstacles). **Avoiding the no-flight zones** is done as follows: the segments of the flight plan generated in previous phases are tested for collisions with the no-flight zones. If any segment intersects a no-flight zone, it is replaced with a flight path which bypasses the respective no-flight zone(s). This is accomplished through finding suitable "bypassing" waypoints, which roughly define the collision-free flight route, and through planning a flight path through them. The situation is shown in Figure 9.

The problem of finding the bypassing path is thus equivalent to finding suitable waypoints that define it. These waypoints have no time constraints. Our approach to finding these waypoints is based on a spatial subdivision of the area of interest by an octal tree to a set of cubic cells. The cells intersected by a no-flight zone are marked as full, while the rest of them is marked as empty. By using the well-known A^* algorithm we find a sequence of empty cells, connecting

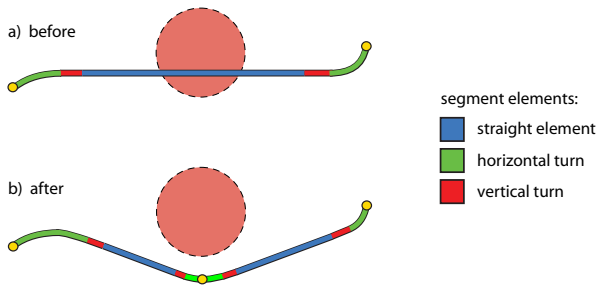


Figure 9: Avoiding the no-flight zone.

the cells containing the start and end waypoints of the replanned segment; this sequence forms a continuous "tunnel" which bypasses the no-flight zones. The waypoints are then placed in the "tunnel". . This approach is similar to that described in [6].

6. REAL DATA INTEGRATION

The designed ATC system integrates four data sources:

- **National Aeronautics and Space Administration (NASA)** – a mosaic of Landsat7 images[2] at the maximum resolution of approximately 50 meters per pixel was used as an underlying ortophoto map of the United States
- **U.S. Geological Survey (USGS)**[3] – detailed vector shapes of 50 U.S. state boundaries; a set of more than 650 U.S. airports, including their names, GPS coordinates with the corresponding average numbers of enplanements per year; a set of more than 26 thousand major U.S. highway segments
- **Geographic Names Information System (GNIS) database**[4] – a set of more than 24 thousand U.S. populated places, including their names, GPS coordinates with the corresponding population; a set of more than 80 U.S. powerplants, including their names with GPS coordinates to act as the no-flight zones
- **ATC flight data** – airplanes including position and direction; rough flight plans in the form of series of points in space through which the airplane is supposed to fly on its way; detailed flight plans

7. FLIGHT VISUALIZATION

The **real-time visio** allows the user to overview the entire simulation in a 2D/3D environment, to efficiently navigate through it, but also to provide the user with all the important data and information at the same time.

An open-source 3D game engine CrystalSpace [2] was used as a platform on top of which the visualization component is built. For efficiency and performance reasons, the entire visualizer is written in C++ and it internally utilizes the OpenGL graphics engine. A binary-coded network communication protocol has been defined and implemented to allow fast data exchange between the core system and the visualizer component.

[2] `http://onearth.jpl.nasa.gov`
[3] `http://seamless.usgs.gov`
[4] `http://geonames.usgs.gov`

The communication is duplex: by sending messages to the visualizer, the core system notifies the visio that an airplane has been spawned, destroyed, it has changed its position etc. On the other hand, the visualizer sends messages to the core system e.g. when it requests specific information about a certain airplane upon user's demand, such as its flight plans. In response to this request, the simulation system selectively sends the required data. This behavior was introduced to reduce the amount of data that need to be sent over the network – instead of sending all the data continuously even if they are not needed.

The visualizer provides two main display modes: a two-dimensional (2D) view (Figure 10) and a three-dimensional (3D) view (Figure 11). They both work with the same underlying data, only the way of depicting them differs.

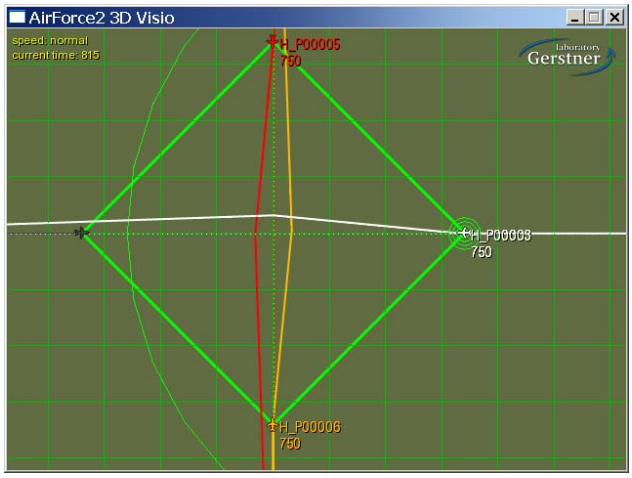


Figure 10: Real-time visualizer: 2D view

The **2D mode** provides a radar-like top view of the scene. Airplanes present in the system are displayed as 2D icons that visualize both current position and direction of each airplane. Once the airplane is selected, the camera starts following it so that it stays in focus. A set of zones is displayed around the selected airplane as a group of concentric circles. Most importantly, these refer to the collision range (innermost) and the visibility range (outermost) of each airplane. The current flight plan can be displayed for the selected airplane and/or for all airplanes that are visible to it. Flight plans in 2D view are represented by solid polylines that interpolate through the intended route.

The **3D mode** (see Figure 11) is useful for observing the entire simulation in a natural 3D environment that allows the user to get a better insight of the spatial relations between airplanes, the actual 3D shapes and possible collisions of flight plans, series of waypoints etc. In 3D, flight plans are displayed as three-dimensional semitransparent corridors of a rectangular profile. As the visio is required to cope with as many as hundreds of airplanes, massive optimizations of visualization of 3D space have been implemented. One of the key features in this respect is the incorporation of multiple levels of detail (LOD) for all 3D models.

Another important feature of real-time visualizer is the ability to interactively control the speed of the simulation. The visualizer displays the current speed of the simulation in the top left-hand corner of the screen, together with the current simulation time.

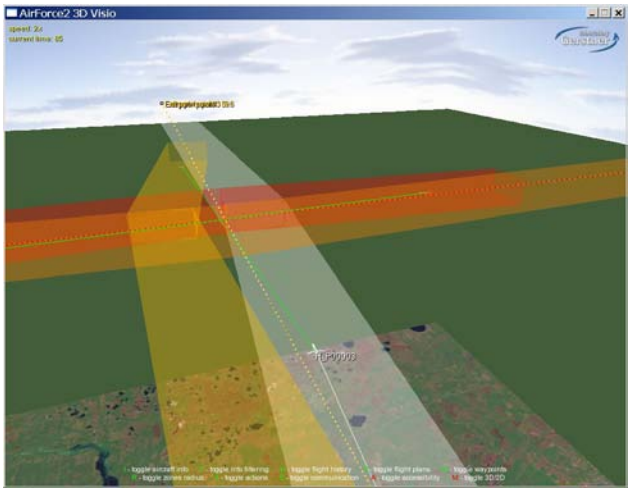


Figure 11: Real-time visualizer: 3D view

The simulation can also be accessed using a **remote web client**, a Java application (see Figures 12 and 13) that connects via network to the simulation system which acts as a server and provides all its clients with regular data updates. This way a number of users can concurrently observe and interact with the same simulation. Accessing the simulation system via network is as simple and straightforward as opening a web browser and entering the IP address of the computer on which the *core system* is currently running.

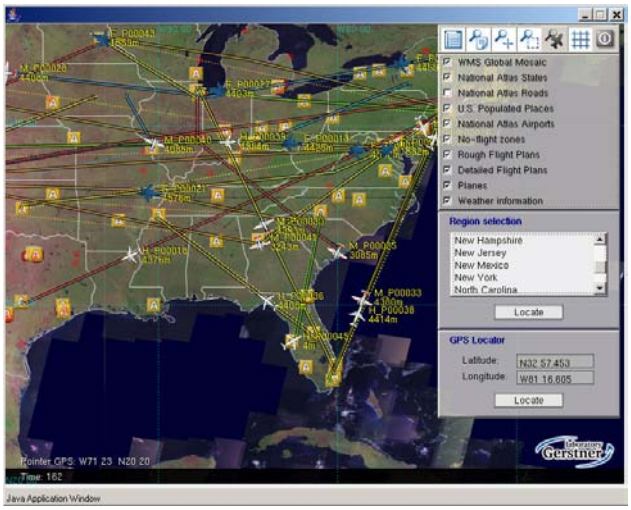


Figure 12: Remote client GUI overview

Before the user can access the simulation, he needs to log in the system by entering a valid combination of username and password. Remote client authentication procedure is secured using one-time hashes for password validation. To minimize network traffic between the remote client and the ATC core system, a combination of HTTP and a special binary protocol is used.

The ATC core system uses a Java Web Start application for loading and starting the client. The client is built on top of JOGL libraries (Java binding for OpenGL [5]) which are

used for accessing graphics 3D acceleration.

The icon palette located in the top right hand corner of the client window gives the user access to application controls, e.g. data layer menu, zooming to a named region, zooming to a selection, zooming to a particular airplane, GPS locator, coordinate grid control and the logout button.

As the user changes the view, the appropriate data segment of the virtual map is requested and downloaded from the server, and therefore at any moment the client needs to keep only a minimal amount of data, which results in its fast and efficient performance.



Figure 13: Flight plans avoiding no-flight zones

8. CONCLUSION

In the paper we present how agents can be used for the negotiation-based deconfliction in the Air-Traffic Control. We describe the prototype of the ATC system where we integrate our agent-based solution with several external data sources. The external online data sources provide necessary information about ground surface, airport locations, weather conditions, no-flight zones and area boundaries for the ATC system. We introduce two types of system visualization: real-time and remote using an internet browser.

The ATC prototype has only simple deconfliction rule-based mechanism for the flight plan changes which assumes that all airplanes use the same deconfliction rules. The system will be extended so that airplanes that detected future collisions would iterate through *monotonic concession protocol* (MCP) to find new flight plans that are collision free. The monotonic concession protocol is a simple protocol developed by Zlotkin and Resenschein for automated agent to agent negotiations [12, 8]. Both airplanes prepare a set of possible flight plan changes scored by the utility function. The utility function includes pilot's own intentions including flight priority, fuel restrictions, time restrictions, etc. From all collision free combinations of flight plan pairs, the possible solution set is created. The iteration protocol results in a commonly accepted solution of the collision. Then each airplane applies the respective flight plan changes. However, the iteration solution can lead to a situation when the solution is not found fast enough. The process has to be extended with an emergency solution that is used when the iteration process does not lead to any fast solution. As the emergency solution, the game theory approach can be used.

Acknowledgement

Effort sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-04-1-3044-P00001 extension of the FA8655-04-1-3044. The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon[5].

9. REFERENCES

[1] A-globe. A-globe Agent Platform. `http://agents.felk.cvut.cz/aglobe`, 2005.

[2] CrystalSpace. Crystal Space 3D – opensource game engine. `http://www.crystalspace3d.org`, 2004.

[3] FIPA. Foundation for intelligent physical agents. `http://www.fipa.org`, 2004.

[4] J. C. Hill, F. R. Johnson, J. K. Archibald, R. L. Frost, and W. C. Stirling. A cooperative multi-agent approach to free flight. In *AAMAS*, pages 1083–1090, 2005.

[5] JOGL. Java Bindings for OpenGL. `http://jogl.dev.java.net`, 2005.

[6] S. Kambhampati and L. Davis. Multiresolution Path Planning for Mobile Robots. *IEEE Journal of Robotics and Automation*, RA-2(3):135–145, 1986.

[7] M. Rehák, M. Pěchouček, and J. Tožička. Adversarial behavior in multi-agent systems. In M. Pechoucek, P. Petta, and L. Z. Varga, editors, *Multi-Agent Systems and Applications IV: 4th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2005*, number 3690 in LNCS, LNAI, 2005.

[8] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter*. The MIT Press, Cambridge, Massachusetts, 1994.

[9] R. Schulz, D. Shaner, and Y. Zhao. Free-flight concept. In *Proceedings of the AiAA Guidance, Navigation and Control Conference*, pages 999–903, New Orelans, LA, 1997.

[10] D. Šišlák, M. Rehák, M. Pěchouček, M. Rollo, and D. Pavlíček. **$\mathcal{A}$-globe**: Agent development platform with inaccessibility and mobility support. In R. Unland, M. Klusch, and M. Calisti, editors, *Software Agent-Based Applications, Platforms and Development Kits*, pages 21–46, Berlin, 2005. Birkhauser Verlag.

[11] C. Tomlin, G. Pappas, and S. Sastry. Conflict resolution for air traffic management: A case study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control*, August 1997.

[12] G. Zlotkin and J. S. Rosenschein. Negotiation and task sharing among autonomous agents in cooperative domains. In N. S. Sridharan, editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 912–917, San Mateo, CA, 1989. Morgan Kaufmann.

[5]The views and conclusions contained herein are those of the author and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

Automating Supply Chain Negotiations using Autonomous Agents: a Case Study in Transportation Logistics

Sander van der Putten[1,2,3] Valentin Robu[1] Han La Poutré[1,3]

Annemiek Jorritsma[2] Margo Gal[2]

[1]CWI, National Center for Mathematics and Computer Science
Kruislaan 403, NL-1098 SJ
Amsterdam, The Netherlands

[2]Vos Logistics Organizing B.V.
Toernooiveld 304, NL-6525 EC
Nijmegen, The Netherlands

[3]Eindhoven University Of Technology
Technology Management Department
P.O. Box 513, NL-5600 MB
Eindhoven, The Netherlands

putten@cwi.nl robu@cwi.nl hlp@cwi.nl

ABSTRACT

This paper presents a case study for the application of agent-mediated negotiation techniques in transportation logistics. More specifically, we consider the interaction between several logistics service providers negotiating over the allocation of transportation orders. In this context, we show that automated negotiation techniques (especially multi-issue or multi-item negotiation) can bring significant advantages, by allowing parties to discover jointly profitable bundles (allocations) of orders. The model, evaluations and results reported in this paper concern the business processes of Vos Logistics, one of the largest European transportation logistic providers.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent agents
H.4.2 [**Information Systems Applications**]: Logistics

General Terms

Management, Performance, Design, Measurement.

Keywords

Automated negotiation, multi-agent systems, transportation logistics, supply chain management.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

1. INTRODUCTION

Negotiation represents a key form of interaction in agent-mediated electronic markets that transcend the sale of uniform goods. Through negotiation, suppliers and consumers can reach complex agreements in an iterative way, which better match the needs and capabilities of different parties. Recent research work has led to a variety of new, increasingly complex algorithms to deal with application of automated negotiation (especially multi-issue or multi-item negotiation) in a variety of settings [4, 9, 11].

Transportation logistics and supply chain management represents a challenging, but potentially very fruitful area for application of automated negotiation. In this setting, it is often possible for all parties in a supply chain to obtain joint savings through more efficient, fine-grained matching of their requirements and capabilities (i.e. orders and transport capacity). The increasing complexity of modern supply chains, as well as increasing competitive pressures in this market has led to an increasing demand and interest for such distributed optimization techniques.

The practical impact of improved allocation which can be achieved through such techniques can be significant. For example, in the Netherlands, the average transport performance is between 40% and 60%[1]. [2]. Improving this utilization rate is also the goal of the DEAL (Distributed Engine for Advanced Logistics) project, which groups together several universities and large logistics service providers in the Netherlands. The work reported here is also carried out in the framework of this project, involving the CWI and Vos Logistics.

The paper describes a case study application of automated negotiation to the logistics domain. It is important to stress that, in this paper, we do not aim to propose a new algorithm or analyze

[1] Defined a defined as the extent to which transportation units are utilized as percentage of total capacity, usually given in ton-kilometer. Thus, a truck that is driving with only half its capacity has a transport performance of 50%.

fundamental aspects of agent-mediated negotiation techniques as such (this has already been done elsewhere, in papers by the same authors at the research track of AAMAS and AMEC [4, 9, 6]. Our goal is to focus more on the specifics of the application domain and to describe our experience with agent-mediated negotiation techniques in the real-life case of the organizing branch of Vos Logistics. The paper is largely based on the Master Thesis work of the first author, performed jointly at Vos Logistics and CWI [13].

The remainder of the paper is organized as follows. In Section 2 we describe the logistics of the transportation domain and briefly outline the profile of Vos Logistics as well as the transportation market in which it operates. Section 3 goes more in depth, by providing a more detailed analysis on how the order allocation and negotiation processes are actually conducted at the Vos Logistics Organizing branch in Nijmegen. Section 4 briefly outlines the potential of existing techniques from agent-mediated negotiation and discusses to what degree and how they are relevant or applicable to this domain. Section 5 describes our proposed multi-issue negotiation solution and how it can be used to discover more efficient allocations (or bundles) of orders during trading. Section 6 presents an analysis of the gains which can be actually achieved by bundling of transportation orders, based on the real order set of Vos Logistics Organizing. Section 7 provides a discussion of how our approach compares to other approaches to this problem (both from the theoretical literature and industrial-grade solutions). We conclude the paper by pointing out some of the challenges which can be encountered in the application of a pilot project on automated negotiation at a company such as Vos Logistics.

2. THE TRANSPORTATION LOGISTICS DOMAIN

Several trends have recently produced a significant impact on the area of transportation logistics. One of these is an increase in competition, with the continual entry of new carriers in the market pushing down expected profit margins. Another one is the increasing complexity and sophistication of modern supply chains. In fact, due to increasing and shifting trade patterns, not only the complexity of transportation chains has increased, but also their variability and dynamics. These challenges require increasingly sophisticated optimization solutions, that exploit all the niches in the supply chains involving different companies.

We note that his optimization has to occur both intra-company (where a company optimizes its own activity), but increasingly also inter-company, between different actors that do not necessarily share the same goals. Inter-company optimization poses considerable extra challenges, since the parties forming the supply chain can be self-interested, so they may be unwilling to share private planning information and strategic concerns make the problem considerably harder than cooperative case. The solutions proposed in this paper specifically address this case of inter-company optimization of the logistic chain.

The transportation management literature [12] identifies two main types of orchestrator companies for transportation logistic chains:

- Third Party Logistics service providers (3PL), which orchestrate the logistic processes and assures they are well executed according to a current logistic concept
- Fourth Party Logistic service providers (4PL), which orchestrate the supply chain and are responsible for the design of the logistic concepts.

There is, however, considerable disagreement about the exact meaning of these terms, both in existing literature and in practice. In our approach (and the remainder of this paper) we use the term 3PL provider companies to denote those that have their own transport capacity (i.e. truck fleet) and plan (or orchestrate) their movement, based on a set of orders from shippers and other companies. Fourth Party Logistic companies (4PL) have recently emerged as a result of the increasing complexity of supply chains and mostly have a role of coordinating these chains. In this paper we denote as 4PL companies those that do not have their own transport capacity: they receive large transport orders from shippers and then distribute them among a set of 3PL companies or independent carriers. This process involves direct negotiation with 3PL companies and often entails breaking up large orders or bundling orders for partial loads, in order to increase efficiency. The remainder of this section describes in more detail the company which forms the object of our solution, as well as the market in which it operates.

2.1 Company profile

Founded in 1944 as a one-truck company, transporting loads between Oss and Nijmegen in The Netherlands, Vos Logistics has grown into one of the larger logistics service providers in Europe. It has over 3000 trucks, 10000 trailers and containers, 350000 square meters of warehousing sites, 325 storage silos and 2 rail service centers. Vos employs 5000 people working at more than 45 locations throughout Europe. Annual turnover is around 800 million euro.

The increasing complexity of transportation chains has determined Vos to offer new solutions to its large corporate customers (shippers), which can now outsource all of their transportation activities to Vos. This lets them avoid the problem of finding and negotiating with individual suppliers, billing, following up orders etc. Another advantage of using this outsourcing service for large shippers is that Vos Logistics has a much better knowledge of the transportation market, so it is better positioned to find suitable sub-contractors.

Vos Logistics Organizing from Nijmegen (henceforth abbreviated VLO in this paper) is a subsidiary of Vos Logistics B.V. that was set up in order to handle such complex supply chain orchestration activities. Based on the taxonomy above, VLO (the subsidiary) can be seen as a 4PL company, though its parent company, Vos Logistics performs mainly 3PL activities. Hence, VLO acts as an intermediary company that acquires large (sets of) orders from suppliers and negotiates the allocation of the orders, the terms of transportation (i.e. delivery deadlines, destination) as well as the price with which 3PL companies that subcontract these orders.

2.2 Market organization. Closed vs. open group negotiations

This section describes the operation of a 4PL logistic company, which simultaneously interacts with customers (shippers), a set of 3PL companies (with actual carrying capacity) and the open market. For the purpose of this paper we identify this 4PL company with Vos Organizing (VLO), but the model should be a general enough to be applied to other companies in the same field.

Figure 1 presents a graphical model of this interaction.As depicted in Figure 1, there are two main mechanisms for allocating orders received from shippers:

- **Closed group negotiation.** Most of the orders (i.e. around 80%) received by the 4PL company are currently not auctioned off to an outside market, but are allocated among a small group of trusted 3PL carriers. The size of this group is around 5-10 companies. The composition of this "closed group" of trusted companies is based on trust and a history of good business relationships – in fact for each shipper there may be a customer specific list of companies with which the negotiation takes place. In current practice, the protocol for conducting negotiations is usually bilateral and sequential.
- **Open market.** A small subsets of orders (around 20%) is offered on the "open market". Usually this means offering the loads through transportation matching sites such as Teleroute [15]. The important point here is that there are no barriers of entry or "admission rules" on these sites (i.e. any company or individual carrier in the Netherlands or in the whole of Europe can make offers by phone for these orders)

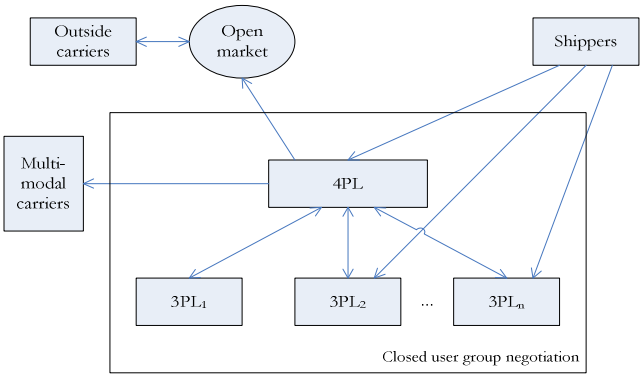


Figure 1: Interactions between parties in a multi-party logistics setting

We argue that both types of mechanisms can benefit from agent-based automation, though they may require different mechanisms. For the open group negotiation, auction protocols would probably be the most suitable, since here all agents are strictly competitive and there are no barriers of entry.

However, in our approach we concentrate more on the first set-up, the closed negotiation, chiefly because it is more applicable for our particular setting, i.e. it has a higher business impact and 80% of the orders are allocated in this way. For this set-up we considered two types of choices. The first one is automated negotiation, especially multi-issue and multi-item negotiation models that enable parties to reach mutually profitable (Pareto-efficient deals [8]) that go beyond strictly price competition. A second choice would be to have a combinatorial auction protocol, where the 4PL company would act as a "center".

In this paper we mainly explore the first option, since we feel it allows us more flexibility in dealing with side constraints and incomplete information. However, the auction option is also being considered, especially if the protocol could be adapted to quantify issues such as trust or other constraints. Further discussion of this issue is given in Section 5.

3. ORDER ALLOCATION AND NEGOTIATION PROCESSES AT VLO

This section aims to give a more in-depth look at how the order allocation and negotiation processes are currently performed at Vos Logistics. This is an important part of any industry analysis, in order to assure that the models considered can lead to usable pilot applications within the company and are not of merely theoretical interest. We focus here on the daily outsourcing (i.e. "spot orders"), which is the main area that could be automated through agent technologies, since it is unlikely that automated agents can be entrusted to take higher-level or strategic decisions.

3.1 Order set characteristics

There are several characteristics (parameters) that describe every transportation order. The following is a non-exhaustive list:

- The volume (or mass) to be transported
- The price (computed either per unit volume or unit mass)
- The time windows (in the form of time for picking up the goods and the deadline for delivery)
- Locations where the order is picked-up/delivered
- Extra transport conditions (e.g. refrigeration, liquids, fragile merchandise etc.)
- Arbitrary constraints that the shipper may impose (e.g. do not transport my goods at the same time as company X's)
- Flexibility of contractual terms (e.g. time ahead of the actual transport when a contract may be re-negotiated or cancelled)

Due to the time limitation of human planners, currently most of the parameters are not explicitly negotiated about every order. From the above set we identify two characteristics that are particularly important from the perspective of how orders are currently priced by carriers:

- The percentage a load takes from the total volume of a truck, commonly referred to in the field through the acronyms FTL (full truck load) and LTL (less than truck load)
- The "fruitfulness" of the region the order originates from or is to be delivered. Here by "fruitfulness" we mean the likelihood that there will be return freight from that region.

In the following sections we discuss these two dimensions in more detail and show their importance with respect to the different types of bundling.

3.2 Bundling of LTL shipments

Making bundles out of individual orders is a well known way to increase offer value [9, 4]. Two potential improvements have been found in the area of bundling; consolidating multiple part loads with overlap in route and the offering of return shipments or subsequent loads at the unloading destination.

VLO can gain considerable savings when multiple LTL or part loads, with overlap in (part of their) route, are offered as a bundle. To illustrate, a typical price table, using actual carrier data, is displayed in Figure 2 which depicts the price per pallet for the various loading percentages of a truck[2]. From Figure 2: Relationship between load and price per pallet, we can see that the pricing of LTL (less than truck load) orders follows a logarithmic–like pricing function. An order for a quarter of a truckload is roughly 50% of the FTL price for example whereas half of a truckload equals approximately 70% of the FTL price.

Currently, the process of bundling is executed by hand by VLO for the shippers that request it. A number of solutions exist, both agent-based and non agent-based, to automate and optimize the process of making efficient bundles out of a given set of orders.

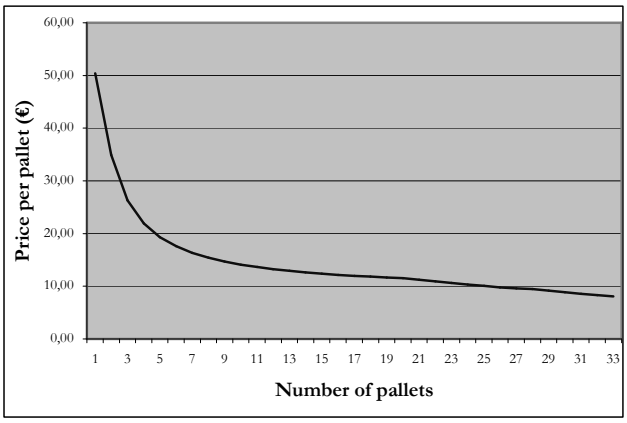


Figure 2: Relationship between load and price per pallet

3.3 Bundling of return or subsequent freight

For the bundling of a shipment with a corresponding subsequent or return shipment, the notion of fruitfulness plays a central role. In a fruitful region much freight is offered and a truck runs only a limited amount of empty kilometers, whereas in a fruitless region not much freights are offered and on average trucks run a significant amount of empty kilometers before pickup of the next freight. Depending on the fruitfulness of the unloading area and the quality of the carrier's own network and his own capabilities in matching demand and supply, a truck has to run a certain number of empty kilometers from the unloading point to the next pickup location. If VLO was to offer a new freight at or close to the unloading location, a significant discount might be obtained from the carrier as the truck is decreasing its empty kilometers. Savings depend on the area the new freight is offered in and the quality of the carrier's own network. If a lot of freight is offered in the region, the discount will be smaller. Similarly, if a carrier's own network is mature and the carriers are very capable of finding subsequent shipments, the discount will be smaller.

This concept can be seen in current business practice at VLO, in which a certain price/km value is used for a 'country of origin' to 'country of destination' combination. Different values are used for each combination, e.g. for a shipment from France to the Netherlands a lower value is used than for shipments from the Netherlands to France. This has to do with the amount of return freights generally available in the country of destination. If a truck coming from France has to deliver freight in the Netherlands, chances are high that there is a freight weighting to get from the Netherlands to France. It is different the other way round. If a truck is empty in France there is a much smaller chance that there are freights weighting to get from France to the Netherlands and as a result there is a higher chance that the truck has to return empty. To illustrate, rates from France to the Netherlands are about € 0,60 to € 0,80 per kilometer, whereas prices from the Netherlands to France range from € 1,25 to € 1,50 per kilometer.

4. AUTOMATED MULTI-ISSUE NEGOTIATION: MODELS & RELATION TO THE TRANSPORTATION DOMAIN

The concept of electronic negotiations or e-negotiations is a very broad one. As shown in [1], it covers a whole spectrum of negotiations ranging from the unstructured exchange of messages using email and chat systems, to partially structured e-negotiations supported by negotiation support systems, negotiating software agents or online auctioning platforms for selected tasks, to completely structured negotiations conducted autonomously by computer systems.

The scope of our approach is the automated case, where parties are represented by software agents. More specifically, we are interested in multi-issue or multi-item negotiation models, which allow agents to exploit so-called "win-win" opportunities. As shown by Rosenschein & Zlotkin [10], such negotiations represent non-zero sum games, where "as the values shift in multiple directions, it is possible for both parties to be better off". In our application domain, this means that negotiations between agents representing different transport providers (carrier companies) do not focus exclusively on price, but also allow the discovery of mutually beneficial bundles of orders, which are customized to accommodate for niches in their supply chains.

In the following we discuss some of the issues identified by our case study at VLO that could potentially form the object of automated negotiation techniques. Some of them (especially the question of time windows) is important because it influences the potential gains which can be achieved through the bundling of orders.

4.1 Negotiation over price

In current practice at VLO, price is the only issue negotiated. This negotiation currently is performed by humans and takes place over the phone. Even for this one issue there are some advantages to be gained from automation.

First, efficiency could be improved, since transaction costs would be reduced by automating the process. Furthermore, humans have

[2] A regular trailer (truck) has a volume of 33 euro pallets, while large-volume trailer can take 38 pallets. Note that a euro pallet is 1.20 meters wide and 0.80 meters long.

a cognitive limitation on how many parties they can negotiate with at the same time and how many counter-offers they can make in each negotiation. In current VLO practice this leads to rather large bid-offer spreads between offers and between negotiations for similar orders. If these negotiations could be automated, this would increase the effectiveness of the deals made as well.

However, we should point out that automated negotiations are however no panacea and should be treated with care. In human-controlled, traditional negotiations, the communication, formulation of offers and making of concessions is a vehicle for both a consensus and understanding [7]. In vaguely defined negotiation situations, a human-controlled negotiation process will probably be more efficient and effective than what would be possible with even the most sophisticated intelligent agents currently available. Therefore, having a clear protocol is crucial in automated negotiation models.

4.2 Negotiation over time windows

Currently, time windows (i.e. time interval when an order is to be delivered) are considered to be fixed, due to the complexity and high cost of negotiating a potentially exponential number of combinations of order bundles/time windows. It has however been indicated that VLO's customers would be willing to be more flexible if doing so would result in savings to them. Basically there are two ways to handle this option. VLO could give a discount if shippers would be willing to extend their time windows, e.g. a 5% discount if the time windows are increased from 1 to 3 days. This would require that VLO knows what savings can be achieved in the various situations, to make sure that the discount percentage given is a realistic one. The other option is to let the customer specify discounts or penalties in case of earlier or later delivery.

	Three days earlier	**Two days earlier**	**One day earlier**	**Original date**	**One day later**	**Two days Later**
Initial amount/ shipper	€ 650,00					
Penalty for date change	Not accept	- €150,00	- €50,00	€ 0,00	- €150,00	Not accept
Actual revenue	n/a	€ 500,00	€ 600,00	€ 650,00	€ 500,00	n/a
Best carrier's offer	n/a	€ 530,00	€ 530,00	€ 630,00	€ 490,00	n/a
Gross margin of VLO	n/a	- € 30,00	€ 70,00	€ 20,00	€ 10,00	n/a

Table 1: Example of margin fluctuations with flexible time windows

Table 1 illustrates this later concept through a fictional example (but which, nevertheless matches characteristics of real orders – as discussed with VLO planners), how VLO can benefit from deviating from the original date, e.g. because capacity on the original date is hard to find and prices are high.

There are several reasons why rates could fluctuate at different time periods:

a) Using extended time windows there is the possibility to better match freights with the location of a carrier's vehicles. For example, in the case of shipments from Germany to the Netherlands, a Dutch carrier will be cheaper as they are likely to have freight from the Netherlands to Germany and can pick it up on the way home. However, these carriers do not arrive in Germany until Tuesday, as trucks and drivers return to their home base during the weekend. If the time windows for a freight which is scheduled at Monday are extended by 1 day, a Dutch carrier is able to take on this shipment, whereas VLO otherwise would have to go with a more expensive German carrier.

b) Extended time windows might also lower specific costs. Again, a simple example helps illustrating this principle. Consider freight from the Netherlands to the North of France with an original pickup date of Wednesday and one from the North of France back to the Netherlands with an original pickup date of Thursday. If the time windows are extended, the carrier could load the freight to France very early Wednesday morning for example and drive to the North of France. If the carrier can pickup the return freight that same Wednesday right after unloading, he can drive back the same day. As a result, instead of chartering an international driver, a much cheaper so-called national driver can take care of these shipments, as there is no need to sleep over in the truck and wait to pickup the freight the next day.

Besides these effects, specific to certain carriers, there may be market-wide or industry-wide effects as well, such as the "end of the month" effect, when shippers want to transport the goods to remove them from their inventory lists, and one-time effects caused by disruptive events such as strikes etc., which cause a backlog of orders in certain periods. In this case, extending the time windows to help carriers better cope with such events may be desirable.

5. PROPOSED SOLUTIONS FOR MULTI-ISSUE NEGOTIATION AND BUNDLING

From the discussion in Sect. 2-4 above, we can see that there is a wide variety of issues that could, potentially, be considered in automating the negotiation processes al VLO.

Here we focus on one of them: automating the "closed group" negotiation (see Sect. 2) between the 4PL and 3PL companies over the distribution of transportation orders. We chose this setting because of its relation both with existing research lines at CWI and with current practice at VLO. In our models, we consider that orders can exhibit utility dependencies of two types:

- Complementarity dependencies, if the orders can be transported by the same truck (i.e. their pick-up/destination regions, delivery time windows and/or capacity fit)
- Incompatibility (or substitutability) dependencies, if orders cannot be bundled together, due to a mismatch between their delivery time windows and/or capacity.

In our model, dependencies are represented in the form of a utility graph, which encodes complementarity/substitutability relations between orders. In previous work (in part by the same authors) presented at the research track of AAMAS [9] a general framework was proposed to handle this type of dependencies, and this was successfully applied to the logistics setting. A software tool was built for this paper (a screen shot of the negotiation

phase is shown in Fig. 3). The tool should enable planners to compute (close to) optimal bundling of different sets orders and to explore different scenarios, by changing the constraints (e.g. time window constraints) and the information shared between parties.

This approach works by dividing the one-to-many negotiation between the 4PL and 3PL companies from the closed group in a series of one-to-one negotiation threads. This approach allows us to preserve the privacy of each party, since each company has to reveal their exact preferences and valuation only to its own automated agent. Furthermore, agents representing different companies do not have to reveal their preference information directly to other agents in the group, since approaches to automated negotiation such as those proposed in [9] allow agents to reach efficient agreements incrementally, with partial or incomplete information revelation.

There are, however, challenges when applying this model to a real-world setting. First, we assume all companies are represented by a proxy agent that does the bidding on their behalf. Second, we do not say much about how planning is performed, our approach concentrates only on optimizing the task allocation between a group of companies. Clearly for such approach to work well in practice, the planning of transportation within each company would need to be automated as well and connected to the negotiating agent. Alternatively, if planning is still done by humans, better mechanisms are needed to encode and elicit the preferences over the bundles - especially time and capacity preferences. An alternative to this automated one-many negotiation approach would be to use an auction-type of approach, instead of automated negotiation. For example, in other work, such as proposed by Sandholm and exemplified in the engine built by Combinenet [14], a similar problem is solved by means of a combinatorial auctions. This has other advantages (e.g. humans could also specify bids, so full automation from proxy agents is not needed), but also assumes that parties are willing to share truthfully their planning information and there is one trusted "center" to compute an optimal allocation.

In any case, it is realistic to assume that at any time, human planners or negotiators should be allowed to overrule the suggested agreements reached by their negotiation/auction proxies. As such, we envisage that the system would work more as a decision support tool which helps to explore various allocation scenarios, rather than a replacement for human negotiators.

A practical disadvantage of traditional auction mechanisms is that competition occurs entirely on price and the seller (in our case, the 4PL company allocating the orders) cannot discriminate between subcontractors based on other aspects, such as trust. This is a departure from real-life settings, where it may be possible that a party with a higher bid is offered the deal, if it is deemed more trustworthy (although this is difficult to quantify, since it is based on previous interactions and experience of the human planners).

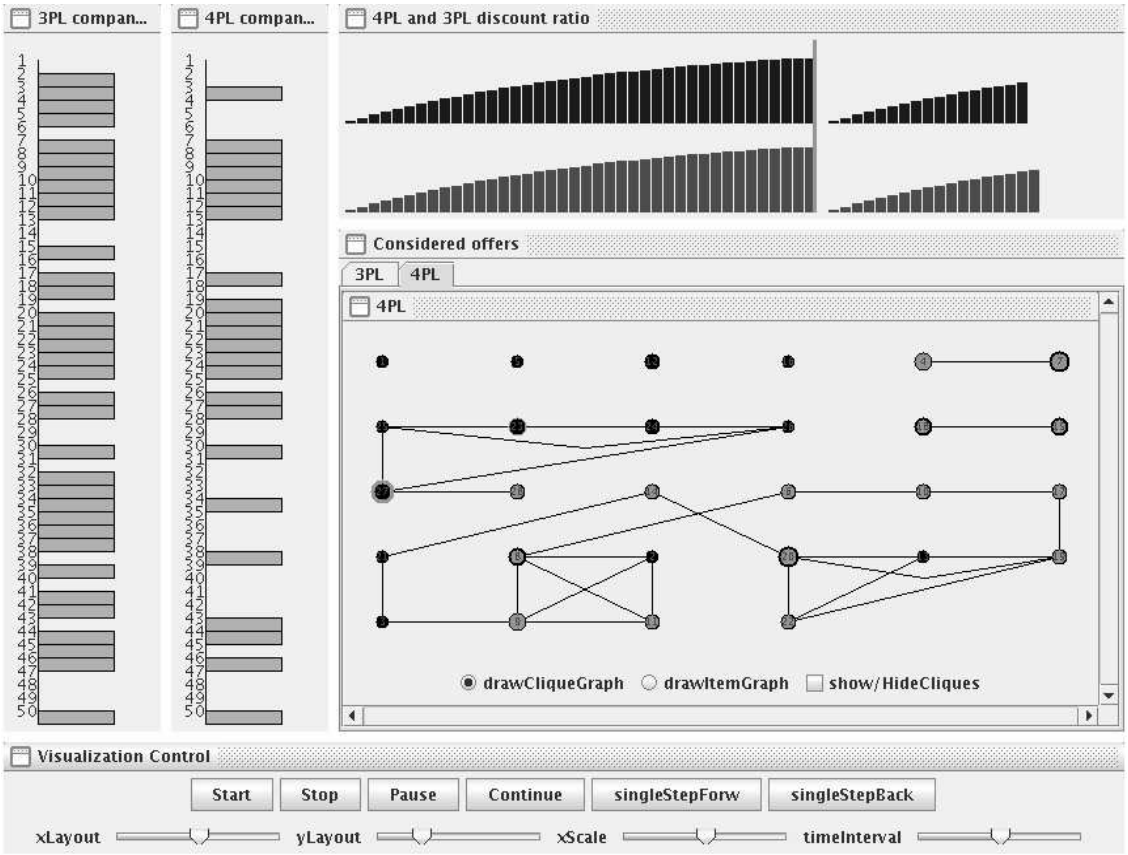


Figure 3: Screen shot of the negotiation tool built to visualize a closed group negotiation thread

By modeling the negotiation as a set of private one-one threads, asymmetric concessions to different parties (based on their trustworthiness) are possible. An alternative would be to modify the traditional open auction mechanism to take into account such considerations. Two alternatives have been proposed (c.f. [13]):

- Allocating closed-group orders through an auction with a pre-selection step, in which only certain companies are allowed (or invited) to submit bids. In this way, 3PL companies are already "screened" before accepting a potential offer.
- Modifying the winner determination mechanism of a traditional auction, such that not only price is minimized, but also a weighed element of trust is taken into account.

All the alternatives discussed above (concurrent, bilateral negotiation and modified auction mechanisms) will be considered as potential implementation options in the proposed pilot project.

6. ANALYSIS OF POTENTIAL GAINS ACHIEVED FROM BUNDLING ORDERS IN THE CASE STUDIED

From Section 5, we see that the way our negotiation model works is by finding more efficient "bundles" of transportation orders, which best exploit the niches in the supply chains of different market participants. The theoretical model presented above has been shown, through simulations, able to discover Pareto-efficient allocations, even in incomplete information settings [9]. However, for a practical setting, validating a model through simulation is clearly not sufficient and we need to explore what are the potential savings which can be achieved through such techniques in practice.

A thorough investigation of the potential savings which can be achieved through bundling of orders for Vos Logistics was performed using historic order data. The data consisted of the set of over 6000 transportation orders handled by the forwarding desk of Vos Logistics for the year 2004. We should mention, however, that in order to protect the competitive advantage of VLO as well as the privacy of their customers and associates, much of this analysis cannot be reported in this paper. For this reason we report here only the percentage improvements achieved through bundling, but without entering into details or monetary figures.

There are two main types of bundling considered with respect to transportation orders:

- Bundling of LTL shipments (i.e. bundling of orders that are less than one truck load)
- Bundling of subsequent/return orders for FTL shipments (full truck loads)

For the bundling of LTL shipments, we considered orders for which both the pick-up/delivery locations (computed based on the first 2 digit of the postal codes area) and the date matched. Additionally the sum of part loads in a combination should not exceed the maximum load capacity.

For the bundling of FTL shipments, the delivery location and delivery date of a first order had to match the delivery location and delivery date of a second order. In both cases, feasible allocations are considered, of course, only those in which an order belongs only to one bundle.

The results of the savings which can be achieved for a set of over 6216 orders handled in 2004 are shown in Table 2:

	Savings over the total over the total value of LTL orders	Savings over the value of all orders in the order set
Bundling of LTL shipments	19%	2-3%
Bundling of FTL shipments	-	12%

Table 2: Potential gains which can be achieved through bundling of orders using historic VLO order data for 2004

From the results in Table 2, two things stand out. First, we see that the potential gains which can be achieved from bundling partial loads seem rather modest, compared to that from bundling return or subsequent shipments. This can actually be explained by the fact that, overall, the percentage of partial (i.e LTL) loads in the order set was rather small. However, if one considers only the value of the subset of LTL orders, one can see that using more efficient bundling the average truck utilization (load) increases from 18% to 39%. Using actual carrier price data, this results in savings of 19% in terms of price per unit load. The result is significant, since it is estimated that in the future, the number of partial truck loads handled by VLO is likely to increase considerably. Finally, for the results above, we assume that only same-day orders can be matched, in order to be consistent with current practice. If more flexible time windows are allowed (through better negotiations), we estimate that the potential savings from bundling can increase considerably.

7. DISCUSSION

There are several other approaches which address similar problems to the ones discussed above, both in the literature on agent systems, but also implemented by commercial companies. Here we review them briefly and compare their functionality to our approach. The LS/ATN system developed by Whitestein Technologies and presented in Dorer and Calisti [3] is a multi-agent system for dynamic transport optimization. The system provides agent-based optimization and execution capabilities that automate order dispatching, a crucial phase of transportation planning. The chief difference from our work is that the system proposed by Dorer & Calisti is mostly concerned with distribution of orders in the planning phase between trucks of the same company, while our system is concerned with automating the negotiation over loads between different companies in a supply chain.

The system proposed by Magenta Technologies [5] uses a multi-agent system to provide intelligent support for transportation scheduling. At the core, the system also relies on a virtual market engine to distribute loads. One feature of the MAGENTA system not currently considered by us are the semantic representation capabilities, as provided in the Ontology Management Toolkit.

Another set of solutions which address problems very similar to the ones described in this paper are provided by Combinenet [14].

Combinenet aims also at capacity optimization, by allowing more expressive bidding and scenario generation. We assume most of their solutions are based on efficient algorithms developed, in fundamental research by Sandholm and collaborators (reported, for example, in [11]). The main difference with the approach discussed in this paper, these solutions rely mostly on combinatorial auction algorithms to compute efficient bundle allocations.

Finally we should point out that there are also several electronic freight exchange sites, such as Teleroute [15] and Transporeon [16]. Teleroute (which is also currently used by VLO for their open market operations) acts more as a platform on which carriers and shippers advertise their order and transport capacity, but any negotiation and matching is done by human through the phone. Transporeon is another platform that provides a slightly more advanced functionality, such as bidding and order matching, yet well short of what can be achieved through automated agent-based solutions. Nevertheless, Transporeon has been used in daily operations by other partners of VLO, with considerable savings.

In comparison to these approaches, we focus mostly on the order allocation between several companies, more specifically on the order allocation between a 4PL and several 3PL companies. Our approach allows parties to discover optimal allocations in an iterative way, without revealing private planning information except to their own negotiating proxy agents. We envisage our tool being used as a human decision support tool, which allows human negotiators to investigate different scenarios, by changing the constraints, information shared and negotiation parameters of their agents. Humans are then free to accept/reject the proposed allocations, or re-run the negotiation in a new scenario.

Finally, our paper would not be complete without outlining some of the challenges we face in getting a pilot project on automated negotiation actually working at a company such as Vos Logistics. In general, in order for such a system to be adopted in practice, we need to show that the profit margins which can be achieved from adopting the new technology must be sufficiently high to warrant the effort and costs. Also, in our case, due to the fact that the system automates the interaction between several companies ("actors") in the supply chain, each of these must be willing to use this technology. Finally, the integration of automated planning of orders for each company with the negotiation proxy agent would be desirable in order to fully exploit the capacity of our system. Even if it is true that some of the companies use automated planning systems, we cannot assume this is necessarily the case for all companies in the supply chain. An alternative would be to allow humans to specify bids and offers, such as in an auction like protocol.

Despite the above challenges, our results from the approach so far have been very well received and a full scale pilot project is under way to apply such techniques in the daily outsourcing processes at Vos Logistics Organizing.

Acknowledgments

The authors wish to thank Walter Kusters and Rob Wijdeveld of Vos Logistics Organizing in Nijmegen for their advice and support at earlier stages of this work and Willem-Jan van Schijndel for his support with the data set.

8. REFERENCES

[1] Bichler, M., Kersten, G. and Strecker, S. Towards a Structured Design of Electronic Negotiations. *Group Decision and Negotiation* 12 (4): pp. 311-335.

[2] DEAL consortium (Almende, Erasmus Univ. Rotterdam, CWI Amsterdam, Vrije Univ, Amsterdam, Post-Kogeko and Vos Logistics) "Distributed Engine for Advance Logistics" *Project Plan*, 2003.

[3] Dorer, K. and Calisti, M. (2005). An Adaptive Solution to Dynamic Transport Optimization. *Proceedings of AAMAS'05 (Industry Track)*, pp. 45-51, Utrecht, 2005.

[4] Gerding, E. Somefun, K., La Poutré, J.A. "Multi-Attribute Bilateral Bargaining in a One-to-Many Setting", *Agent Mediated Electronic Commerce VI*, Springer LNAI, vol. 3435 (to appear).

[5] Himoff, J., Skobelev, P., Woolridge, M. "MAGENTA Technology: Multi-Agent System for Industrial Logistics", . *Proc. AAMAS'05 (Industry Track)*, pp. 60-66, Utrecht, 2005.

[6] 't Hoen, P.J., La Poutré, J..A. "A Decommitment Strategy in a Competitive Multi-Agent Transportation Setting". *Agent Mediated Electronic Commerce VI*, Springer LNAI 3048, pp. 56-72. Melbourne, 2003.

[7] Kersten, G.E. and Lo, G. "Negotiation Support Systems and Software Agents in E-Business Negotiations". *First Int. Conf. on Electronic Business*, Hong Kong, 2001.

[8] Raiffa, H. "The art and science of negotiation", *Harvard University Press*, Cambridge, Mass., 1982.

[9] Robu, V., Somefun, D.J.A., La Poutré, J.A. "Modeling Complex Multi-Issue Negotiations Using Utility Graphs", *Proceedings of AAMAS'05*, pp. 280-287, Utrecht, The Netherlands, 2005.

[10] Rosenschein, J.S., Zlotkin, G. – "Rules of Encounter", *MIT Press, Cambridge Massachussets*, 1994.

[11] Sandholm, T. "Algorithm for optimal winner determination in combinatorial auctions", Artificial Intelligence, Vol. 135 (1-2), pp. 1-54, Febr. 2002.

[12] van Amstel, P.W. and Goor, A.R. "Van Logistiek naar Supply Chain Management". *Kluwer Publishers*, 2003.

[13] van der Putten, S. – "Smart Bargaining: Intelligent software agents in the outsourcing processes of Vos Logistics Organizing", *Master Thesis, Eindhoven University of Technology*, 2006.

[14] Combinenet website. Retrieved November 28, 2005, from http://www.combinenet.com

[15] Teleroute website. Retrieved November 28, 2005, from http://www.teleroute.com.

[16] Transporeon website. Retrieved November 28, 2005, from http://www.transporeon.com.

MAGENTA Technology
Multi-Agent Logistics i-Scheduler for Road Transportation

Jonathan Himoff
MAGENTA Technology
130 Shaftesbury Avenue
London W1D 5EU, UK
Tel.: +44 (0) 20 7149 3750
jon.himoff@magenta-technology.com

George Rzevski
MAGENTA Technology
130 Shaftesbury Avenue
London W1D 5EU, UK
Tel.: +44 (0) 20 7149 3750
george.rzevski@magenta-technology.com

Petr Skobelev
MAGENTA Development
1a Osipenko St.
Samara, 443010, Russia
Tel.: +7 846 270 66 85
skobelev@magenta-technology.ru

ABSTRACT
The paper describes main features of an Intelligent Scheduler for Road Transportation Applications based on Magenta agent technology and characterized by a number of unique, advanced features such as real-time, incremental, continuous scheduling and schedule improvement, scalability from small to very large enterprise networks, multi-criteria schedule analysis and rich decision support for the user, pro-active agent negotiations.

The scheduler is event driven, which means that it construct schedules taking into account the occurrence of all schedule affecting events (such as the arrival of a new order, order cancellation, truck failure, road unavailability and driver no-show). As an event occurs the scheduler rapidly re-schedules only the affected part of the schedule thus causing the minimum disruption to already allocated resources.

The Scheduler has been tested on real data supplied by several UK Road Logistics operators.

The Scheduler was developed using protected Magenta technology for agent negotiations in virtual environment and knowledge discovery.

Categories and Subject Descriptors
H.4.2 [Information Systems Applications]: Type of Systems - decision support, logistics

General Terms
Management, Measurement, Design, Economics, Performance

Keywords
Intelligent scheduler, real-time planning, road transportation, multi-agent systems, virtual market, microeconomics, cargo to resource allocation, own fleet utilization improvement.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

1. INTRODUCTION
Road transportation logistics is amongst the most complex business problems. The complexity is caused by the exceedingly high variety of possible solutions (large solution space), which rules out traditional combinatorial search algorithms, and uncertainty due to high dynamics and volatility of the operational environment and openness of business networks, which makes optimisation impractical – a single optimisation run is typically an order of magnitude longer than a typical interval between two consecutive changes in operational conditions.

Resource allocation is an ongoing continuous decision making process in real time where criteria are changing "on the fly". Therefore the two key capabilities currently required from schedulers are supporting complex business networks and planning in continuous mode.

An effective road transportation scheduler must handle transportation instructions (TI) from many different loading points to many different destinations (e.g. customer locations and cross docks where cargoes are offloaded and consolidated) and many different routes by which orders can be delivered. Choosing the best route based on consolidation or other criteria is referred to as dynamic routing. The scheduler must also be able to allocate cargos of many different sizes and weights to many different types of trucks and trailers; take into account preferences of owners, operators and drivers and fit the schedule into numerous constraints imposed by warehouse working hours, driver work rules, safety regulations and enterprise policies, eg, on choosing between own fleet and third-party carriers. Different companies have different critical constraints, e.g. permission to override time or other constraints to achieve a more efficient schedule. The schedule created must be not only feasible but also efficient, i.e. possibilities for backhauls and consolidations should be found.

Complexity is also defined by the number and variety of orders (and other events that affect scheduling) per day and the number and variety of transportation resources such as trucks.

In addition, the scheduler is expected to rapidly reschedule orders and transportation resources affected by unexpected events such as: the arrival of new orders, cancellations, failures, bad weather conditions, road works and no-show of drivers or loading crews.

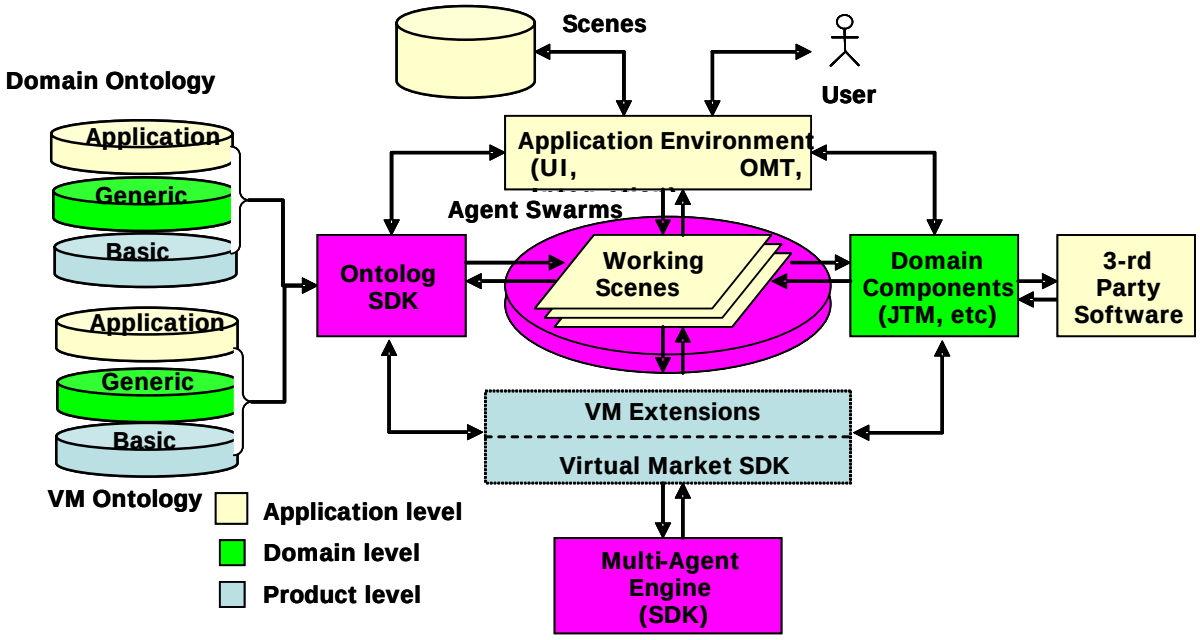


Fig.1.i-Scheduler Architecture

To enable enterprises to plan and re-plan continuously, reacting to events in real-time, schedulers must support "Planning / Commit / eXecute" (PCX) stages and plan across a multi-day planning horizon.

In the planning stage orders are assigned to a truck and its journey is constructed. During this stage orders can be added or removed and the route planned for the truck can be changed as a result of subsequent events

At some point the scheduler must commit the truck. This will trigger communications to warehouses, driver shift planners, truck servicing etc to make ready the truck for its journey. During this phase changes to the truck schedule are undesirable because there would be knock on effects for the warehouse, driver assignment, etc.

The execute stage starts with the driver performing his pre-journey checks and continues until his debriefing at the end of his shift is completed. During this phase a high level of sophistication would be needed to alter the truck schedule in transit.

Magenta i-Scheduler supports PCX by allowing commit of individual trucks from a rolling schedule.

To achieve competitive advantage schedulers must take into account real-time economy where decisions are based not on some average rule but on the detailed analysis of the current situation. For example, a truck loaded by only 10% with a special cargo may be very profitable whilst a rule-based scheduler would not allow a nearly empty truck to start a journey. The requirement is to assess the economy of each truck, each journey, etc., which implies using the activity-based cost model. The current generation of batch schedulers cannot satisfy these requirements; a fundamentally new approach to the task of allocating resources in real-time is therefore needed.

Designing a scheduler that can cope with such a variety of operating conditions, handle uncertainty related to the occurrence of events and at the same time continuously produce schedules that maximise the specified value (or minimise transportation costs) is a real intellectual challenge. To the best of our knowledge such schedulers have not been proposed in the literature or implemented in practice.

2. SOLUTION

A complex problem can be solved effectively only with a complex tool. The tool must be capable of providing the "requisite variety" of behaviours, as postulated many years ago by Ashby [1]. Only the tools based on distributed (rather than centralised) decision making fully satisfy this criterion.

The scheduling tool described in this paper handles complexity by balancing and resolving conflicts of interests of many active players rather than by following given rules and satisfying specified constraints.

This is achieved primarily by assigning an autonomous agent to every player in the transportation system and tasking agents to obtain the best possible deals for their clients. Players here include the transportation enterprise as a whole (represented by the Enterprise Agent) and all individual transportation demands and resources (including drivers and resource owners). Demands

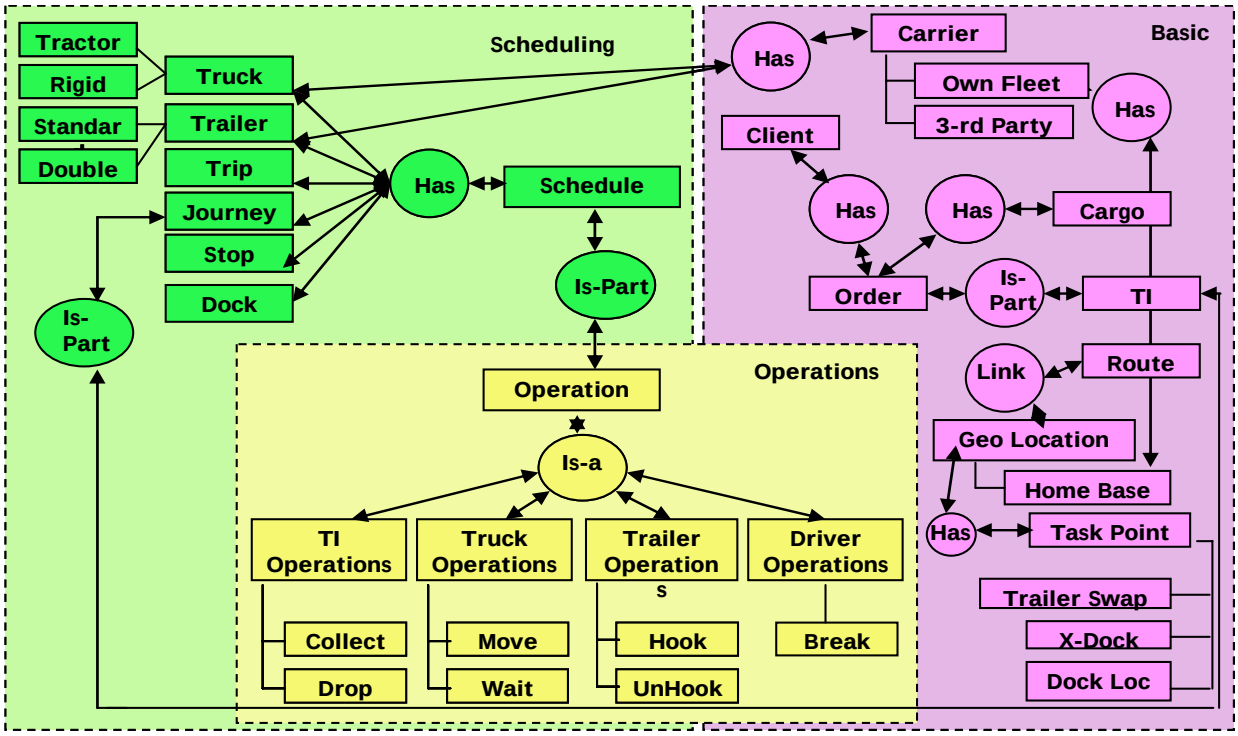


Fig.2. i-Scheduler Ontology Architecture

and resources that have common interests self-organise into groups represented by a single agent. Agents may decide to compete or co-operate depending on prevailing circumstances.

Human operators are provided with facilities for monitoring the scheduling process and overruling the decisions made by agents. Magenta i-Scheduler supports users in their interaction with the system by providing them with options and evaluating consequences of their decisions.

Conceptual knowledge on scheduling is separated from the resource allocation mechanism, which greatly simplifies updates and increases the reuse of code.

The process of formalising domain knowledge helps in refining it and closing the gaps left due to the empirical nature of knowledge collection.

To construct a schedule Magenta i-Scheduler uses a formal description of business domain knowledge (Ontology), formal description of particular situations (a Scene), formal description of agent goals (e.g. to increase profit, maximise service level, decrease risk, etc) and real-world events. Formal descriptions can be stored, changed and examined by decision maker at any time.

Agents working in the system are goal-driven. They do not perform combinatorial search and instead negotiate to achieve the goal, which allows overcoming the problem of the very large solution space. Each agent searches only for the solution which is the best for its client, for example, to occupy a particular time slot. If this option is already taken by another agent, the conflict of interest is identified and resolved by negotiation aimed at maximising the specified enterprise value or minimising transportation costs.

Agents are knowledge-driven. They use a Scene for analysis of the situation in the business network represented by the Scene with a view to improving quality and performance of the scheduling process. Scenes can be constructed and analysed dynamically by agents or human operators, forming the basis for Continuous Planning and Execution.

Scenes are the basis for a sophisticated interaction with users. The user is informed about the current situation by the current Scene and is given support for decision making. The fact that a Scene is a formalised description of a business situation helps to recognise patterns and identify gaps in the user knowledge and understanding.

Finally, the Road Transportation i-Scheduler can work in conjunction with Magenta knowledge discovery tools, with a view to discovering useful patterns of behaviour by means of clustering (finding groups of records with similar properties) which amounts to learning from experience.

3. ARCHITECTURE

The Road Transportation i-Scheduler is based on the standard Magenta software architecture as depicted in Figure 1 and described in [2] and [3]. The key features of Magenta technology are protected [4] and [5]. The earlier version was reported in [6].

The software is powered by a Multi-Agent Engine which contains all runtime tools necessary for the agent operation. Formalised knowledge is in Ontology supported by a powerful ontology

editor. Agents negotiate scheduling decisions in the Virtual Market.

The Logistics i-Scheduler is implemented on a J2EE platform.

4. ONTOLOGY

Ontology is constructed as a network whose nodes are objects (order, transportation instruction, cargo, truck, etc) and links are relations (Journey consist of TI, etc). It contains also attributes of objects like truck capacity, etc.

As depicted in Figure 2, Ontology is divided into three main parts. Basic part of ontology includes main logistics concepts. Schedule Ontology consists of schedule owners and schedules and contains references to the operations which constitute schedules, described in operation ontology. This structure allows the introduction of any type of operations into schedules.

A typical Scene is shown in Figure 3. The scene representation is design to help the user to understand the network situation. Looking at "Client" it is possible to find out where all TIs are at any given moment. Looking at "Location" it is possible to see all Journeys that are planned to go through this location.

Scenes also make decision making logic more visual and understandable to the end-users.

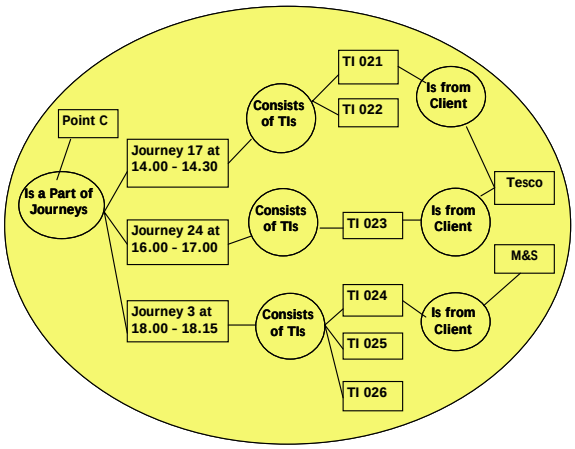


Fig.3. An Example of Scene

5. VIRTUAL MARKET

Virtual Market is a virtual environment in which agents negotiate the allocation of transportation resources to orders.

5.1 Agent Types

The Logistics i-Scheduler has several key types of Agents, including:

- Customer Agent – looks after the required Customer Service Level
- Order Agent – splits orders into transportation instructions and monitors its Key Performance Indicators (KPIs). If the allocation of a Transportation Instruction (TI) is unsatisfactory the Agent will try to improve it by searching for the most appropriate journey
- Transportation Instruction Agent – searches for the best journeys
- Journey Agent – looks for the best TIs, devises good consolidation, looks for the best routes with minimal mileage
- Cross-dock Agent – supports the schedule for cross-docking and allocates Journeys to time slots
- Group Agent – keeps similar TIs together
- Enterprise Agent – minimises fixed costs by decreasing number of trucks used; switches company strategies; monitors schedules and causes disturbances to destroy 'bad' journeys

Each type can act as a Demand Agent or a Resource Agent depending on the situation.

5.2 Generic Virtual Market Architecture (Agent Communities)

Magenta i-Scheduler has a generic virtual market architecture, which enables agents to form natural coalitions, e.g. TIs on a Journey or Journeys which together constitute a Driver Shift (all journeys made by a driver during his working day). This means that virtual market is organised in many levels, with nested groups formed by common interests where each group has an agent representative (Group Agent) who monitors what is happening in the group. Agents can join the group and leave it. The whole group can move (e.g. Journey goes to another Driver Shift).

This architecture provides flexibility in developing communities of different types of agents which can work as coalitions. Such approach also gives more options to find a right balance between interests of individual egoistic agents (TI, journey, etc) and interests of communities of agents (fleet, journey, etc). Also, this balance can be changed dynamically in reaction to events (X-Dock under terrorist attack). Finally, this architecture forms the basis for ongoing self-improvement of schedules.

The generic architecture provides stable, well-designed and elegant basis for rapid development of new applications in a wide range of new domains (supply chain, factories, containers, projects, finance, nurses, conference boxes, etc).

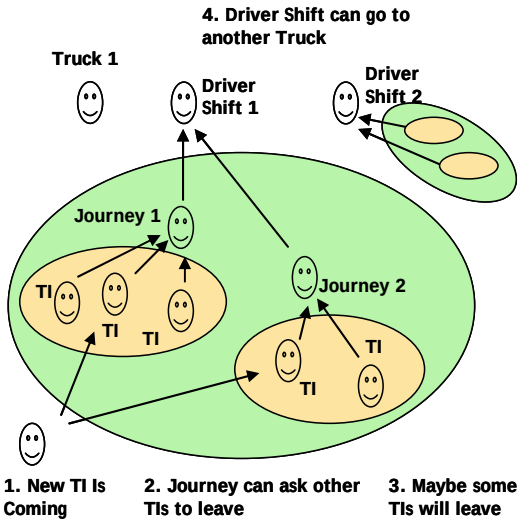


Fig.4. Multi-Level Virtual Market Architecture

5.3 Agent Negotiation Strategies and Protocols

Every agent is given the task of obtaining the best possible match for its client and a right to drop a previously agreed match if a better opportunity presents itself as events continue occurring. The abandoned partner pro-actively seeks a new match.

Agents on the Virtual Market use enhanced Contract Net negotiations protocol. Within Contract Net protocol agents pass the following iterations:

- Ask for pre-matching (rough) estimation
- Ask for plan, if possible to allocate without drop
- Ask for complete plan (drop is possible)
- Accept proposal
- Receive notification that plan is applied/failed

At each stage agents can cut or prioritize negotiation branches, thus detecting uninteresting options with fewer calculations.

If a plan fails, the agent can re-send a corrected proposal.

If an agent has two equal options, one of which involves drop and another is without drop – it will prefer the option without drop, as it saves time.

The protocol is decomposed into two parts: the contractor role and coordinator role. The contractor role is responsible for (and fully encapsulates) message exchanges between a particular pair of agents representing demand and resource.

Coordinator role contains decision-making logic of an agent at the protocol level and allows the demand agent to interact with several resources simultaneously (demand agent's coordinator role) and resource agent – with several demands (resource agent's coordinator role).

Magenta agents have multiple objectives, or criteria (quality, price, time, risk) each with a weight coefficient. This can be changed by the user for each individual agent. At any moment of time the user can increase the relative importance of the service level or maximise profit criterion. If for example orders from VIP customers are allocated and a new order comes from an even more important customer with tighter conditions, this may result in decreasing the service level for already allocated orders within the allowable interval. Changes in weight coefficients represent the change of agent strategy of working in the virtual market.

5.4 Microeconomics

The aim of microeconomics in Magenta multi-agent systems is to ensure that at any moment of time each agent knows which journeys are profitable for the company and which are not. There is a possibility to take into account not only "physical costs" but also the cost of planning. For this, the following principles are introduced. Each order finds out the cost of its transportation using price list and finds the most profitable journey. The profit received from transportation is shared between TI, Journey and Truck Agents. The microeconomics includes a system of taxation aimed to further improve scheduling efficiency. It does this by applying a penalty (tax) for making changes to the schedule close to Commit time. This ensures that only changes giving significant benefits are made as planning time runs out.

6. HOW THE SCHEDULING WORKS

The Initial Scene, which is a model of the transportation system at the beginning of scheduling, is constructed by an operator using a user-friendly editor or imported from an XML file. The Scene depicts all transportation resources (trucks, trailers, depots, warehouses, loading points, delivery points) and their locations on the transportation network. Goals, criteria, objectives (KPI), strategies and preferences are specified.

As the first order arrives, agents split the order into transportation instructions and allocate transportation resources to each TI. The Initial Scene is thus modified into the Current Scene, which is being updated after every event. The Current Schedule is extracted from the Current Scene and can take the form of a table or a diagram.

New events are added to the current schedule. They can be imported from third-party systems, from XML-files or added manually. Events are assigned individual objectives and strategies. When new events are entered into the system, the system stops other activities (or delays the incoming event) and starts TI allocation.

If events are entered in a batch, prioritization is carried out. Outstanding TIs can be prioritized with the help of prioritization logic defined in ontology or manually.

Next, new TIs are planned incrementally. TI agents pre-match by making rough option estimation and checking constraints. Next, they start negotiation with Journeys telling them the amount they are ready to pay for allocation. If TI is interesting for the Journey, it tries to allocate this TI using Open Slot, Shift or Drop algorithms. If an option is found, Journey makes the final option estimation and checks constraints. Journey sends message to TI to confirm allocation and informs other TIs belonging to it that new TI was allocated and they can re-calculate their values and costs.

If there are no new events, unsatisfied TIs, Journeys, Driver Shifts and Trucks could start pro-active improvement of their allocation. An important advantage of Magenta software is that this pro-

active improvement takes place in the system perpetually – i-Scheduler continuously works towards the best possible schedule. Any type of agent can be pro-active and agents who are unsatisfied with their current allocation are most likely to get pro-active.

User can review results using a Gantt chart or a map and analyse reports. There is a possibility to manually adjust the schedule asking the system to suggest alternatives. The system can also highlight parts of the schedule that needs improving.

Each Journey has its individual commit time. As commit time approaches, the agents get more and more active to utilise the last chance to find a better option. After the commit time of a Journey (e.g. 3 hours before the driver sets off) no further changes are made to this Journey schedule – Journey is passed to commit stage. In future versions, in order to improve versatility of the system, it will be possible to carry out additional scheduling at commit stage.

The incremental nature of Magenta scheduling and the use of different planning strategies means that the i-Scheduler will always produce a plan for an order, even if it is entered into the system very shortly before its commit time.

7. USER INTERFACES

The main window of the Logistics i-Scheduler, as shown in Figure 5, allows importing orders and placing them in the queue, starting planning and reviewing the schedule using a Gantt chart (Figure 6) and map, including detailed information about journeys and KPIs. Different parts of the window can be easily configured and re-arranged.

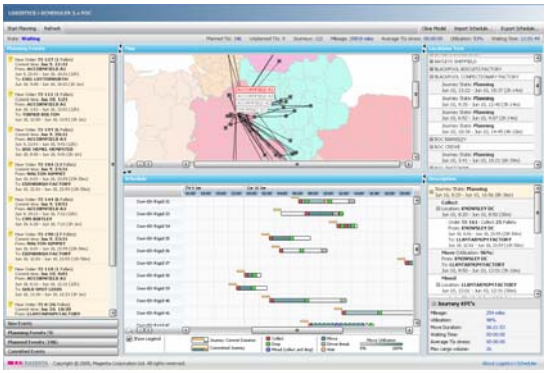


Fig. 5: Desktop Interface with Easily Configured Dashboard

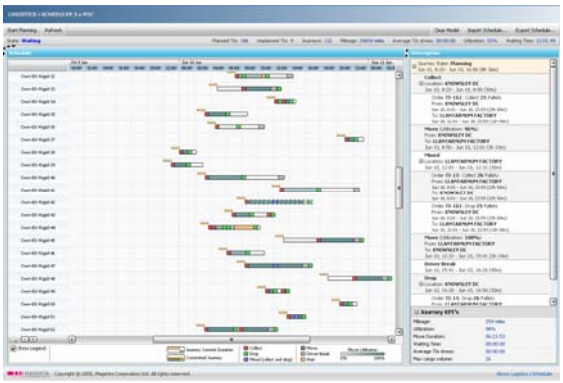


Fig. 6: Schedule Gantt Chart

At any moment user can view agent inspector (Figure 7) showing which agents are currently working in the system and analyse a negotiations log (Figure 8).

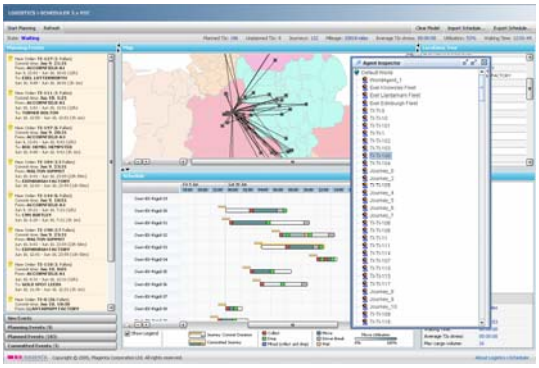


Fig. 7: Agent Inspector

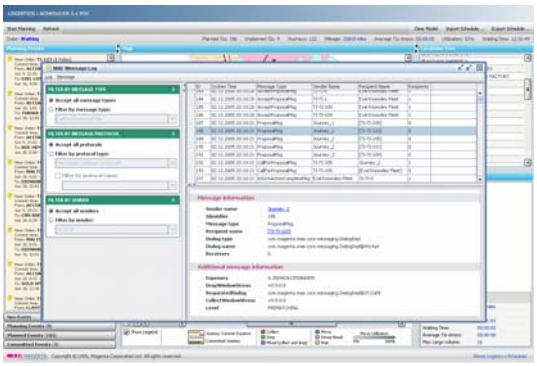


Fig. 8: Agent Negotiation Log

8. TESTS WITH CLIENT DATA

The first version of the Road Transportation i-Scheduler was tested on two sets of real-time data obtained from Client A, a 3rd Party Logistics Provider, and Client B, a provider of logistics and freight management services.

8.1 Client A Test Results

Client A requirements were to create transportation schedules in real time for 200 transportation instructions and 50 trucks (36 own fleet trucks plus 15 third-party carriers) operating on the UK Business Network. The network included 9 Distribution Centres/factories, cross-dock points for primary/secondary moves consolidation, 3 truck bases doing shared operations. Additional requirements were handling transportation instruction availability windows, backhaul, consolidation, vehicle capacity availability windows and constraint stressing.

The Client A currently employs two operators who require four to five hours to produce the schedule. Because of the current slow scheduling process, the Client plans two days in advance of execution. The Client also experimented with conventional batch schedulers and found that they could schedule only 70% of journeys correctly. The 30% of journeys had to be re-scheduled manually.

The Magenta i-Scheduler has completed scheduling of 200 transportation instructions in 8 minutes, planning 116 journeys of a total of 20790 miles. The result exceeded expectations with 95% of journeys scheduled correctly without any manual rework. The remaining 5% could not be scheduled during the tests because of incomplete information. Using Magenta scheduler Client A could complete planning and execution in the same day.

8.2 Client B Test Results

Client B requirements were to create transportation schedules for 4000 transportation instructions and 200 trucks operating on the UK business network. The network included primary and secondary deliveries between about 600 locations, 3 cross docks, 4 secure trailer swap locations and other types of locations. The network was also characterized by considerable number of very small orders. Special requirements included dynamic routing, cross-docking, handling location availability windows and driver breaks.

It took Magenta i-Scheduler about 4 hours to build a schedule for 4000 orders and 200 trucks with dynamical routing through 3 cross docks. This schedule showed an effective consolidation of small orders onto trucks. It was capable of incrementally planning new orders in near real time (a few seconds was required for each new order). As far as we know, this has not been achieved by any other transportation scheduling system. Before testing Magenta software the Client has spent considerable resources searching for a scheduler capable of handling their requirements and found none.

8.3 Performance Tests

During the tests on client data Magenta conducted an investigation into the effect of the number of orders on scheduler performance. As Fig. 9 shows, in situations in which a new order arrived when previous orders have been allocated, the time required to complete the schedule is, approximately, linearly dependent on the number of orders.

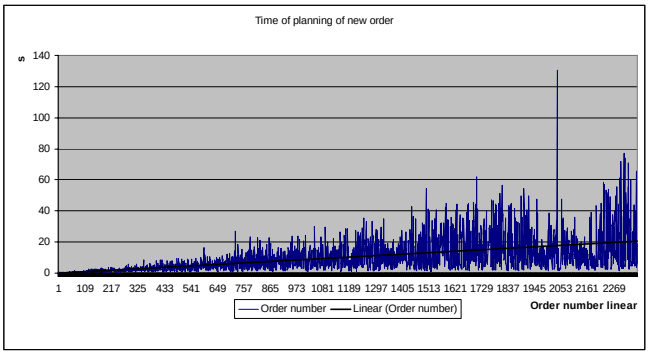


Fig. 9: The Effect of the Number of Previously Planned Orders on Scheduling Time for a New Order

This result is highly significant. It shows that Magenta schedulers can be easily scaled up to handle any size of order allocation.

A close inspection of the diagram shows that although the average increase in time is linearly dependent on the number of orders there exist significant number of low points and picks. Low points signify a situation where the new order was allocated with a minimum of re-scheduling, and picks are present where the arrival of a new order required a considerable re-allocation of previously assigned resources.

A very strong advantage of Magenta i-Scheduler is that planning is very fast and therefore can be done in real time. The rapid scheduling is achieved by the use of the incremental scheduling process, as opposed to the batch mode. Incremental planning gives an additional benefit of re-scheduling only those parts of the schedule that are affected by the occurrence of a new event, without having to completely destroy the schedule and re-build it "from scratch".

9. CONCLUSIONS

The test results described in Section 8 of this paper show that Magenta multi-agent technology, in general, and Magenta Road Transportation i-Scheduler, in particular, are capable of significantly improving scheduling effectiveness in comparison with conventional rule-based centralised schedulers. The difference is particularly visible in cases where there is a need to match a very large number and wide variety of orders and resources in real time.

The key features that contributed to the effectiveness of scheduling are summarised below:

- The distributed nature of decision making, which enables the swarm of agents to achieve a common goal whilst each agent member of the swarm is allowed to pursue the achievement of its own goal and yet be ready to negotiate when a conflict of interests is discovered.
- The ability to incrementally react to the occurrence of a new event, such as the arrival of a new order, by re-scheduling only affected parts of the previously agreed schedule.
- Pro-activity of agents, which drives them to improve the previously agreed allocations whenever practical.

10. REFERENCES

[1] Ashby, W.R. An Introduction to Cybernetics. Chapman & Hall, 1956.

[2] Rzevski, G., Skobelev, P. Magenta Technology: Software Platform v2.1 – Magenta Technology Whitepaper, July 2005

[3] Wooldridge, M. Responding to Real-World Complexity: Introduction to Multi-Agent Technology – Magenta Technology Whitepaper, August 2004.

[4] Agent Method and Computer System For Negotiating in a Virtual Environment, Patent Reference No: WO 03/067432 A1. Published 14.08.2003

[5] Data Mining. Patent Reference No: GB 2 411 015 A. Published 17.08.2005

[6] Himoff, J., Skobelev, P., Wooldridge, M. MAGENTA Technology: Multi-Agent Systems for Industrial Logistics – Magenta Technology, August 2004.

Commercializing a Multiagent-Supported Collaborative System

Leen-Kiat Soh and Hong Jiang
I-MINDS, LLC
Computer Science and Engineering
University of Nebraska-Lincoln
256 Avery Hall, Lincoln, NE 66588-0115
1-402-472-6738
{lksoh, jiang}@cse.unl.edu

ABSTRACT
In this paper, we describe our effort and lessons learned in commercializing an innovative infrastructure to support student participation and collaboration and help the instructor manage large or distance classrooms using multiagent system intelligence. The system, called I-MINDS, has a host of intelligent agents for each classroom: a teacher agent ranks and categorizes real-time questions from the students and collects statistics on student participation, a number of group agents that each maintains a collaborative group and facilitate student discussions, and a student agent for each student that profiles a student and finds compatible students to form the student's "buddy group". Our commercialization plan is three-fold: (1) to strengthen our understanding of potential customers, (2) to understand our competitors and alternative products, and (3) to examine and estimate the market potential for I-MINDS. To accomplish the above, we have conducted experiments to further investigate the impact of I-MINDS in actual use, comprehensive research on existing products and technologies, secure legal protection of our technology, plug-and-play I-MINDS for various demonstrations to companies for joint research and development.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *multiagent systems, intelligent agents.*

General Terms
Algorithms, Design, Experimentation.

Keywords
Multiagent, Cooperative Learning, Distance Education.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005…$5.00.

1. INTRODUCTION
In December 2005, we founded a start-up company called I-MINDS, LLC, with the intention of commercializing a multiagent system called I-MINDS. Subsequently, we received funding from the National Science Foundation's Small Business and Innovative Research (SBIR) program to further improve the system technically and commercially. This paper describes our effort this past year and the lessons learned.

I-MINDS is a computer-supported cooperative learning (CSCL) system. There is need for CSCL systems especially in large or distance classrooms. Students in large or distance classrooms often do not enjoy the same level and quality of student-to-student and student-to-instructor interactions during the class (synchronously) or after the class (asynchronously) as in smaller classrooms. Such interactions are, however, important to foster group-based learning and student affinity in a class. To improve such interactions, educators have turned to technology such as personal response systems, online chat rooms, teleconferencing tools, and computer-supported cooperative learning (CSCL) environments. However, these systems do not adapt to diverse student and instructor needs and behaviors. These systems are passive participants in the process and students and instructors often resort to other modes of interactions (e-mail, phone, or face-to-face meetings) in order to address individual problems. Further, the advent of Internet and multimedia technology has meant potentially drastic changes in the teaching and learning process from the traditional classroom setting to a more geographically distributed, virtual but still interactive one. Hence, there are both motivation and opportunity to develop intelligent CSCL systems.

The long-term goal of our technical and commercial effort is to ultimately build systems to support intelligent virtual classrooms where students enjoy peer interactions as well as on-site, small classroom students, and where students enjoy student-instructor instructions as well as on-site, small classroom students. A secondary goal is to extend the virtual classrooms concept to other group-based applications such as decision support, focus group studies, and corporate training, and other cooperative work where agents might play a role [14], since the technologies and features of a CSCL system could be of significant use in these applications. For example, agents that support collaboration between two students can be used to mediate the discussions between two focus group participants.

Our commercialization effort for started with three main objectives to study the commercial feasibility of the I-MINDS technology.

Objective 1. To strengthen our understanding of potential customers. To assure that the technical design of the I-MINDS system meets education needs of teachers and students and successfully reach customers in the later-phase commercialization, we have researched to gain an in-depth understanding of potential customers from the following perspectives: (a) needs of teachers and students in real-time virtual classroom environment, (b) the current market offerings and how I-MINDS can fill the gap to fulfill unsatisfied customer needs, and (c) the decision-making processes of purchasing and implementing technology improvement in post-secondary schools in both academia and industry.

Objective 2. To understand competitors and alternative products. An effective evaluation of our competitors and alternative solutions will help us understand the online educational market. With this evaluation, we will be able to identify important issues during the commercialization phase, such as the I-MINDS business model, pricing strategy, competitive advantages and potential entry barriers.

Objective 3. To examine and estimate the market potential for I-MINDS. Further, we have investigated the current situation for overall education and training industry and our specific targeted market. Based on our analysis of the information collected about customer, market and competitors, we will be able to identify the key success factors in the industry which will help commercialize I-MINDS more effectively.

In the following, we first describe I-MINDS and its capabilities in Section 2. Section 3 discusses specific steps that we have taken towards achieving our three aforementioned objectives and more. Section 4 presents the lessons learned. Finally, we conclude.

2. I-MINDS

Our innovation is an infrastructure called the Intelligent Multi-agent Infrastructure for Distributed Systems in Education (or I-MINDS). I-MINDS consists of different intelligent agents working together to actively support student-student and student-instructor interactions. Specifically, there are agents that rank and categorize questions, and profile students to help the instructor manage a classroom in real-time; agents that find compatible agents to form "buddy groups" for students; and agents that form and maintain structured cooperative learning groups.

We have pilot tested I-MINDS and further deployed the software in the laboratory of an introductory computer science course at the Computer Science and Engineering Department of the University of Nebraska. Tests based on student assessments and control-treatment groups showed that I-MINDS, though still needing better graphical user interfaces and further development, could support and improve student performance and instruction in several pedagogical aspects.

2.1 Architecture

I-MINDS is built using a loosely-coupled layered architecture [7], which makes I-MINDS flexible and usable in heterogeneous environments.

The bottom layer (i.e., the network layer) in I-MINDS provides the basic communication functionalities by using sockets. Since sockets are available in a variety of platforms across the network, this layer allows I-MINDS to communicate in heterogeneous situations.

The second layer provides system-level protocols and encapsulations with necessary abstraction to provide convenient communication and deployment functions to the upper layers. This level includes a Relational Database Management System (RDBMS)-type database (MySQL) and an audio/video server (Macromedia Flash© Communication Server). The database is used by the agents for fast storage and retrieval of information and the audio/video server helps establish audible and visual communication between the teacher and the students.

Finally the topmost two layers of I-MINDS contain the intelligent agents i.e., teacher agents, student agents, and group agents. Each intelligent agent has two sets of modules: content dependent and content independent. The content independent set provides the definitions and processes for general education-related services, while the content–dependent module handles specific course-related information and knowledge base, providing the required data and the heuristics used to gather, analyze, disseminate and process the generated data.

2.2 AGENTS and TOPOLOGY

Figure 1 shows an example of the topological infrastructure of I-MINDS [8]. The manager manages the ongoing classroom sessions. It manages system level information such as the list of ongoing classes, list of courses and teachers, the login names and passwords for the students for each of the classrooms, etc. I-MINDS is capable of holding multiple concurrent classroom sessions. In each I-MINDS classroom session, the teacher, students and student groups are assigned the teacher agent, student agents and group agents, respectively. These agents support the person/group that they serve.

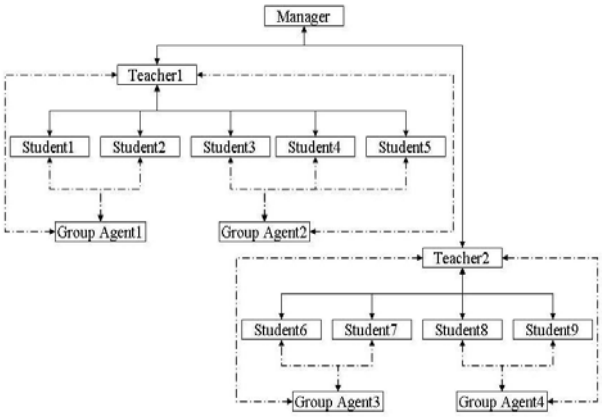


Figure 1: I-MINDS Agents and Topology

Intelligent agents are autonomous and can operate robustly in rapidly changing, unpredictable, or open environments. With these intelligent agents serving and catering to students' unique needs and behaviors, students will be able to participate in a computer-supported cooperative learning (CSCL) environment actively rather than listening to the lectures passively as in a traditional large or distance classroom. Currently, I-MINDS has three

types of intelligent agents: (1) teacher agents, (2) student agents, and (3) group agents.

2.2.1 Teacher Agent

A teacher agent, interacting with a teacher, is responsible for disseminating information streams to student agents, maintaining profiles for all students, assessing the progress and participation of different students, ranking and filtering of the questions asked by the students, and managing the progress of a classroom session. A teacher agent mainly interacts with an instructor. It also sends instructions to the group and student agents.

Question ranking is a key feature that allows the teacher agent to manage large or distance classrooms for the instructor. When questions are asked, the teacher agent scores each question based on a set of keywords and heuristics. Keywords are subject-topic specific and weighted. Heuristics are course-specific. For example, a question asked by a student with a high-score profile will be given a high score; a question asked by a student who has never asked a question before will also be given a high score to encourage the student to ask more questions in the future; and a "why?" question is scored higher than a "what?" question. After scoring, the questions are ranked and displayed to the instructor. The instructor may choose to answer or discard a question. This is how the instructor's actions trigger the reinforcement learning mechanism of the module. When the instructor chooses a lowly-ranked question over a highly-ranked question to answer, the teacher agent takes this as an indication that it has ranked the question incorrectly. It thus traces back to the heuristics and keywords that have contributed to the high score of the question and lower their weights; likewise, it increases the weights of the heuristics and keywords that have scored the question low in the first place. Thus, by answering and discarding ranked questions, the instructor implicitly teaches the teacher agent how to better score and rank the questions. Details of this module can be found in [9].

Question grouping is another feature that allows the teacher agent to address multiple similar questions together and profile students who ask similar questions. Future applications of this feature include automatically generated and answered Frequently Asked Questions (FAQs). Question grouping is based on question classification and keyword matching. To achieve question classification, we adopt the *utterance classification* approach in the AutoTutor [3]. We use the ApplePie parser [11] and the utterance classifier program of AutoTutor [10] to classify a question into one of twelve classes such as Contribution, Discovery, and so on. This classification feature also allows the group agent to monitor the roles of group members in group discussions. Two questions are also considered similar if they share a high percentage of keywords. Details of this module can be found in [6].

A student is profiled by I-MINDS' agents in several ways. First, student-instructor interactions are profiled mainly based on the quality of questions that a student asks. Based on the grouping of these questions, students can also be profiled relative to others in terms of compatibility.

2.2.2 Student Agent

In an I-MINDS supported classroom, a student agent serves a unique student, and, similar to a teacher agent, it also has a GUI front-end. A student agent primarily interacts with a student. It also exchanges information with the teacher agent and the group agents.

The student agent first allows for student login and course registration through communicating with the manager and the teacher agent, respectively. The student agent then acts as a medium: displaying the instructional materials received from the teacher agent to the student and relaying information from the student to the teacher agent.

When a student asks a question or performs some kind of collaborative activities (e.g., initiates a forum discussion or participates in sketching out ideas), his or her profile is changed and subsequently updated with the teacher agent and other student agents. The change might trigger other student agents to recruit the student to join their buddy groups. Thus, a buddy group is dynamic. Further, if a student is not responsive to a buddy group that he or she is a member of, he or she can be dropped from the group. Hence, more responsive students will be approached more frequently than the others. The student agent also adjusts its heuristic rules according to the current classroom environment. For example, in a course in which students are very active in collaborative learning in every classroom session, a buddy who has not been active for the entire session may be dropped. Thus, the student agent keeps a profile of its student's buddies in terms of their responsiveness in collaborative activities (forums and digital whiteboard). These include the number of messages sent, type of messages, frequency of messages being sent, time spent on "idea sketching", the number of times a student is dropped from a buddy group, and so on.

2.2.3 Group Agent

In I-MINDS, a group agent is activated when there are structured cooperative learning activities. Structured cooperative learning involves specified activities that explicitly require students to cooperate. Since a group agent works behind the scene, it does not have a GUI front-end.

To support structured cooperative learning, the content-independent module set is capable of forming groups (or coalitions) of students based on their profiles, evaluating student perception of teamwork and other members through online surveys, tracking student participation, and improving its heuristics in monitoring and forming better groups.

To support the Jigsaw learning model [1, 2, 4], we have designed an auction-based learning-enabled coalition formation algorithm called VALCAM that combines automatically tracked collaboration activities with subjective peer-based evaluations of a student's performance in structured cooperative learning.

In VALCAM, the teacher agent acts as a coordinator or the coalition formation process and makes global decisions, such as, what should the least number of members in a coalition be, how long should the coalitions last, how the performance of each coalition should be evaluated, etc. The group agents then manage the coalitions. Each group agent monitors the performance and activities of the members of its assigned group. After the coalition has completed their tasks, the group agent also evaluates the performance of each student agent as a group member.

VALCAM algorithm is based on reinforcement learning of the student agents. In the beginning of every session the teacher agent chooses a few students and then initiates the group formation.

During the group formation process, each student agent bids—an iterative Vickrey auction—to join its favorite group based on virtual currency earned from previous Jigsaw activities. Once the groups are formed, the student agents work with their group members to complete tasks assigned by the instructor. Finally, when the assigned task is completed, the student agents are rewarded with virtual currency based on their individual performance and their performance as a group member through surveys of their peers' evaluation of the student. Students who performed well will receive more virtual currency, allowing them to more successfully bid for their favorite groups in future sessions. Details of this algorithm can be found in [5].

3. COMMERCIALIZATION EFFORT

During the past year, we have taken several steps towards commercializing the I-MINDS system and its technology, in accordance to our three feasibility objectives. Specifically, we have conducted the following.

First, we have worked with our university to protect the intellectual property of our technology. This has involved identifying what we want to market as our product since we want to protect the product as well.

Second, we have continued to improve the I-MINDS technology and product. Based on our deployment of I-MINDS in actual classrooms, we have identified features that would improve I-MINDS' user-friendliness and also identified agent-based technologies to better utilize agent intelligence to support a virtual classroom.

Third, we have continued to deploy I-MINDS and collect more data to evaluate I-MINDS' impact on student learning.

Fourth, we have continued to solicit additional funding and sponsorships. We succeeded in obtaining funding from Microsoft to port I-MINDS to the Conference XP Platform.

Fifth, we have worked with a consultant company to research the potential markets for I-MINDS. Under their guidance, we have conducted two comprehensive studies into the e-learning market and the focus group market.

Sixth, we have contacted several companies and demonstrated I-MINDS. For each demonstration, we have customized I-MINDS to fit the individual company's application needs.

3.1 Intellectual Property

Protecting the intellectual property of our technology is of paramount importance. Identifying what to protect is also critical. Since the company executives are all associated with the University of Nebraska, and since the I-MINDS was developed using university resources and on university properties, our agreement with the university is as follows. The university has the rights to the technology. However, the university gives us the exclusive rights to commercialize the I-MINDS technology developed thus far. Further I-MINDS technologies developed using non-university funds and off the university properties, as a result, will belong to our company.

3.1.1 What Is Our Product?

Identifying the specific innovation to protect and the type of protection is critical. After numerous discussions with the University's technology transfer office and legal experts, there are several choices: (1) the entire I-MINDS as a software product, (2) the I-MINDS technology as an enabling technology, and (3) the I-MINDS technology as a data collection service. The first choice basically sells I-MINDS as a standalone, packaged software product. Choosing this option implies that our business plan would be to create different versions of end-to-end, full-feature I-MINDS to suit different needs such that I-MINDS can be used with plugging it into other existing software modules that our customers might already have. This choice is less desirable as it would impose a heavy product development cost on our startup company, which we cannot afford.

The second choice is to provide technical expertise to companies that would like to incorporate I-MINDS technologies into their existing software modules. Here, the unique I-MINDS technologies are the three groups of agents, equipped with machine learning capabilities, and the multiagent system technology. Choosing this option implies that our business plan would be to embed intelligent agent and multiagent technologies into our customer' existing software and to enable computer-supported collaborative environments. This choice is more attractive to us as it would allow our company to grow in steps that are more manageable.

The third choice is to help our customers build agents to support virtual collaborative environments to collect, analyze, and report data for them. Choosing this option implies that our business plan would be to promote the autonomous capability of these agents in observing their environments, making decisions, actuating those decisions, and tracking every action and outcome, and how one could gain insights from looking at how users interact in a collaborative environment. This choice is less attractive than the second choice as it would probably detract our company from what we initially set out to do. That is, our long-term goal is to research and develop intelligent computer-supported collaborative systems. However, we recognize that this third choice might be a feasible as an intermediary one to make our first baby step into the commercial world.

3.1.2 What Kind of Protection?

There are several types of protection. We are concerned with formal patent protections and various software license agreements. Filing a formal patent protection has its pros and cons. In terms of advantages, it protects the product or technology (as described in the first two choices as described in Section 3.1.1) strongly. It also gives us the satisfaction of being recognized as inventors. However, the patent application process is expensive and lengthy. While the process is taking place, it would also be not convenient for us to publish related research papers on the technology. On the other hand, software license agreements are more flexible and can be structured at a per customer basis. Other contractual agreements can also come into play. As a result, as we develop more and more features and technologies, we do not have to keep filing for patent protections.

Presently, we have duly filed for a provisional patent with the United States' Patent and Trademark Office. Meanwhile, when we demonstrate our products to other companies, we require both parts to sign a non-disclosure agreement.

3.2 Technical Improvement

Building a research prototype and developing a marketable product is different. For the latter, the product needs to be reliable, user-friendly, well-designed and well-suited for the customers. A reliable product does not crash and, if it does crash, should be

easily re-started without loss of data or incurring additional costs on the users, for example. Towards this end, we have continued to simplify how our product can be downloaded, installed, and executed, and to improve the look-and-feel of the graphical user interface. Some examples of improvement are as follows. Currently, our I-MINDS tool supports the Jigsaw cooperative learning model and also provides visual analysis of classroom data.

3.3 Deployment and Evaluation

Actual deployments and evaluations are important to our commercialization effort as we can (1) study how I-MINDS impact the users—in our particular situation, how students learn, and (2) study how to improve I-MINDS' look-and-feel to be a more polished product ready for the commercial market. Encouraging results have allowed us to pitch I-MINDS more convincingly to companies as we try to seek commercial partnerships in the past year.

3.3.1 Pilot Study

To determine the potential impact of I-MINDS on student learning, a pilot study [12] was conducted in May 2003 where the tool was used by subjects in a controlled experiment to assess what impact it had on student learning of Global Information Systems (GIS) content.

In the pilot study, the *control* group was conducted like a traditional classroom, with the instructor and the students in the same room. The *treatment* group was conducted as a distance classroom where the instructor and the students were in separate rooms and their only communication channel was through I-MINDS. Results from the two testing sessions were encouraging. Here we outline the results as reported previously in [12].

Although there was a slight difference in the means of the experiment and control groups for Test 2, this difference was not statistically significant ($p > .05$), and *the amount that the treatment group improved from the pretest to the posttest was nearly twice that of the control group.*

Comments from the university professor who used I-MINDS in teaching both of the content lessons were also encouraging. He indicated that the teaching tool was very easy to learn and use. He also said that the tool could enhance distance learning, especially by making it possible for building an archive of information that could be accessed "on-demand" by students. The instructor also noted that questions asked of him via I-MINDS tended to have higher quality, reflecting a deeper understanding, and demand a richer response than those questions posed during the control sessions.

3.3.2 Deployment in Actual Classrooms

In Spring 2005, we deployed I-MINDS in the laboratory of an introductory computer science course at the University of Nebraska. The course is CS1, the first core course for computer science and computer engineering majors in the Computer Science and Engineering Department. The course had three 1-hour weekly lectures and one 2-hour weekly laboratory sessions. In each lab session, students were given specific lab activities to experiment with Java and practice hands-on to solve CS problems. In Spring 2005, there were three different lab sections, each with about 20 students.

Our study utilized two sections. In the control section, students worked in Jigsaw cooperative learning groups without using I-MINDS. Students were allowed to move around in the room to join their Jigsaw groups to carry out face-to-face discussions. In the treatment section, students worked in Jigsaw cooperative learning groups using I-MINDS. Students were told to stay at their computers and were only allowed to communicate via I-MINDS. With this setup, we essentially simulated a distance classroom environment.

For each lab, the students were given a lab handout with a list of activities—thus, a lab is a task and its activities are the subtasks. We conducted the study for three weeks, covering topics in debugging and testing, Unified Modeling Language (UML), and recursion.

Table 1 shows the post-test scores of the three sessions for the "Control" (i.e., Jigsaw without I-MINDS) and "Treatment" (i.e., Jigsaw with I-MINDS) sections. The results indicate that students using I-MINDS for the Jigsaw activities were able to obtain comparable and even better post-test scores. We had initially hypothesized that the students in the control section would perform better than the students in the treatment section simply because the students in the former would have a chance to discuss face-to-face and were able to use paper and pencil to draw and share the same computer screen during the stages 2 and 3 of the Jigsaw process. On the other hand, I-MINDS, still lacking user-friendly GUI, had been expected to hinder such free, natural interactions among students, leading to ineffective collaborations. Thus, the result had not been expected. Students in the Treatment section performed better than the students in the Control section in the later two sessions. Further, students in the Treatment section also achieved better standard deviation—meaning that these students' post-test scores were more tightly clustered than those of the Control section. Upon closer analysis, we suspect that the act of typing and communicating through the forum and digital whiteboard of I-MINDS forced the students to articulate explicitly their thoughts and focused their attention to the tasks at hand. This in turn improved student understanding of the subject matter.

Table 1: Student performance for the Control (Jigsaw only) and Treatment (Jigsaw w. I-MINDS) sections

Peer Rating Questionnaire Results				
Week	**Control Section**		**Treatment Section**	
	Mean	**Std. Dev**	**Mean**	**Std. Dev**
1	42.02	2.68	35.60	5.97
2	39.06	7.05	33.30	8.65
3	39.90	4.80	38.59	6.34

The results were very encouraging: without face-to-face interactions, students carried out their I-MINDS-supported Jigsaw tasks and performed relatively well in individual post-tests.

Details of the results can be found in [13].

3.4 Other Sponsorships

We have also submitted proposals for additional funding and have received a grant from Microsoft Research to port I-MINDS technology into the Microsoft Conference XP Platform. This partner-

ship allows us to develop an I-MINDS product enabled by a far more reliable infrastructure in the Conference XP Platform than we could ever build on our own in the short term.

3.5 Market Research and Consultants

The National Science Foundation's Small Business and Innovative Research (SBIR) Program has retained Foresight Science and Technology, a high-tech consulting firm specializing in "moving technology out of people's minds and into practical applications," to help SBIR grantees devise a business plan (or commercialization plan). Working with Foresight, we have conducted two comprehensive market studies: (1) e-learning, and (2) focus groups. Here we report briefly on the market research studies.

3.5.1 E-Learning

Here, our *market drivers* (economic, social, or political) are (1) the consolidation of the e-learning market, (2) increased reliance on technology, particular the Internet, (3) increased number of online universities, and (4) decrease in funding in traditional post-secondary education. In terms of the *market size*, according to a report by Eduventures, the market for fully online distance learning will grow 38 percent over 2004, reaching revenues of more than $5 billion. According to a report from Frost & Sullivan, by 2010 the total market size for e-learning is estimated to be $12.18 billion. The education segment of that market is anticipated to be 28.5%, giving us a market for educational e-Learning that is roughly $3.5 billion. The corporate segment is anticipated to be 67%, giving us a market of approximately $8 billion. The government segment is anticipated to be 4.5%, giving us a market of approximately $500,000. In terms of *competitors*, there are numerous commercial, popular products such as eClassroom, Blackboard, Desier2Learn Suite, Centra, and so on, that support e-learning. Some also support cooperative learning, but none uses agent or multiagent intelligence. In terms of *potential patent infringements*, we searched the US Patent Office (USPTO) for both issued patents and patent applications with a set of relevant terms. We have identified several similar patents but are able to distinguish our technologies clearly from these patents. We have also looked into recent and *ongoing research and development* and identified a host of projects on e-learning software. For example, the University of Illinois at Urbana-Champaign presented a framework for mutual online concept learning for multiple agents in 2002, the University of South Carolina at Columbia has an NSF-funded project to develop multiagent technology for e-commerce and other applications which could have implications for the educational and training markets, and so on. In terms of *customers*, we look into both competitors and users. Competitors could be potential customers for joint research and development. Potential users are universities that provide online education. In terms of *promotion*, participating in conferences, workshops, and trade shows is one way. Further, having a good website (homepage) for our company, posting the results of our studies, and contacting educational organizations such as the Association for Educational Communications and Technology and the United States Distance Learning Association are also worthwhile promotion approaches.

3.5.2 Focus Groups

Here, our *market drivers* are (1) consumer behavior becoming much more difficult to understand using traditional focus group survey methods, (2) lack of training on how to conduct such surveys and to analyze such survey data, and (3) online focus groups save time and money. In terms of the market size, it was roughly $300 million in 2001. According to Gantry Group Newsletter (issue no. 3, October 2001), "the movement to online research is considered to be the largest paradigm shift in market research since the high penetration of telephones in the late '50s. Marketing research is currently a $6 - 7 billion dollar industry in the U.S., with online research making up just under $300 million. Larry Gold, editor of Inside Research, has predicted that this figure will double in each of the next several years." In terms of *competitors*, there are several products such as WebSurveyor, QualBoard, and SnapSurveys suite of survey software. In terms of *potential patent infringements*, we once again searched the US Patent Office (USPTO) for both issued patents and patent applications with a set of relevant terms. We have identified several similar patents but are able to distinguish our technologies clearly from these patents. We have also looked into recent and *ongoing research and development* and have not found specific projects that are related to supporting online focus groups. One relevant project is the Group Support Systems at the University of Western Australia. In terms of *customers*, once again, both competitors and users are our potential customers. In terms of *promotion*, associations of interest are the Council of American Survey Research Organizations, the Market Research Society, the American Marketing Association, etc.

3.6 Customized Demonstrations

In the past year, we have also demonstrated our product to three potential customers—three companies who could partner with us for joint research and development of the I-MINDS technology. For each demonstration, we first investigated about what the company wanted and what their application needs were, then we added customized features to I-MINDS, and finally tailored our demonstrations to the company's domain application. One example of such customizations is the use of vocabulary pertinent to each company on our software and in our presentation. Other customizations have included building special visual displays, adding new functionalities to I-MINDS, and setting up specific test scenarios for the demonstration. These have been time-consuming activities.

4. LESSONS LEARNED

In this section, we report on the lessons learned in our year-long commercialization effort of I-MINDS. In Section 3, we have outlined and described activities such as identifying our product and intellectual property protection, improving the technology, deploying and collecting results on the technology, soliciting additional funding, studying the markets, and presenting the technology with customized demonstrations.

Overall, the process has been educational and time-consuming. We have learned about the legal issues, market studies, promotion, and how to build a startup company from the ground up. Here we specifically talk about our experience in five aspects: (1) how to distinguish our technology or product and the notion of agents, (2) being innovative or "cutting edge" does not always translate into marketability, (3) how deciding on the intellectual property protection strategy help shape our focus, (4) the costs and benefits of customized demonstrations, and (5) the market research studies.

4.1 The Notion of "Agents"

Based on our market research studies, we identify the intelligent agents and the multiagent infrastructure as the distinguishing technology that I-MINDS has in the commercial world for online collaborative applications and systems. However, the use of agents is still not commercially mainstream in our e-learning and focus group domains. The awareness of agents or the appreciation of what agents and a multiagent system can do is still not widely apparent. Thus, in our contacts with potential companies, we have had to explain the notion of intelligent agents: what they can do individually, and what they can do as a team, and what they can learn to do overtime to adapt to their environments. We have also learned that the notion of intelligent agents is not practical without a context—specifically a context that the potential customers are familiar with, preferably a context that is tailored to their applications or domains. We have also learned that promoting intelligent agents is not always the right thing to do as how "these modules" can offer the potential customers what they need but do not have is more important.

Also, explaining the notion of intelligent agents and multiagent systems in layman's term is very important. We have learned that situating agents in a big picture, within the context of the potential customers' needs and applications, is key in cases where they appreciate the advantages of agent technology.

4.2 Curse of Cutting Edge

Due to the economic downturn, at least in the U.S., for the past few years, companies might not be willing to invest as much in the "research" portion of their research and development (R&D). Some companies look for applications that are almost ready to be plugged into their mainstay of software products; some companies look for applications that can be used off-the-shelf. We have had difficulty in locating companies that are willing to invest in joint R&D effort. Partly this is due to the nature of our technology and product. Being a software product, companies do think that if one can build such as product, then they can also build it. Thus, there is no point in joint R&D.

More importantly, if the technology is "cutting edge", then companies might not be willing to invest in joint R&D and wait for a few years for the technology to transfer into matured applications.

In a sense, we tend to believe that we are visionistic and see the potential impact of our technology and product in changing how people collaborate online. And we have not succeeded in convincing our potential customers of that vision.

4.3 Intellectual Property

Securing an intellectual property protection is of paramount importance in terms of protecting the revenue of the company. However, we have also learned that the process of identifying the product or technology that we want to protect and types of protection suitable could help shape what we really want to market. Thus, this discussion should take place early and reach a decision as soon as possible.

Based on our experience, initially, we had a vague idea of marketing I-MINDS as a standalone, packaged system. However, as we clarified on what we wanted to protect through our discussions with our technology transfer office and legal experts, we have come to realize that it is also the use of agents and the multiagent system framework that could be marketed as well as the expertise and experience that we have accumulated during the R&D of I-MINDS. These could be translated into enabling technologies or services, which might not be easy to patent but could be contracted or protected through other agreements.

4.4 Customized Demonstrations

Customized demonstrations have their pros and cons. The pros include (1) better, more application-specific demonstrations to our potential customers, and (2) continuous technical improvement of our product and added application of our technology. However, as alluded to earlier in Section 3.6, customizations are time consuming. Particularly, half-baked customizations do not help while genuine customizations require us to (1) interact and communicate with our potential customers to find out what they want to see, (2) find a midway point between what we currently have and what they want to see where it is feasible for us to build before the demonstration, and (3) customize the software product. During the customization, we also need to design the use cases or test scenarios so that we can, in a short demo, clearly highlight what our software can do.

A demonstration should also be captivating and pertinent to the potential customers right from the start. Unlike a presentation or demonstration at a research conference or workshop, these demonstrations are usually to company executives who might not understand or might prefer to not be concerned with the underlying technology (e.g., the agent technology in our case) but who might appreciate what they could see and point-and-click in real-time. Thus, it is important to show what the bottom line is right from the start of the demo: what and how can the technology or product benefits them in the short term in particular, and in the long run in general.

4.5 Market Research Studies

Based on our experience, we realize how valuable it is to have the opportunity to work with a consulting company for those who are not experienced in commercializing an innovation. Our first market study was about e-learning. Initially, we supplied the consulting company a set of documents describing our product and technology. The consulting company then did the research and gave us the outcome, which is summarized in Section 3.5.1. Given this document and experience, we conducted our second market study about focus groups on our own. From the first study, we learned about the online resources to obtain necessary information and the strategies and approaches to such a study. Key components to research include market drivers, potential patent infringements, ongoing projects, competitors, customers, and promotion. Strategies include investigating the above components with a keen eye on distinguishing the uniqueness of our product or technology.

A market research study is time consuming and may require exhaustive, investigative searches online as well as in the libraries. For our second market study, it took more than 40 person-hours.

5. CONCLUSION

We have described our effort and lessons learned in commercializing an innovative infrastructure to support student participation and collaboration and help the instructor manage large or distance classrooms using multiagent system intelligence. The system, called I-MINDS, uses a multiagent framework to support teachers

and students. Our commercialization plan has three objectives: (1) to strengthen our understanding of potential customers, (2) to understand our competitors and alternative products, and (3) to examine and estimate the market potential for I-MINDS. Guided by these three objectives, we have methodically carried out six primary sets of activities: (1) identifying the intellectual property protection and the product to be protected, (2) improving the I-MINDS product and technology, (3) deploying and evaluating I-MINDS, (4) securing other sponsorships, (5) conducting comprehensive market research studies, and (6) giving customized demonstrations. Based on our year-long effort, we have had some lessons learned. We have found out that the one thing—the intelligent agent technology—that distinguishes our product is also the one thing that is difficult to "sell". We have also found that a product that is cutting edge is not always commercially marketable. We have also learned that securing intellectual property protection should be done early, not only to protect our product and technology, but also to help shape the very "thing" that we want to market, that demonstrations to solicit joint R&D differ significantly from demonstrations in academia, and that for inexperienced people like us in devising a business plan, having a consulting company to work with is invaluably important.

6. ACKNOWLEDGMENTS

The I-MINDS project was first developed with a seed grant from the National Center for Information Technology in Education (NCITE). It was later sponsored partially through an NSF SBIR grant DMI-044129. The authors would like to thank Foresight Science and Technology in their help in market research. The authors also would like to acknowledge Art Zygielbaum, a vice president of I-MINDS, LLC, and our software developers and research students Xuli Liu, Nobel Khandaker, L.D. Miller, Xuesong Zhang, Phivas Vemuri, and Suresh Namala.

7. REFERENCES

[1] Aronson, E., Blaney, N., Sikes, J., Stephan C., and Snapp, M. *The Jigsaw Classroom*. Beverly Hills, CA: Sage, 1978.

[2] Clarke, J. *Pieces of the Puzzle: The Jigsaw Method.* In *Handbook of Cooperative Learning Methods*. Westport, CT: Greenwood Press, 1994, 34-50.

[3] Graesser, A. C., Wiemer-Hastings, K., Wiemer-Hastings, P., Kreuz, R. and Tutoring Research Group. Autotutor: A simulation of a human tutor. In *J. Cognitive Systems Research*, 1999, vol. 1, 35–51.

[4] Johnson, D. W., Johnson, R. T. and Smith, K. A. Cooperative learning: increasing college faculty instructional productivity. In *ASHE-ERIC Higher Education Report No.* 4, George Washington University, 1991.

[5] Khandaker, N. VALCAM - an auction based learning enabled multiagent coalition formation algorithm for real-world applications. M.S. Thesis, Univ. of Nebraska, NE, 2005.

[6] Khandaker, N. Question classification and question similarity in I-MINDS. In *Independent Research Report*, Dept. Computer Science & Engineering, Univ. Nebraska, Lincoln, NE, 2004.

[7] Liu, X., Zhang, X., Al-Jaroodi, J., Vemuri, P., Jiang, H. and Soh, L.-K. I-MINDS: An Application of Multiagent System Intelligence to On-Line Education. In *Proc. IEEE-SMC*, 2003, 4864-4871.

[8] Liu, X., Zhang, X., Soh, L.-K., Al-Jaroodi, J. and Jiang, H. A Distributed, Multiagent Infrastructure for Real-Time, Virtual Classrooms. In *Proc. ICCE*, 2003, 640-647.

[9] Namala, S. An intelligent module for I-MINDS. M.S. Project Report, CSE Dept. Univ. of Nebraska, NE, 2004.

[10] Olney, A., Louwerse, M. Mathews, E., Marineau, J., Hite-Mitchell, H. and Graesser, A. Utterance classification in Autotutor. In *Proc. HLT-NAACL Workshop*, 2003, 1-8.

[11] Sekine, S. and Grisman, R. A corpus-based probabilistic grammar with only two non-terminals. In *Proc. IWPT*, 1995, 216-223.

[12] Soh, L.-K., Jiang, H. and Ansorge, C. Agent-Based Cooperative Learning: A Proof-of-Concept Experiment. In *Proc. SIGCSE*, 2004, 368-372.

[13] Soh, L-K., Khandaker, N., Liu X. and Jiang, H. Computer-Supported structured cooperative learning. To appear in *ICCE'2005*, 2005.

[14] Ye, Y. and Churchill, E. *Agent Supported Cooperative Work*. Boston, MA:Kluwer Academic Publishers, 2003.

The Role of Agents in Enterprise System Management: A Position Paper

Onn Shehory
IBM Haifa Research Lab
Mount Carmel, Haifa
31905 Israel
+972-7689462
onn@il.ibm.com

ABSTRACT

The size and complexity of enterprise computer systems are growing rapidly. As a result, system management has become increasingly difficult and expensive. In fact, management costs are typically estimated at 50%-70% of the total cost of ownership. Despite large investments in management software and personnel, enterprise computer systems are usually managed sub-optimally. This situation calls for a fundamental change in the way systems are managed.

Recent studies suggest that systems manage themselves autonomously. Initial studies towards that goal reside in the fields of Autonomic Computing and Self* Systems. Many of those studies devise mechanisms for system monitoring, data filtering, problem determination, problem fixing and system adjustment and reconfiguration. The mechanisms are based, in large part, on methods from Artificial Intelligence, Statistics, Operations Research and Software Engineering.

Current enterprise system management solutions – theoretical and practical – are centralized. However, the typical enterprise system is distributed. We claim that a centralized solution presents several risks. For instance, it introduces a management bottleneck, it incurs communication overheads, etc. We hence advocate that enterprise system management should be addressed in a distributed manner. Combining distribution and AI, we believe that an agent-based solution is appropriate. In this paper we present in more detail the suggested approach.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]
I.2.11 **[Distributed Artificial Intelligence]**

General Terms

Management, Performance, Reliability.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

Copyright 2006 ACM 1-59593-303-4/06/0005...$5.00.

1. INTRODUCTION

The size and complexity of enterprise computer systems are growing rapidly. Contemporary enterprise systems typically include thousands of hardware and software components distributed across networks and geographies. These components are heterogeneous in their functionality, their interface with other components and with human users, and their manufacturers. Additionally, management practices and tools, when provided with the components, vary considerably in their capabilities. For instance, some tools merely monitor a small set of operational parameters and provide a report on their values, whereas others may be collecting a large and diverse set of parameters and provide complex analysis such as trending, prediction and bottleneck analysis. Note however that, even had all of the components been provided with the best management tool each, the management of the entire enterprise system will not instantaneously be resolved.

In managing an enterprise computer system, one needs to manage not only its components, but the relationships between them as well. Theoretically, this introduces a combinatorial problem, as any combination of components should be considered in the solution. Indeed, in recent years system management has become increasingly difficult and expensive. The costs of management are typically estimated at 50%-70% of the total cost of ownership, as can be inferred from multiple market studies (see e.g. [12]). Large investments in management software have not yet produced good solutions. Apparently, as many system administrators admit, enterprise computer systems are usually managed sub-optimally. This situation calls for a fundamental change in the way systems are managed. We advocate that this change will require a combination of Distributed Systems [9] and Artificial Intelligence techniques. The discipline that combines such techniques is Distributed AI—the parent discipline of agents and multi-agent systems.

Recent studies suggest that systems manage themselves autonomously [5]. Initial studies towards that goal reside in the fields of Autonomic Computing [3][6] and Self-* Systems [8]. The idea of systems that manage themselves is not new, though. Computer scientists have envisioned such systems since the era of Turing and von-Neumann [10][11]. Attention to these ideas has returned only in the last few years. Research in the area studies, among others, self-diagnosis, self-organization, self-configuration, self-awareness, self-management, etc. Many of these studies

devise mechanisms for system monitoring, data filtering, problem determination, problem fixing and system adjustment and reconfiguration (e.g., [2]). The mechanisms are based, in large part, on methods from Artificial Intelligence, Statistics, Operations Research and Software Engineering. Notably however, techniques for addressing distribution-related other than networking aspects of self-management are rarely employed. Further, the extent to which solutions address heterogeneity is limited.

Current enterprise system management solutions – theoretical and practical – are mostly centralized, although the typical enterprise system is distributed. Management solutions require human intervention and they are far from being autonomous. We argue that a centralized solution presents several limitations and risks. For instance, it introduces a management bottleneck, it incurs communication overheads, etc. We hence advocate that enterprise system management should be addressed in a distributed manner. We also argue that increased autonomy of the management system should reduce its TCO. As suggested above, we believe that an agent-based solution is appropriate. In this paper we present in more detail the suggested approach.

2. CURRENT SOLUTIONS

The goal of an enterprise computer system is to provide the enterprise with services of data processing, data communication and data storage. These services should be provided at a certain quality, to support the business operations and goals of the organization. The goal of system management is to assure that the system indeed provides the services expected, at the quality expected.

For instance, consider an organization whose business is graphical design. One goal of the system may be the retrieval of graphical files at a maximum delay of one second, regardless of the location of the file and its requestor. Of course, one should expect the initial system configuration to support this requirement. However, since data volumes and usage patterns change over time, and since system components may fail or degrade, the system might, in times, fail to address the required retrieval time. The major means for reducing the rate of occurrence of such problems is high-quality system management.

As stated above, the ultimate goal of system management is to assure service provision at a certain quality. Had management solutions fully addressed this goal, there would be no other requirement from these solutions. However, since no solution fully addresses the management goal, other requirements are typically set. These include the following:

- Reports on the system status: are the goals set for the system satisfied? What are the deviations from these goals?
- Trending and prediction: what are the system's patterns of operation? What system status would they lead to?
- System analysis, e.g.:
    - Dependency analysis: which application or component depends on which, and in what ways?
    - Root-cause analysis: what is the root-cause of a reported problem?
    - Bottleneck analysis: which components in the system experience the highest load, and which ones will fail first as a result of a high load?
- Remediation: what can be reconfigured or fixed, and how, to resolve reported problems, and to avoid problem build-up?

The requirements listed above are addressed to a limited extent by existing solutions. Fielded solutions, and in particular enterprise system management tools, mainly address the reporting and trending requirements. That is, such tools perform the following:

- Monitor the system – its components and applications – and store the collected data (possibly after some filtering and processing).
- Generate reports from the stored data
- Detect abnormal system and component status
- Compute simple trends
- Support a restricted set of component and system reconfiguration actions, including automated component replacement

The management functions listed above are not supported by all of the management tools. In practice, many tools offer a rather limited management function: they usually monitor the system and report on system status and trends, detect some types of abnormal behaviour, and, in some cases, they allow for (partial) automated reconfiguration of the system to address detected problems, commonly in a human-in-the-loop manner.

Solutions to the system management problem performed in research labs and published in the literature provide a variety of functions which are not yet part of fielded solutions. In particular, analyses such as dependency analysis, root-cause analysis, bottleneck analysis and prediction are addressed (e.g., [1][4][7]). Additionally, system re-configuration and a few self-healing functions are also resolved. However, research-grade solutions have several limitations, as follows:

- They usually solve a single, isolated problem, and do not provide a comprehensive solution as a management tool should.
- They tend to be theoretical, in the sense that they typically make assumptions that do not hold in the field.
- Commonly, they do not attempt to reduce the computational complexity of the suggested solution

Despite these limitations, it is expected that research solutions will eventually move into fielded tools. Note however that research solutions, although more advanced than tools, are far from addressing all of the requirements listed above. This invites additional research, and as stated above, agents and multi-agent systems could be the right discipline for this research.

3. TYPICAL ARCHITECTURES

In this section we describe the common architectural properties of enterprise system management tools. Although typical enterprise systems are distributed systems, both the fielded and the research solutions to the management problem are mostly centralized.

A management tool consists of five major layers: monitoring infrastructure, a database, an analysis layer, an activation layer and a user interface. An illustration of a typical architecture is depicted in Figure 1, where the management tool appears in grey. Typically, the monitoring infrastructure is not monolithic. Rather, it is comprised from multiple monitoring elements. Each such element (in times referred to as an agent) may be located on, or near, the component(s) it monitors. It collects data from the components it is assigned to using standard protocols such as SNMP and CIM/WEBM. The monitoring elements send the collected data, in some cases after some simple filtering, to a *central* management point to be stored in the database and analyzed as found appropriate.

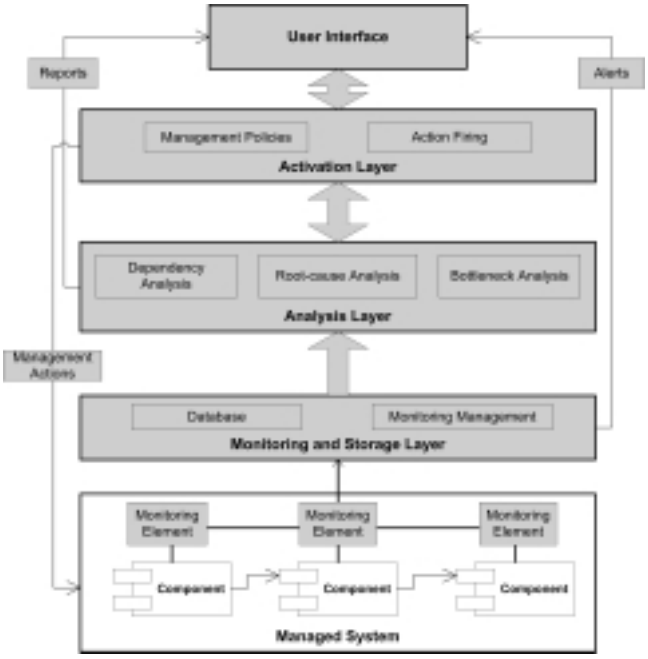


Figure 1. Management tools: a typical architecture

The analysis layer is typically located at the central management point. It may be implemented as a single monolithic software component, or as a collection of such components each specialized in a specific type of analysis. The analysis layer retrieves from the database data collected and filtered by the monitoring layer. The results of the analysis are sent either to the user interface as reports, or to the activation layer, for policy updates, planning and action firing.

The activation layer, located at the central management point as well, consists of an action firing component. It usually consists of means for representing management policies and activation rules too. Action firing is performed either by the user, or automatically via activation rules, following management policies. Actions are performed directly on system components and may include parameter setting, component and system reconfiguration, component replacement, and others.

The user interface usually runs at the central management point, however web access to the user interface is becoming increasingly common. The user interface provides the user with access to an array of management functions. These include systems visualization, reporting and trending, analysis activation and results, action firing (see details above), and management of the management tool itself.

The architecture described here does not cover all types of management tools, however it represents well many of them. One of the prominent common properties of these tools is their centralization. Typically, such tools will be running on a server dedicated to system management. The advantages of this approach are the computational efficiency of data retrieval and analysis, and the high reliability of the servers on which the management software runs. However, the centralized solution has several weaknesses, as follows:

- **Communication overloads**: Frequent monitoring of multiple components across the managed system generates large data streams. These are sent to the central management server, possibly overloading the server and the network. Contemporary centralized management tools address this problem by reducing the frequency of their monitoring. As a result, the quality of management may be compromised.
- **Computation overloads**: When multiple analysis methods are activated concurrently to analyze a large number of components and their operational parameters, a central server might exhibit computation overloads, which will in turn result in slow analysis and delayed response to problems detected in the managed system.
- **Single point of failure**: A widely known problem of centralized solutions is them being a single point of failure. In this respect, system management tools are not an exception. A failure of the management tool may result in degradation in the quality of services provided by the managed system, and may lead to its failure.

The weaknesses discussed above call for a change in the way management tools are designed and deployed. In particular, distribution could provide some remedy to all issues raised. However, distribution may introduce new problems, as follows:

- **Synchronization and consistency**: Once the management of a system is distributed, the widely known clock synchronization problem arises. Additionally, when data is distributed across multiple management points, inconsistency among them may arise. These problems may in turn result in errors in data analysis and reporting, and in action firing.
- **Communication overhead**: The distribution of management across the managed system requires coordination among the managing points. This in turn results in increased levels of communication traffic and may impose a communication overload. Careful design of the coordination can minimize this risk.

It appears that, although distribution could provide significant advantages to system management tools, there is a need for a careful design to reduce the risks introduced by distribution. In the following we propose a possible approach to the distributed design of enterprise system management tools.

4. AN AGENT-BASED APPROACH

Given the requirements for data collection, filtering and analysis, and for action firing, across a distributed heterogeneous system,

we believe that an agent-based solution is adequate. Our proposed solution approach is as follows:

- **Data collection**: In current solutions, data is collected and sent to a central management point. We suggest that collection will be performed by agents that implement some intelligence. In particular, they should intelligently filter the data collected, and transmit, either to a central point or to other management entities, a small subset of the collected raw data. This should resolve the communication overhead present in current solutions.
- **Data analysis**: In current solutions, analysis is performed at a single central point. We suggest that each type of analysis be performed by a specialized component or agent, possibly located on a separate machine. This should resolve the computation overhead present in current solutions.
- **Action firing**: In current solutions, management policies are maintained at, and actions are fired from, a single central point. We suggest that some policy maintenance and action firing will be moved to the agents residing next to the managed components. In particular, some policies and actions are relevant only to a small subset of the system, and it should be efficient to maintain the former, and fire the latter, in vicinity to their target components. By this, communication overhead will be reduced and timeliness of management actions will be improved.
- **Coordination and collaboration**: In current solutions, due to their central approach, no coordination or collaboration are implemented. In a distributed solution where data collection, filtering and analysis, and action firing, are performed locally by agents, there is a need to coordinate among them to avoid inconsistencies and prevent conflicting and redundant actions. Collaboration could further enhance the capabilities of the management tool. We suggest leveraging multi-agent coordination and collaboration methods to address this goal.

5. CONCLUSION

The solution approach suggested in this paper is described at a conceptual level. The main aim of this description is not to provide guidelines for developing agent-based management systems. Rather, we want to raise community awareness to a problem domain in which, to date, little was studied, and where agent-based solutions appear as a promising solution approach. As we demonstrate, distribution, local autonomy, intelligence and coordination could provide significant improvements to the quality of enterprise system management solutions. Since agent-based systems inherently realize these properties, we believe that agents and multi-agent systems should be studied and utilized in this context. We are confident that the results of such research will span far beyond this specific application domain, influencing the disciplines of system management, self-*, autonomic computing and, of course, agents and multi-agent systems.

6. REFERENCES

[1] Breitgand, D. Henis, E. Ladan-Mozes, E. and Yerushalmi, E. Root-Cause Analysis of SAN Performance Problems: an I/O Path Affine Search Approach. In IFIP/IEEE 9th International Symposium on Integrated Network Management, IM'05, Nice, France, May 2005.

[2] Dilman, M. and Raz, D. Efficient Reactive Monitoring. IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Recent Advances in Network Management, 20(4).

[3] Ganek, A.G. and Corbi, T.A. The Dawn of the Autonomic Computing Era, IBM Systems Journal, Vol. 42(1), 2003.

[4] Hellerstein, J., Zhang, F. and Shahabuddin, P. A Statistical Approach to Predictive Detection. IEEE Computer Networks, January 2000.

[5] Herrmann, K., Mühl, G. and Geihs, K. Self-Management: The Solution to Complexity or Just Another Problem?. IEEE Distributed Systems Online, V 6(1)1, 2005.

[6] Kephart, J., Chess, D. The vision of autonomic computing. IEEE Computer, January 2003, 41-50.

[7] Kar, G., Kochut, A., Bobroff, N. and Beaty, K. Management Issues in Storage Management Networks: Detection and Isolation of Performance Problems. In IFIP/IEEE 9th International Network Operation and Management Symposium (NOMS'04), Seol, Korea, March 2004.

[8] Proc. of SELF-STAR: International Workshop on Self-* Properties in Complex Information Systems, 31 May - 2 June 2004, University of Bologna, http://www.cs.unibo.it/self-star/program.html

[9] Tanenbaum, A.S. and van Stee, M. Distributed Systems: Principles and Paradigms, Prentice-Hall, 2002.

[10] Turing, A.M. Intelligent Machinery, 1948, Report for National Physical Laboratory, in Machine Intelligence 7, Meltzer, B. and Michie, D., Eds. 1969.

[11] von-Newmann, J. and Burks, A.W., Eds. Theory of self-Reproducing Automata, University of Illinois Press, 1966.

[12] http://www-1.ibm.com/linux/whitepapers/robertFrancesGroupLinuxTCOAnalysis05.pdf

Implantable Medical Devices as Agents and Part of Multiagent Systems

Touby Drew
Medtronic Neurological, Inc.
800 53rd Avenue NE; MS N240
Columbia Heights, MN, 55421-1200
touby.drew@medtronic.com

Maria Gini
University of Minnesota
4-192 EE/CSci, 200 Union St SE
Minneapolis, MN 55455
gini@cs.umn.edu

ABSTRACT

The consideration of medical implants as an increasingly important population of isolated agents is a valuable perspective that should not be ignored by the agent community. Implanted Medical Device (IMD) applications are complex, naturally distributed, and could benefit from such attention. This paper explores implantable medical devices and their attributes in an agent context and terminology. It submits that an increasing body of IMDs should be considered agents and that there are opportunities for incorporating these implantable agents into multiagent systems (MAS). This will include: (i) Discussion of several IMDs in traditional agent terms. (ii) Discussion of trends and issues in IMDs related to their potential role in MAS. (iii) Experimental exploration of some potential MAS applications in the problem of medical monitoring. (iv) Broader discussion of the value of framing IMDs and applications involving them in the agent paradigm.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Artificial Intelligence

General Terms

Medical Monitoring, Implant, Medical Device, Agent, Multiagent System, Implantable Agent

1. INTRODUCTION

Currently there is a wide range of different implantable medical devices (IMDs). Although the importance of stents, bio-implants, valves, joint replacements and other "passive" IMDs is clear, this paper focuses largely on those which contain more sophisticated processing capability (sometimes called 'active' IMDs in industry). Still, in this category there are quite a number of IMDs made by a range of companies. A variety of pacemakers, Implantable Cardioverter Defibrillators (ICDs), neurostimulators, programmable drug pumps, and other devices are currently being used by patients around the world. These devices are created by companies including Medtronic[1], Guidant, St. Jude Medical, Boston Scientific, NeuroPace, and many others.

1.1 Implants and the Agent Literature

There is already discussion in the literature of both medical monitoring and systems interacting with implantable devices [2, 6, 9, 10, 15, 19, 24]. As part of this there has even been the limited suggestion that IMDs responsible for monitoring patients "might also benefit from natural language and artificial agent technologies" [6]. To the knowledge of the authors, however, no one has discussed implants as agents or explored them as participants in agent-based systems. The broader discussion of what is and what is not an *agent* is already present in the literature and provides an important context in which we consider this material in Section 2.

1.2 Brief History of Pacemakers

About a decade after the first external pacemakers were designed and built, Medtronic began to produce early implantable pacemakers[2] and continues to manufacture their complicated descendants today. The earliest pacemakers were fairly simplistic as they supported no real sensing, processing, or communication capabilities. After a few years pacemakers could be recharged and switched to different rates of pacing. Development of wireless techniques for communication emerged. Pacemakers were eventually made to be "responsive" to their environment in a closed loop fashion so that they could, for example, sense the level of activity of their host and change the pacing rate. Many modern devices involve significant processing to facilitate sensing, monitoring, detecting, responding, communicating, and other activities. Several of these support so called "downloadable firmware" in which at least some software code can be sent to the device for it to run in addition to or as a replacement

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

[1]This company employs the primary author of this paper, is the largest IMD maker with a broad range of cardiovascular, neurological, and other devices.
[2]The earliest work was done by a Swedish team on a patient who ultimately went through 22 pacemakers and 5 lead systems over about 43 years until he died in 2001 from a problem not related to his pacemaker systems [11].

for existing code.[3] [4]

2. IMPLANTS ARE AGENTS?

We submit that some IMDs might be considered to meet many of the criteria and completely satisfy some of the notions set for agency in AI tradition and literature. By reviewing specific IMDs and agent concepts we hope to explore and help elucidate the place of implants in the spectrum and taxonomy of agents. In a longer version of this paper [4] we address these questions in greater detail, including agent properties such as persistence, action-perception relationship, autonomy, social ability, mobility, benevolence and veracity, etc. Here we will discuss environments, sensors, and effectors as key aspects of agents which are "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors." [23]

2.1 Implants' Environments

IMDs transition through a few environments (e.g. in the sequence Manufacturing, Implant(ed), and finally Explant(ed) with the majority of device life in an implanted environment). Consider the SynchroMed II^{TM}(S2) implantable drug pump as an example. It may be useful to think of an implant as a very high reliability robot, but one which is never turned off and is almost always capable of wireless communication. Generally, within a minute of power a S2 has completed some initialization, diagnostics and logging; enabled wireless communications; sounded an audible alarm, and depending on its configuration (or lack there-of) begun to pump drug and sense its capability to do so. After it is completely assembled, a S2 is communicated with wirelessly to ask it to configure itself and perform diagnostics. Eventually, a clean-room worker would put the implant into a sterile package where it would remain until taken out in the operating room. Eventually the device would be put into its implant(ed) environment. At this time, a doctor uses the physician's programmer to inform the implant of this change. The device will do some processing and make some changes to its behavior. As time passes and different events occur the device eventually will indicate that the physician should consider replacing it (Elective Replacement Indicator or ERI) and may change its behavior (especially if allowed to reach End Of Service or EOS). At some point, the device will be removed from the patient (explanted) and discarded or returned to Medtronic for analysis.

2.2 How Implants Sense Their Environment

There is a great variety of sensors employed in IMDs to allow them to better and more appropriately sense their environment. The S2 device, for example, senses its electromagnetic environment for the presence of wireless telemetry communications. The S2 also monitors itself, checking for stall, measuring its battery voltage, tracking the passage of time, self-diagnosing numerous other issues that might arise and responding appropriately based on its understanding of its environment, capabilities, status, history, and configuration. During testing each S2 pump expels fluid while monitoring itself. If the pump were to stall (e.g in the presence of a strong magnet or damaged part) the S2 would detect this and indicate it by audible alarm and indication in a following telemetry session. For a more subtle example consider that the S2 tracks how much volume it has pumped and if it believes it is implanted when this exceeds some threshold it will trigger an audible alarm and consider itself in an ERI or EOS state. Furthermore, if at that time a physician asks the S2 to return to its "implant behavior" it will not (even where the condition causing it to leave its implant behavior was transient such as recognition of a temporary dip in battery voltage). Even so, relative to many other IMDs the S2 is relatively sensor poor.

In the broader pool of IMDs more sensors have been proposed and put to use. We mentioned earlier that cardiac pacemakers make use of sensors to determine patient activity. The determination of activity, posture, etc is accomplished for example using 1, 2, or 3 axis accelerometers in a range of devices like the Medtronic Kappa 900^{TM}. For the most part, cardiac devices also monitor the electrical activity of the heart (ECG) using leads to propagate the signals. Recently some devices including Medtronic's Reveal^{TM}Insertable Loop Recorder and Reveal Plus have enabled "leadless" or subcutaneous sensing of the ECG. Yet other cardiac devices are sensing oxygen, pressure, or other[5] levels [20].

Furthermore, though traditionally limited in their sensing capabilities, neurostimulators and other IMDs are bringing traditional and novel sensing modalities to new areas of the body; for example the NeuroPace Responsive NeuroStimulator^{TM}(RNS) which measures the electrical activity of the brain (EEG) to detect epileptic seizures [18]. Temperature sensors have also been used (e.g. to address tissue heating with recharge). Some devices utilize sensors to simplify patient communication (e.g. using accelerometers to detect a patient tapping on the implant or respond to the presence of a simple magnet). In the future, research may lead to direct "Brain Computer Interfaces" [13]. For shorter life IMDs, a number of other sensors, including notably pH sensors, like those in Medtronic's Bravo^{TM}, can be used. In the near future, IMD sensors may sense a range of new things including chemicals like "hormone levels, anesthetic agents and other factors within the human body such as ... drugs" [16].

2.3 How Implants Affect Their Environment

The S2 devices, used as an example previously, affect their environment by delivering drug to it. With the ability to kill or incapacitate a patient by delivering drug too quickly (overdose; e.g with morphine in pain therapy) or too slowly (under-dose; e.g. with Baclofen in spasticity therapy), the significance of this effector can hardly be understated. Although more rarely utilized, another effector is the S2's built in audible alarm which can generate a range of tones and patterns to alert the patient, caregivers, or other user of different conditions. Also of great importance, but more subtle is the device's ability to generate wireless fields (for the purpose of communicating).

While also able to communicate wirelessly, cardiac devices have slightly different effectors that often include the ability to electrically stimulate the heart. In the future

[3]This is presumably valuable for confronting issues related to software/firmware which occur in the field but more fundamentally illustrates the flexibility of the devices.

[4]The capability of IMDs varies; typical clock speeds 32kHz to 3MHz, memory (RAM,ROM,EEPROM,Flash,...) 8kB to 2MB, and miscellaneous custom circuitry.

[5]For example respiration or lung fluid levels via impedance measures.

cardiac and other devices may include both electricity and drug delivering effectors [8]. Furthermore, stimulation is not targeted only to the heart, for example, neurostimulators like Medtronic's Activia™ device for Parkinson's therapy and NeuroPace's RNS for epilepsy both use stimulation targeted to the brain, and those like the Cyberonics device deliver stimulation through cuff electrodes to the vagal nerve. Cochlear implants, various sight related implants (including direct interface to the visual cortex, subretinal implant, etc.) [26], diaphragm stimulation [3], and a number of other devices and therapies reflect the diversity of stimulation targets. Even beyond electrical stimulation, drug delivery, wireless communication and audible alarm there are some "other" and emerging technologies for implantable effectors. Certainly an exhaustive list of sensors or effectors is beyond the scope of this paper, but consider for example potential implantable thermoelectric devices, to be used for example in cooling the brains of epilepsy patients [12].

3. IMPLANT TRENDS AND POTENTIAL ROLES IN MULTIAGENT SYSTEMS

The IMD industry is large, competitive and growing quickly as it reflects the developing market for the devices it produces. IMDs themselves are becoming smaller, longer lasting, more broadly applicable and better reimbursed both in response to and as a fuel for their growing industry. As medical devices and other technologies keep patients alive longer the aging population further supports the industry with replacement and combination therapy needs[6]. Important results of these positive feedback systems are that IMD systems are becoming more prevalent, flexible, complex, capable and accessible to potential customers and users. As this continues, we submit that the potential role and importance of IMDs in multiagent systems and as implantable agents themselves is also growing.

While there is strong pressure to add intelligence, capability and flexibility to implants, these things often bring complexity. Even more so than in other systems, there is also pressure to keep implantable components as simple as possible so as to increase their longevity[7], simplify testing, reduce the likelihood of error, reduce device cost[8], and speed time to market. With these forces there is pressure to move some system capability outside of the implantable device and into instruments which communicate with it from outside of the body. In practice there has been an increase in communication capabilities, range and complexity in instruments like the patient and physician programmer devices that are designed to communicate with implantable devices.

This growing role of instruments can be seen not only with traditional, life-sustaining uses of implants (e.g. pacemakers), but also as IMDs are increasingly used for non-life-sustaining therapy, combined therapy and monitoring, exclusive-monitoring and other applications. As this trend continues, the volume and complexity of the data available from these devices also grows, presenting the primary problems of medical monitoring with implantable devices. This results in further pressure for increasing memory, automation, decentralization, and prominence in the role of external instruments and in many cases has affected a shift to a more patient centric system involving patient programmers, telemedicine, and transferable electronic medical records. Despite the complexity more patients, physicians and caregivers are using instruments which interact with medical devices and the data they gather and as this continues the role of human computer interaction and "human factors" becomes increasingly important [14].

3.1 Implantable Agent Interaction

Traditionally IMDs have been fairly solitary agents, for the most part interacting only with their environment and the humans that communicate with them almost directly (through simple programmers and the applications that run on them). Consider that in the future this could change, and implants may begin to interact with other agents to a greater extent. This could take the form of implant to implant communications, implant to wearable agent communications, or implants communicating with agents that are transiently (or ubiquitously) available in the environment.

3.2 Implanted Networks

As some of the trends like increasing device prevalence continue, the likelihood of finding multiple implantable devices within communication range increases. This enables implant to implant communication in the field. Although the idea of the IMDs in two acquaintances or even strangers utilizing their transient link is fascinating, the simplest and most tangible cases to consider might be between implants concurrently implanted in the same person. There is some precedent for this, concern, and even design relating to this in existing IMDs. Despite this concern and ongoing work, there is significant potential for and interest in implant-to-implant communication within a patient's body [1, 7]. Some of the motivations for this might include surgical implantation issues, device coordination issues, robustness[9], as well as resource and capability sharing[10].

3.3 Implantable-Wearable Systems

One of the simplest and most obvious uses of implantable-wearable systems is one in which a single wearable component regularly interacts with a single implantable compo-

[6]It has been well documented in agent literature that the aging population presents a significant target for agent applications [21].

[7]Implants often do as little processing as feasible to reduce the power consumed by the microprocessor and other circuitry. By reducing power, the device can last much longer with the same size battery.

[8]As with any highly regulated, mass produced, embedded system where space and power is at a premium, the cost of memory and other miniature electronic components, even read only memory, can be high and have an effect on production cost.

[9]In the simplest case, the implants could regularly check on each other, and if for some reason one consistently does not respond appropriately, the other could respond (e.g. alert the patient, log troubleshooting information, etc.). More practically implants could propagate messages to each other increasing both the reliability and size of their telemetry cloud.

[10]For example, only one device need have an audible, vibratory or other means of alerting the patient and other devices could use it. In another embodiment, a primary closed loop therapy (e.g. neurostimulation for epilepsy) may recognize that it is being unsuccessful in suppressing a life threatening event and call in the services of a secondary device (e.g. implantable drug pump delivering a potent anti-epileptic drug).

nent. The most fundamental uses of such systems are valuable for supplementing basic implantable capabilities (like processing, sensing, and memory). Consider for example that modern cochlear implants generally include an external device which senses and processes audible inputs and an implanted device which stimulates the auditory nerve. Conversely, as an example, implantable memory is not sufficient to store the large amounts of data modern cardiac (and other) devices monitor and so these devices can be used with an external holter monitor like device to record much greater amounts of data (which may be particularly useful in tuning therapy). For the Medtronic Restore^{TM}Neurostimulator system, there is currently a wearable system for recharging the implantable. Over a period of time the Restore recharger provides a strong field in short bursts enabling inductive recharge. In between bursts, the recharger measures and models the temperature and then decides if and how long to wait before continuing the recharge (to avoid implant heating issues). Another potential embodiment for such a system, and one perhaps more in line with the tradition of telemedicine, might be one where wearable devices are not the primary link between implantable agents and the rest of the system. Instead, implantable agents may interact directly with agents which are only transiently available (e.g. not worn or implanted in the same body-area network).

3.4 Implantable Communication with Transiently Accessible Devices

Telemedicine systems which occasionally interact with implantable devices are already achievable and in place. Consider for example that a relatively small, privately held company NeuroPace has developed and tested the NeuroPace Data Transmitter (DTR) system. A recent abstract on this system for "on demand monitoring in the Outpatient Environment" explains: "On an intermittent basis, the DTR is connected to a phone line and data is transmitted to the secure server where it can be viewed by the treating physician over any Internet connected computer." [22] To the knowledge of the authors, it is not yet discussed in the literature whether findings in more traditional cardiac centered telemedicine systems can be generalized to systems and therapies like the NeuroPace DTR. With this caveat in mind, there is evidence that such systems are practical to implement and provide clear potential for cost savings [5]. These systems are not perfect with respect to usability, patient compliance, data loss and other concerns, but Medtronic's CareLink^{TM}system, with over 50,000 patients, is an example of their viability at a fairly large scale. These traditional telemedicine systems involving implants require the patient to consciously interact with the system on a regular basis. We submit that the interaction between a transiently available telemedicine system and implants could be further automated and its users better served using a multiagent approach as we will discuss in Section 5.

Beyond traditional telemedicine motivations, there are other applications and embodiments of implantable communication with transiently accessible devices some of which might benefit from agent technologies. We leave these for future work, but will suggest that these may involve distributed clinical research, anonymization, authentication and identification, patient based electronic medical records, device update, maintenance and support, data integration and summarization, emergency response and localization, trauma and triage, automatic fleet management, improved software platforms for interacting with medical devices, auditing, reporting, novel physiologic computer interfaces, etc.

4. IMPEDIMENTS TO IMPLANTABLE AGENT APPLICATIONS

4.1 Business Case

Although it is fairly clear that IMDs could technically be used as part of a multiagent system, the business need and benefits of such a system are, perhaps, less obvious. As discussed elsewhere in this paper, the use of a MAS approach could improve data management, instrument usability and possibly other aspects of IMD systems, but the cost of their implementation, validation, deployment, is a significant business hurdle. Even a relatively large telemedicine system like CareLink^{TM}would have a hard time overcoming these costs in the sale of its services. Consider, however, that this is not the business model that really matters. Revenue from telemedicine systems like this is very small in comparison to that from device sales. When viewed as a feature that could differentiate a product or company and potentially grow market share and increase implant sales the business case and potential is much more apparent. Potential future uses in distributed clinical and device research as well as new device applications (e.g. requiring collaboration or improved adaptability) may also become business drivers.

4.2 Technical Stability and Complexity

It is certainly true that some implants are already complex, real-time systems, difficult to upgrade or fix without replacement, performing critical functions, and need to function reliably for many years. For these and other reasons, keeping the implants simple is important because in may ways, simpler is safer. On the other hand, as an underlying premise of this paper we submit that some active implants are already capable of acting as agents there would be little or no added complexity *in the implant*. Interestingly and although not widely discussed in agent terms, this existing design is in large part due to the desire to provide flexible but robust and autonomous systems and is part of what makes it transparent to the implant whether it is communicating with another agent or simply a program.

As a broader MAS including non-implantable agents, there would also be added complexity and thus cost, perhaps especially in testing. The related issues would almost certainly mean slow adoption and limits to functionality initially. It is almost certain in the near future that the connection of IMDs with all other agents and resources of a MAS will be limited. Throughout this paper it is assumed that implants would play a limited, intermittent role in MAS's but could still have significant potential benefits and in some part it is this very requirement for some independence and automation that makes IMDs and the components that interact with them natural candidates for AS and, we submit, MAS applications.

One interesting subtlety worth noting is that instrument software which communicates with implants currently has an extremely long life cycle as it is expected to be used and maintained or replaced from launch date until the last date at which any device of the supported type is still implanted (which could be more than a decade after its implant) or

on the order of 15 years. Over this time tens of millions of dollars are spent on migrating applications to new platforms as operating systems and other once plentiful and well supported things become obsolete. For this reason portability, extensibility, testability and longevity are especially important. To address some of these and other concerns, some have proposed that applications interact with a single set of autonomous, persistent agents (e.g. for telemetry or data management) which are discovered and interacted with at run-time using appropriately flexible languages.

4.3 Legal, Regulatory, and Security

Though the specifics of these and other important concerns are well beyond the expertise of the authors, it is worth mentioning explicitly their existance and some initial opinions. First of all whether certain IMDs are viewed through the agent lens or discussed with agent terminology or even designed with certain agent properties in mind is not of direct importance to regulatory agencies like the FDA. Instead, the clear and potential risks, mitigations and benefits of such designs are of primary importance. Furthermore, we submit that certain products already approved by such agencies are already reasonably considered agents and in agent terminology by ourselves and some of our peers. On the other hand, this clearly does not change the fact that solid verification, validation, and consideration of risks and alternatives needs to happen for any medical product, especially IMDs. Privacy (e.g. in HIPAA compliance), robustness in the face of misuse, intellectual property clearance, and potential liability issues are at least in large part the responsibility of the manufacturer and essential items to address in potential products (AS, MAS, or not).

5. MAS IN MEDICAL MONITORING WITH IMPLANTS

There are significant motivations for medical monitoring. The most basic approaches to medical monitoring with implantable monitors center on having the patient periodically come in to a clinic where a caregiver manually uses an external tool (e.g. programmer) to off-load data from the IMDs. Telemedicine systems are becoming another important approach, but one which requires patients to manually use a device in their home to send data to physicians electronically (e.g. indirectly through the CareLink^{TM}system or on electronic medium recorded by a patient programmer). Both of these approaches have positive and negative aspects. For example, a traditional clinical visit may be more easily reimbursed, does not require a patient-device interaction (e.g. CareLink^{TM}base-station upload session or patient programmer downloading session) and might better facilitate a verbal dialog or other complex, high-fidelity interaction (e.g. visual inspection) which may have positive effects on patient perception as well as outcome. On the other hand, telemedicine approaches may reduce travel costs, increase the speed and ease with which data can be accessed and reviewed, automate electronic documentation, and allow both the patient and physician to participate asynchronously and outside of the clinic. These are generalizations, but we believe that there may be advantages to a medical monitoring system based on a multiagent framework in which an implantable agent interacts with at least one external agent. Some of these benefits could be realized by a MAS as simple as a basic telemedicine system in which the traditional base-station software has been changed to accommodate an agent which autonomously propagates information from an implantable agent to the physician (or other part of the system).

It is difficult to assess a MAS approach to medical monitoring because such a system did not exist at the start of this project and to test it appropriately would require a clinical study. Furthermore, to generalize the results of such a study the patient population would have to be sufficiently sizable and reflective of the many different device technologies, patient types, and other factors over a sufficient period of time. However, as a small first step on the theoretical, confidence-building journey to this end we provide the results of some simple simulations to explore some of the issues discussed.

5.1 Methods

Using the Swarm agent based simulation framework simulations were created to explore the medical monitoring domain and especially the data loss problem. The simulation was parameterized to allow easy configuration of settings related to simulation type (traditional, telemedicine, or MAS), range over which the rate of data accumulation is set for the population, range over which time elapses between office visits, frequency with which data-offloading is attempted (by the patient in the telemedicine style simulation or by the external agent in the MAS style simulation), etc. Experiments were defined by giving values to these parameters. The amount of data lost and amount of data off-loaded was calculated for each of the experiments described later to achieve the average of 10,000 agent-runs with each agent being simulated for 500 days.

5.2 Experiments

The most basic experiment explored simulation of the traditional approach to medical monitoring with IMDs in which data is accumulated (at a rate randomly distributed between 0 and 30% of capacity each day) until the implant's memory is full and from that point continually lost until the next office visit (set 180 days apart). To explore simulation of the modern telemedicine approach a second simulation alters this such that users attempt (with between 80 and 100% success) to off-load their IMD's data regularly (every 14 days) at home as well as during office visits. We also explored potential implementations of multiagent approaches to medical monitoring including one aimed exclusively at improving the success rate of home off-loading sessions (resulting in between 90 and 100% success).

Finally we explored a more realistic, though at the time not yet realized, multiagent approach to medical monitoring with IMDs that is aimed at approaching continuous monitoring. That is, it simulates agents which not only off-load their data during office visits but also interact with external agents to off-load their data at home. The key difference in this approach is that because the external agents and IMD agents autonomously off-load the data, they can attempt to do this at least daily and keep trying until they are successful (perhaps more than a day later) or determine no data need be off-loaded[11].

[11]In some respects, the interactive communications to determine the need to download data or to avoid downloading could be the most interesting. Telemetry with implantable devices affects their longevity through power drain to carry

Parameter Selection

In real clinical settings, the frequency of office visits, success rate of data-offloading attempts and particularly range of data accumulation rates vary significantly. The choice of 180 days between follow-ups reflects the conservative end of common practice with pacemakers (about every 6 - 12 months) and many other devices. The success rate of telemedicine offloading is also at the conservative end of that reported for CareLink [24]. Though many IMDs sense and process continuous real-time physiological signals and this data would be valuable for various retrospective uses, to our knowledge, there is currently no IMD which is capable of storing all of this for more than a short period. Instead IMDs tend to use tricks including lossy compression, saving only short data snippets surrounding detected events, and using priority schemes so the lowest priority data is overwritten first. The application of these approaches and interpretation of how much of the data is actually valuable varies widely, but we submit that our choice is conservative and would be so conservative for an external device recording the same signals that it would be laughable. It is also worth noting that in our experience the simulation results were not highly sensitive to relatively small changes in these choices.

LATITUDE and CONEXUS

The practical use and implementation of such a MAS may be hard to recognize, but could improve patient care in many ways. The industry is just now taking the first steps in this direction; consider that since the start of this work, Guidant has released the LATITUDE^{TM}Communicator system. This allows automated daily monitoring via a bedside device that operates in a manner largely transparent to the patient and is part of what Biotronic, Inc. spokesmen are saying is "definitely the wave of the future." [25] This system also includes an agent scale component which weighs the pateint and gives them spoken feedback and adaptive instructions. Further, since the first submission of this paper Medtronic issued a press release for the first clinical trial of its new line of IMDs with 2 to 5 meter range "Conexus Wireless Telemetry" which "will enhance efficiencies at device implant and during in-office follow-up visits, as well as enable *automatic*, wireless data transmission from the patients device to a home monitor." [17]

5.3 Results

Indicator Functions and Metrics

In the experimental scenarios described previously in this paper, only data which has been off-loaded is retained to be accessed outside of the implant. With this in mind, the amount of data that has been off-loaded is a valuable metric in determining success. Furthermore, the loss of data in medical monitoring is generally undesirable. The value of absolute data retention, or minimal to moderate levels of data loss depends on the intended applications, but may well not be linear. For example, some cases, like the retrospective analysis of continual real-time algorithms on the gathered data, may require continuous or near continuous data for very long periods to accurately reflect performance. In the experimental scenarios described in this paper, any data which is gathered by an implant full of not-yet-off-loaded data is lost. In volume this is equivalent to the loss of the oldest data not yet off-loaded in a full implant and can alternately be thought of this way if desirable.

out a full off-load. Interactions between IMD and external device could limit or terminate an off-load if the IMDs power reserves are low or if the data buffer is not yet sufficiently full to warrant data collection. In principle, this type of MAS-like interaction could optimize IMD performance across several parameters.

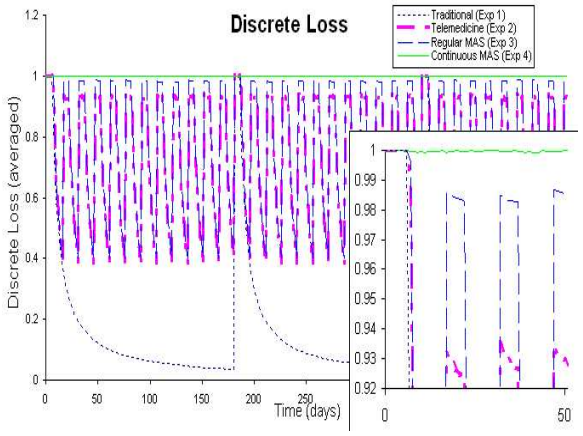


Figure 1: Discrete Loss: Experiments 1 - 4.

The mathematical relationship between the data lost by an agent pair (IMD and external home monitoring agent) is equal to the difference between the data on the device and the sum of the data off-loaded and the total data acquired by the IMD(see equation 1).

$$\begin{aligned}\text{dataLostFrom(agent)} \quad = \quad & \text{totalDataAcquiredFrom(agent)} - \\ & [\text{dataOffloadedFrom(agent)} \\ & + \text{dataOnDeviceOf(agent)}] \end{aligned} \tag{1}$$

We created a metric allowing better understanding of the data in graphical analysis called *discrete loss* or dloss. In this context, dloss is defined for each agent as 1 if the agent has not lost data since the last off-loading and 0 if it has (see equation 3). The discrete loss of the agent population is thus the average across agents (see equation 2) and reflects the portion of the population that has not lost data since their last off-loading. Thus for n agents data loss is defined as:

$$\text{dloss} = \frac{\sum_{agent=1}^{n} \text{dloss(agent)}}{n} \tag{2}$$

where for each agent

$$\text{dloss (agent)} = \left.\begin{array}{ll} 1 & \text{for : dataLostSinceOffload} = 0 \\ 0 & \text{for : dataLostSinceOffload} > 0 \end{array}\right\} \tag{3}$$

This indicator is useful in recognizing and understanding cases in which only a fraction of the agents are responsible for all of the data loss.

5.4 Discussion

Comparing the results of the simulations, it is clear that, given our assumptions, there are some potential advantages to be had with the application of multiagent techniques over traditional and telemedicine approaches in terms of data loss and data off-loading.

The simulations clearly reflect the data loss relationship (equation 1) and the parameters chosen relating to device memory capacity, data acquisition rate and device off-loading rate. As discussed, the parameters chosen reflect our subjective experience with medical monitoring implants but underly key assumptions which might be further explored in clinical studies and other future work. In speaking with medical and other experts who work with data and electronics, the diminutive amount of memory available in long-term implantable devices is often surprising.[12] In contrast consider that a single channel of EEG or ECG signal sampled at 256Hz with 2-byte precision might generate about 30kBytes per minute, filling all of memory for even extreme IMDs in only 1 or 2 hours.

The intuitive benefits of telemedicine over traditional approaches was apparent in our results with more than an order of magnitude improvement over the simulated traditional approach (Experiment 1) in terms of data off-loaded (and a factor of about 2.3 reduction in data lost). This improvement is due to the additional data-offloading opportunities afforded by the supplementary telemedicine approach.

Also as one might expect, the multiagent system (MAS) simulation aimed only at improving the off-loading success rate showed a small improvement over the simulated telemedicine approach as a result of the increase in off-loading success rate (gains in data off-loading and data loss rates of about 7 and 10% respectively). Considering only these differences, this rudimentary approach provides only incremental improvement that would likely not justify the cost of designing, implementing and validating it. On the other hand, it may be worth noting that this approach allows the patient to not worry about the process or their implant and perhaps even forget that it exists. This may be highly beneficial for patient usability and compliance (especially hard with elderly, cognitively imparied, or non-tech savvy), and perhaps in other ways as well. For example, it has been noted that patients who feel that their role is to recharge or otherwise service an implantable device may do so too frequently. This can negatively affect the device (e.g. depleting the battery) and reflect lost time and increased anxiety. Similarly, patient confusion and misuse of telemedicine equipment can result in costs to their healthcare and service providers by burdening call centers and technical service providers.

The final experiment, aimed at exploring a more realistic MAS approach demonstrated an improvement of almost 60% in data off-loaded and about 600% reduction in data loss (over the rudimentary MAS approach aimed only and increasing off-loading success rate). At the same time benefits in usability and transparency of the rudimentary MAS approach are still present. This simulation reflects a significant potential advantage of thinking of the medical monitoring system as a MAS. Though the findings may be intuitive, this is key evidence that suggests a MAS approach is appropriate because it hinges on autonomy and avoids patient compliance as a major issue. In other words, as a more complete paradigm shift it approaches a level of autonomous communication and cooperation between implant and telemedicine agent that is not practical in traditional approaches where asking the patient to constantly measure the contents of their implant, make off-loading decisions and initiate appropriate off-loads would be unreasonably burdensome and far from transparent.

[12]Implantable memory takes up valuable space and current in an IMD. In contrast to commercial memory, implantable memory is specially designed and validated to be highly reliable and draw minuscule amounts of power.

Implantable Agents in Multiagent Systems

Traditionally the implantable medical device market has been focused very heavily on physical implantable technologies and products (as these drive revenue for companies like Medtronic). Telemedicine, however, underscores the increasingly important role of instruments and software. Furthermore, as IMD-based systems and software continue to develop, the need for them to be distributed as highly robust, portable and increasingly automated systems is clear. We submit that agent based approaches can be applied to address this need. The first step may be thinking of implants as agents. IMD-wearable, implanted network, and transient IMD interaction with agents in the environment are emerging interaction patterns that can be used as lenses through which opportunities for IMD systems to benefit from the agent paradigm can identified and developed.

The experimental work in this paper is meant only as an example, not as a conclusive prescription for implementation. Taking the paradigm shift to agency and thinking of implants as potential parts of MAS allows for other, and perhaps better, solutions to this and many other problems. For example, we recognize that the implantable-wearable solution to medical monitoring discussed previously has not been considered adequately in our simulations.

One potential use of implantable-wearable systems is in providing access between IMDs and other external entities. Currently the external components of such systems are primarily simple tools to enable patient interaction with implantables (e.g. a patient programmer device which can be worn on the belt). In the future, however, small wearable telemetry booster devices could act to connect the short-range, proprietary wireless systems in modern implants to remote systems via longer distance wireless schemes. Were such a system to come into place it could create potential for a very large system in which implantable agents, hospital systems, emergency systems, company systems, and perhaps even patients' personal systems are intertwined. The role of agents in this system is not certain, but their advantages in automation, etc. could well shape this potential into what may become a huge and heterogeneous multiagent system with an important fraction of its constituents taking on an implantable form. This type of approach could open up new opportunities for a richer, more reliable medical monitoring systems addressing not only continuity of data issues, but also reporting, data priority, caching, analysis automation, data integration, distributed anonymous clinical research, and other important areas. Although some of the applications of these systems could well address these various goals and concerns, we leave this for future work but examine some examples in the longer version of this paper [4].

6. CONCLUSIONS AND FUTURE WORK

The use of IMDs is a large, complex, and growing domain which naturally lends itself to applications of distributed artificial intelligence. This field is particularly appropriate for autonomous agent and multiagent system consideration as a result of its vast base of existing autonomous, distributed, largely solitary agents and emerging needs and

capabilities for these agents to work as part of multiagent systems in medical monitoring and therapy. Though work in this nascent area of overlap between AAMAS and IMDs has already begun[13] it is not thought of in these terms and could benefit significantly from more explicit consideration.

7. REFERENCES

[1] M. A. Arent and I. A. Sun Microsystems. EP1022035A1: Communication network and devices to be implanted within a subject.

[2] T. Bachmor, J. Schochlin, and A. Bolz. Transmitting patient and device data via gsm - central management for decentral mobile medical devices. *Biomedizinische Technik. Biomedical Engineering*, 47(1):346–349, 2002.

[3] A. F. DiMarco. Restoration of respiratory muscle function following spinal cord injury. Review of electrical and magnetic stimulation techniques. *Respir Physiol Neurobiol*, 147(2-3):273–287, Jul 2005.

[4] T. Drew and M. Gini. Implantable Medical Devices as Agents and Part of MAS. November 2005.

[5] L. Fauchier, N. Sadoul, C. Kouakam, F. Briand, M. Chauvin, D. Babuty, and J. Clementy. Potential cost savings by telemedicine-assisted long-term care of implantable cardioverter defibrillator recipients. *PACE*, 28(2):S255–S259, January 2005.

[6] G. Flammia. The Web: A communication medium for healthcare. *IEEE Intelligent Systems*, 17(2):88–89, 2002.

[7] H. D. Funke and I. A. Medtronic. Patent: US4987897: Body bus medical device communication system.

[8] S. D. Harpstead, L. M. Otten, T. R. Prentice, and I. A. Medtronic. Patent: US5697951: Implantable stimulation and drug infusion techniques.

[9] J. Huang, N. R. Jennings, and J. Fox. An agent architecture for distributed medical care. In *ECAI-94: Proceedings of the Workshop on Agent Theories, Architectures, and Languages on Intelligent Agents*, pages 219–232, New York, NY, USA, 1995. Springer-Verlag New York, Inc.

[10] V. Koutkias, I. Chouvarda, and N. Maglaveras. Multi-agent system architecture for heart failure management in a home care environment. *IEEE Computers in Cardiology*, 30(3):383–386, 2003.

[11] B. Larsson, H. Elmqvist, L. Ryden, and H. Schuller. Lessons from the first patient with an implanted pacemaker: 1958-2001. *Pacing Clin Electrophysiol*, 26(1 Pt 1):114–124, Jan 2003.

[12] R. P. Lesser, W. R. S. Webber, Y. Motamedi, Gholam K.and Mizuno-Matsumoto, and T. J. H. U. (Asignee). Patent: US20050149123A1: Techniques using heat flow management, stimulation, and signal analysis to treat medical disorders.

[13] S. P. Levine, J. E. Huggins, S. L. BeMent, R. K. Kushwaha, L. A. Schuh, E. A. Passaro, M. M. Rohde, and D. A. Ross. Identification of electrocorticogram patterns as the basis for a direct brain interface. *J Clin Neurophysiol*, 16(5):439–447, Sep 1999.

[14] C. Lewis. Emerging Trends in Medical Device Technology: Home Is Where the Heart Monitor Is. *FDA Consumer*, May 2001.

[15] N. Maglaveras, G. Gogou, I. Chouvarda, V. Koutakias, I. Lekka, D. Adamidas, C. Karvounis, G. Louridas, and E. Balas. Using contact centers in tele-management and home care of congestive heart failure patients : The CHS experience. *IEEE Computers in Cardiology*, 29(02):281–284, 2002.

[16] I. Medtronic. Carmel Biosensors Collaborates With Medtronic In Research And Development Of Innovative Biosensors. [Online; accessed 07-Nov-2005].

[17] I. Medtronic. Medtronic Initiates "Concerto AT" Study In United States. [Online; accessed 01-February-2006].

[18] I. NeuroPace. Implantation and testing of responsive neurostimulator (RNS) system for epilepsy. *Annual Meeting of the American Society for Stereotactic and Functional Neurosurgery*, May 2003. [Online; accessed 07-November-2005].

[19] Z. Obrenovic, D. Starcevic, E. Jovanov, and V. Radivojevic. An agent based framework for virtual medical devices. In *AAMAS '02: Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 659–660, New York, NY, USA, 2002. ACM Press.

[20] A. Ohlsson, S. H. Kubo, D. Steinhaus, D. T. Connelly, S. Adler, C. Bitkover, R. Nordlander, L. Ryden, and T. Bennett. Continuous ambulatory monitoring of absolute right ventricular pressure and mixed venous oxygen saturation in patients with heart failure using an implantable haemodynamic monitor: results of a 1 year multicentre feasibility study. *European Heart Journal*, 22(11):942–954, June 2001. Clinical Trial.

[21] M. E. Pollack. Intelligent Technology for an Aging Population: The Use of AI to Assist Elders with Cognitive Impairment. *AI Magazine*, 2005.

[22] E. K. Ritzl, E. H. Kossoff, G. K. Bergey, P. S. Coe, and D. S. Gupta. Complementing the responsive neurostimulator system with a patient operated data transmitter on demand monitoring in the outpatient environment, 2005. [Abstract; accessed 11-Nov-2005].

[23] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[24] M. H. Schoenfeld, S. J. Compton, R. H. Mead, D. N. Weiss, L. Sherfesee, J. Englund, and L. R. Mongeon. Remote monitoring of implantable cardioverter defibrillators: A prospective analysis. *PACE*, 27(1):757–763, June 2002.

[25] J. Van. Remotes keep tab on heart devices: Wireless tracking offers piece of mind. *Chicago Tribune*, November 2005.

[26] P. Walter. [Electronic visual prostheses]. *Klin Monatsbl Augenheilkd*, 222(6):471–479, Jun 2005.

[13] Consider previous examples like the Restore™IMD that works with a wearable recharger under heating constraints.

A Hotel Information System implementation using MAS technology

Armando Robles P.
Pablo Noriega B.V.
Institut d'Investigació en Intel·ligència Artificial
IIIA-CSIC.
Bellaterra, Barcelona, Spain
{arobles,pablo}@iiia.csic.es

Marco Julio Robles P.
Héctor Hernández T.
Victor Soto Ramírez.
Edgar Gutiérrez S.
TCA Research Group
Monterrey, N.L. México
{mjrobles,hhernandez,vsoto,egutierrez}
@grupotca.com

ABSTRACT

This paper reports on our progress towards a framework for enabling Intelligent Organizations with agent technologies. We have built and deployed the part of the framework consisting of organizational middleware and domain agents. The organizational middleware reads workflow scripts at run time and interprets them delegating to specialized server agents access to business rules and data bases. Those server agents, in turn, communicate with specialized user agents that facilitate human interactions through traditional plain and grid forms. We have used these ideas to transform a conventional Hotel Information System into a multi layered, agent supported information system. Our MAS-ified hotel information system is already in operation in 10 hotels with a total of 2040 rooms.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Artificial Intelligence—*Programming languages and software*; H.1 [**Information Systems Applications**]: Models and principles—*Systems and information theory, User/machine systems*; D.1 [**Software**]: Programming Techniques—*Miscellaneous*

General Terms

Applications of multi agent systems, Electronic Institutions

Keywords

Agent-oriented software engineering and agent-oriented methodologies, applications of autonomous agents and multi agent systems, Electronic Institutions

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

1. MOTIVATION

Grupo TCA is a medium-size privately owned information systems company. It has been active in the design and development of integral information systems for the Latin American market, since 1982. Its main business comes from integral systems for vertical industries such as hotels, hospitals and retailers.

Ten years ago, TCA became aware that the vertical industry market was being sensitive to the *Business Intelligence* buzzword and TCA realized that, in the years to come, its use would provide competitive advantages in spite of the exaggerated claims being made at that time. TCA decided then to anticipate this threat and started *re-engineering* its integral information systems in order to facilitate reusability of the components and making artificial intelligence resources available in the systems. The strategy was to build a business rules repository that would become readily available to user applications that were subject to a predefined workflow and could also profit from different artificial intelligence techniques. Along the way it became evident that agent technologies were a core element in that effort. Over the last 5 years, TCA implemented its Hotel Information System (HIS) as a consolidated set of business rules available to a middleware workflow engine that reads workflow definitions and delegates concrete tasks and procedures to participating user and server agents. This modestly *MAS-ified* HIS whose architecture is reported in this paper is already operational in 10 hotels with a total of 2040 rooms. It is also the boot-strapping version of an agent-pervasive HIS that we are in the process of designing.

The paper gives a quick sketch of the conceptual background of the hotel information system and a brief description of its architecture. In section 3 we outline the implemented architecture, in section 4 we discuss some relevant results and in section 5 we mention the framework that generalizes the approach we report in this paper.

2. BACKGROUND

2.1 Organizations and Corporate Information Systems

An organization, a firm, is in essence a group of individuals that pursue their collective or shared goals by interacting in accordance with some shared conventions and using their

available resources as best they can [4, 5, 2]. Hotels, or other types of organizations, have conventions that "organize" their activity in consistent ways so that employees and clients have some certainty about what is expected of them or what to expect from interacting with members of the organization. These organizing conventions usually take the form of canonical documents, standard procedures, rules of conduct or records that participants adhere to in a more or less strict way. Corporate information systems (*CIS*s) provide organizations with powerful means to capture the way the organization works by implementing, somehow, those procedures, rules, documents and records. A *CIS* controls the processes that the organization follows, stores its experience and keeps track of the interactions that the members of the organization have with other staff members and also with suppliers, clients and other individuals who are not part of the organization.

Traditionally, *CIS*s are developed around the main business processes of the organization which are usually organized in a hierarchical system whose components —as is notoriously the case among vertical industry organizations— are tightly interrelated. Consequently, building a *CIS* usually starts by making explicit the system components, that is, the business processes of the organization and their main functionalities. Once there is complete business process definition for an organization, the current practice is to address the design and development of a *CIS* by performing three types of activity: *forms designing*, *work–flow modelling* and *business rules programming*. Although there are different ways of organizing these *CIS* development activities, the following two alternative approaches are prevalent:

- *Form centered programming.* The flow of activities is governed by the design of forms. That is, the intervening business rules are invoked by fields of a form where they read or write data, in a sequence that is determined by the form design. This approach has the advantage of easy programming, but provides no facilities to implement work–flow control on the intervening processes and there is no room for capture declarative conventions that could govern the interactions between the intervening components.

- *Work-flow centered programming.* The sequence of business rule applications is specified from a work–flow perspective: the intervening business rules are invoked by states of the the specified work–flow. This approach has the advantage of having all processes in the right sequence and with the proper follow-up. However, all links between the states and also the links with both business rules and forms have to be specified and programmed at design time, resulting into inflexible implementation of processes.

We propose a third approach that is guided by a high–level specification of how the organization is supposed to function. Such *prescriptive approach* makes explicit the *institutional* aspects of the organization and makes them operational through agents that mediate the organizational interactions that constitute the *CIS* (see Sec. 5 below). By relying on agent technologies we are able to address, separately, interaction or procedural conventions, declarative or decisional conventions, and the actual operation of the IT components of the *CIS*. The intended outcome is that , in addition to the obvious modularity and reusability advantages, such separation allows the fine-tuning of the *CIS* at different levels and at different times and therefore brings larger flexibility in the design and update of the *CIS*.

As the initial motivation could suggest, our goal is to develop the type of information systems that support the so-called Intelligent Organizations with their inherent focus on knowledge management and their need to adapt to a dynamic business environment. [1] Crudely put, with this approach we mean to enable Intelligent Organizations with agent technologies by building *CIS*s that capture corporate knowledge in an effective manner in order to support the work of people (and agents) that make use of IT resources in a distributed and dynamic environment. We intend to capture the established procedures and practices that organize *CIS* user interactions through the *performative* representation of the organization as an electronic institution. We will further encapsulate organizational guidelines, policies and decision criteria into the deliberative component of staff agents that support —or take over— tasks that are currently performed by employees of the organization. Finally, keeping in mind that we want to develop *CIS*s for vertical industries and that companies in those industries have many similarities, we will take advantage of agent technologies to deal with standard process components (like forms and business rules) and conventional IT resources (e.g. databases or display devices).

2.2 The Hotel Information System

Grupo TCA deployed its first hotel information system —*INNSIST*— in 1986. Its successors have evolved over the years and are now supporting the integral operation of more than 200 hotels in Latin America. The *INNSIST* Hotel Information System (HIS) supports the management and operation for the Front and Back-office of a hotel or hotel chain. Those management and operational functionalities are implemented as twelve modules, whose specific procedures were programmed to become the content of a *business rules repository*. The following is a list of these modules and the "business context that organize their operations":

Reservations. Individual reservation, group reservations, airline crew management, rooming list, availability control, availability forecasting.

Front Desk. Guest check-in, group check-in, hotel status, credit management, guest folios, client's account statements, guest checkout.

Housekeeping. Room status, room blockade, guest requirements service, maid work orders.

Telephone control. Tariffs catalog, schedule discounts, service costs, receipt printing, telephone switch control, maid status, line management, call accounting.

Night audit. Daily room update, automatic posting, cashiers shift review, audit sheet.

[1] An *Intelligent Organization*, as Liebowitz characterizes them [3], is a "knowledge–based organization whose business operations and internal processes are founded on knowledge competencies and the value of its products and services is given by the know–how, the intellectual capital and the technological advantage of the organization".

Sales. Sale statistics, packages and plans, travel agencies, allotments control, groups, sale forecasting.

Guest History. Guest stays history, promotional plans, guest preferences.

Club Management. Resort club services: Spa, boutique, snack bar, therapies control, special plans, tennis, golf.

Accounting. Forecasting, departmental accounting, transactions, uniform system of accounts, financial models, executive financial information.

Accounts Receivables. Statements, AR projection, balance due.

Accounts Payable. Buy conditions, payments suggestion, discount advantages, partial payments, multiple payments, payments projection.

Inventory Control. Physical inventory, item location, inventory value, inventory cost, inventory level.

Purchase Orders. Order quote, authorization levels, automatic purchase order generation, automatic orders, merchandise reception.

3. ARCHITECTURE OUTLINE

The hotel information system is implemented in a two-layer framework as shown in Figure 1. The bottom layer contains the actual domain components for the hotel *CIS*: data and business rule repositories, forms and display devices, and the corresponding server agents (for domain components) and user agents (for staff and clients that interact with the *CIS*). The top layer is formed by a work-flow engine that guides the actual execution of the *CIS*, and an organizational middleware that, directed by the specified workflow, articulates users and domain components.

3.1 The middleware and workflow engine layer

The organizational middleware runs the *CIS* by putting *business domain* elements and users in contact subject to the workflow specifications. Hence, the basic functions of this middleware are:

- to log users into the organization, controlling user roles, agent resources and security issues.
- to monitor user interaction,
- to load and interpret the *workflow specification.*

The workflow engine has two components: the workflow specification language and the workflow interpreter.

3.1.1 Workflow specification language

The commands available for workflow specification are the following:

- define TAG
- define business-rule-< *id* >
- define data-base-access-definition-< *id* >
- define report-definition-< *id* >
- conditional command execute on TAG
- conditional command execute on EVENT
- form handling.
  - activate function keys
  - define plain form
  - define grid
  - plain form
    - deactivate fields
    - input fields
    - display fields
    - goto field
  - grid form
    - deactivate fields
    - input fields
    - display fields
    - goto field
- call business rule (business-rule-< *id* >)
- call data base access (data-base-access-definition-< *id* >)
- call report (report-definition-< *id* >)

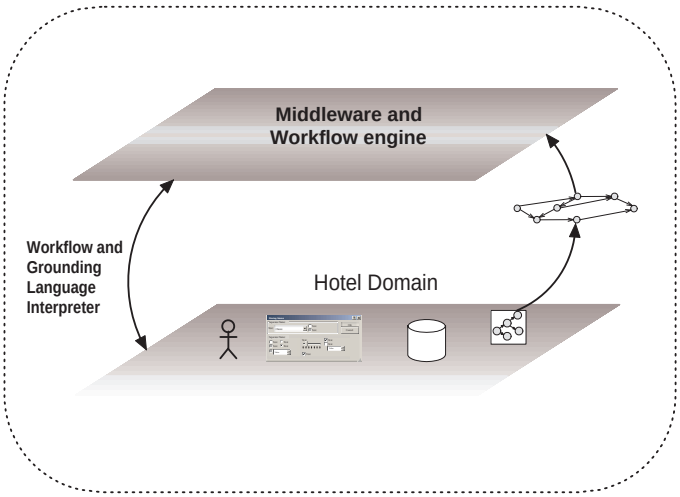


Figure 1: The workflow engine of the organizational middleware loads and interprets workflow specifications and monitors agent interactions

3.1.2 Workflow interpreter

The commands available for workflow execution are the following:

- read work flow specification
- load defaults
- initialize variables
- execute command
  - on TAG
  - on EVENT

3.2 The hotel domain layer

This layer contains the following components:

Hotel Domain Agents. We consider two types of agents:

- *User agents* (human or software) that are external users (clients, service providers and staff members of the organization) of the *CIS*, and
- *Server agents* that act as front ends for all the repositories and devices of the hotel domain and thus handle the interactions with other domain agents.

Interaction forms. These forms implement interfacing capabilities between user agents and other domain elements, e.g. form handling, data base calls, business rules triggering.

Workflow specification. Interpretable *workflow* specification that defines procedural behaviour for the organization.

Repositories. Business rules and data base repositories accessible to agents.

4. SOME COMMENTS ABOUT AGENT INTERACTIONS AND WORKFLOW EXECUTION

As indicated above, the hotel *CIS* we have implemented is run by the middleware and worfk-low engine that interacts with the hotel domain components and *CIS* users. Figure 2 illustrates how the middleware supervises the agents that handle the specialized domain components, such as data bases or business rule applications —a specialized *business rule* server agent (Bag) fetches business rules from a central repository that use data provided by another specialized *data base* server agent (Dag), to provide input to a *user agent* (Uag) that displays it in a user form.

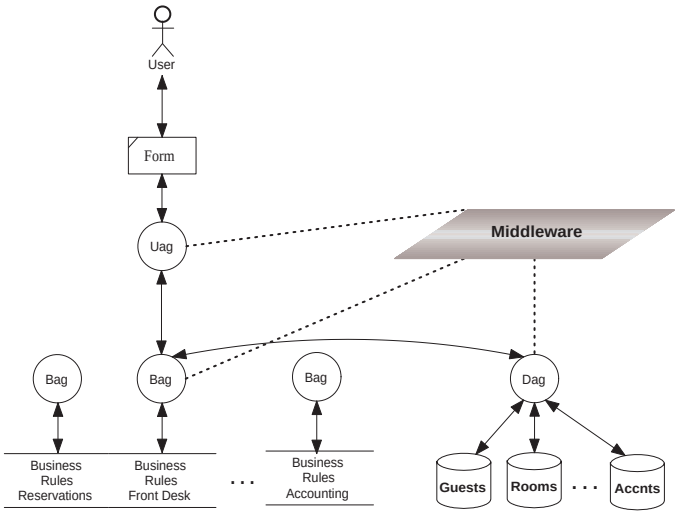


Figure 2: The middleware supervises business rules (Bag) and data base (Dag) *Server Agents* that handle all specialized tasks to serve the requirements of *User Agents* (Uag)

The middleware acts as a link between the user interface and the data base and business rule repositories, but all interactions are mediated by ad-hoc agents. The user interface is mediated by a user agent that is regarded as a client for the business rule and data base server agents. The main function of the specialized server agents is to act as a business domain components (including business rule repositories and data bases) facilitators for all user agents that may be logged-in at several client sessions. User agents can be located anywhere and in any platform; they may be implemented as independent applications, even as applications that require browsers for operation. All the communication with the middleware is made using a Tag-Value encoding via a XML protocol.

Once the interaction between the *user agent* and the *server agent* is established, the infrastructure does nothing but assuring that the communication between both agents is persistent until one of the agents decides to terminate. This is a very important point, because it is the responsibility of the agent programmer to consider all the communication contexts and to include a piece of code to deal with all the information interchange requirements. For example, in the workflow definition shown in Appendix A, once the F4 event is true and the workflow interpreter issues the command
`Interact(DataBaseSvrAg(DB_SUPPLIERS,NEW))`
it is the responsibility of the programmer to send all relevant information from the user form to the data base server agent in order to have the data base properly updated.
In a similar way, once the `SELECT_SUPPLIER` event becomes true, the workflow interpreted issues the command
`Interact(BusinessRuleSvrAg(BR_SELECT_SUPPLIER))`
and it is also the programmer's responsibility to trap the result of the business rule execution setting the value for some data or status variables. The programmer is responsible for maintaining the context of all agent interactions because as agent interaction evolves, they modify the context of the world, updating data and status variables as required. All the global variables are available in the scope of the workflow definition, that is, in the workflow specification the programmer can test for the value of variables defined as global by any *Server Agent*.

The workflow interpreter and the middleware environment impose no control over variables, neither global or local. That is, it is the programmer's responsibility to define and maintain the proper scope for the required variables. The middleware assumes that all agent interactions are between the User Agent and either the business rules server agent or the data base server agent.

Regarding workflow execution, tag and/or event value verification takes precedence over sequential process execution; that is, in the middle of a conditional execution of a TAG value, it is possible to break the sequential flow and skip directly to the first command of another conditional clause.

Regarding scope of workflow execution, once a flat form or grid is addressed, all subsequent workflow commands will be made in the scope of that specific flat form or grid, until another flat form or grid is addressed.

This system developed so far, however, has one major limitation: the workflow engine has no control over what is said between agents, as it deals only with specific conditional commands that test for contextual changes represented by changes in data and status variables. If we want to deal with complex interactions, this is an important limitation

because (up to now) we are forced to "hardwire" the control code for the execution of alternative procedures depending on what is said between agents.

In the implementations described in this paper, we only use *reactive* agents. Such a primitive implementation is enough for our current needs but we may readily change their specification to involve more sophisticated behavior. The benefits provided by the fact that they may potentially negotiate their information interchanges in a flexible way are evident. We should also note that our implementation of agents already assumes that they interact with other agents exclusively through speech acts. Such design decision will enable us to make them functional in the electronic institution environments that are under construction for the extended framework.

The middleware language outlined in this paper is being refined and extended to deal with new functionalities. We also want to go one step further and implement the concept of commitments between agents in order to provide the framework with the flexibility needed to deal with temporal constraints, that is, the effect of some interaction between agent A and agent B at time i, might commit agent B to do something at time $i + 3$, say.

The system described in this paper is currently operational in the following hotels:
– Avalon Grand de Cancún (154 rooms)
– Avalon Reef –Isla Mujeres (137 rooms)
– Blue Bay Cancún Club (310 rooms)
– Blue Bay Cancún Getaway (290 rooms)
– Blue Bay Desire Resort (110 rooms)
– Coco Beach Cancún (202 rooms)
– Karmina Palace Manzanillo (322 rooms)
– Mayan Palace Mazatlán (280 rooms)
– Radisson Casa Grande Chihuahua (115 rooms)
– Reef Club Isla Cozumel (120 rooms)

5. CLOSING REMARKS

In this paper we describe an agent-enabled information system that is being used to manage and operate medium sized hotels. In this system, agents are used as mediators between *CIS* components and system users under the control of a workflow engine.

We claim that the example indicates how current *CIS* development practices may be improved with:

- An Intelligent Organization environment that enables available information technologies resources (such as data bases, business rule repositories, data mining tools and automated decision making devices) with *MAS* technology.
- An agent oriented methodology for building complex information systems.
- Facilitating the separation of form design, workflow specification, and the programming of business rules and agent behaviour.

We are developing a framework that generalizes this example. More specifically, in order to deal with complex interactions, we want to use the theory and notions of organizations and institutions to implement prescriptive specifications that may be properly enacted. We are working with the notion of Electronic Institutions and extending the current IIIA EIDE environment to allow teleological and normative specification of agent interactions [1]. The purpose of the framework is to facilitate the building of *CIS*s that implement a high-level prescriptive specification of an organization by encapsulating the institutional elements of the organization through standard procedures and corporate decision-making criteria encapsulated in (corporate) user agents. We intend our framework to allow for the construction of *CIS*s that are flexible enough to adapt to changing requirements and business conditions. The system discussed in this paper will allow the boot-strapping of the proposed framework because, on one side, the the middleware layer developed so far is being extended to deal with richer electronic institutions and, on the other side, the workflow language and interpreter, as well as the user and server agents developed so far are being extended to provide more general *grounding language* and domain agents.

Figure 3 shows our proposed extended framework. The diagram shows how our present middleware layer is to become an organizational engine that implements an electronic institution —not just a workflow engine— that controls the business domain objects. That electronic institution engine, in turn, is the implementation of a declarative description of how agents should interact or, more precisely, of the institutional conventions that will govern the business domain interactions. These developments have been partially reported already. We have reported the implementation of a *Bayesian reasoning engine* [6] and the use of this architecture in our work on *Problem Solving Plans* [8]. Finally, in [7] we have discussed a conceptual framework and outlined an architecture and its integrative elements to enact an Intelligent Organizations using the Electronic Institutions approach.

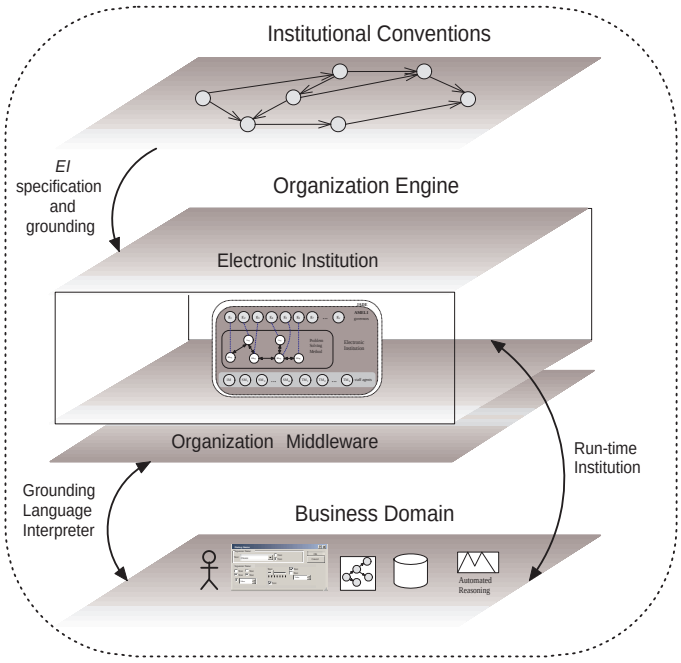


Figure 3: General Architecture using the concept of Electronic Institutions as part of the organization engine

Acknowledgments

This research is partially funded by the Spanish Ministry of Education and Science (MEC) through the Web-i-2 project (TIC-2003-08763-C02-00) and by private funds of the TCA Research Group.

6. REFERENCES

[1] J. Arcos, M. Esteva, P. Noriega, J. Rodriguez-Aguilar, and C. Sierra. Environment engineering for multiagent systems. *Engineering Applications of Artificial Intelligence*, (18):191–204, Elsevier Ltd. January 2005.

[2] R. M. Cyert and J. G. March. *A behavioral theory of the firm*. Prentice-Hall, Englewood Cliffs, N.J. USA, 1963.

[3] J. Liebowitz and T. Beckman. *Knowledge Organizations*. Saint Lucie Press, Washington, DC, 1998.

[4] J. G. March and H. A. Simon. *Organizations*. John Wiley and sons, New York, USA., 1958.

[5] D. C. North. *Institutions, Institutional change and economic performance*. Cambridge Universisy press, 40 west 20th Street, New York, NY 10011-4211, USA, 1990.

[6] A. Robles, F. Cantú, and R. Morales. A bayesian reasoning framework for on-line business information systems. In *Proceedings of the Eleventh Americas Conference on Information Systems*, Omaha, NE, USA, August 2005.

[7] A. Robles and P. Noriega. A framework for building eienabled intelligent organizations using mas technology. In M. Gleizes, G. Kaminka, A. Nowé, S. Ossowski, K. Tuyls, and K. Verbeeck, editors, *Proceedings of the Third European Conference in Multi Agent Systems (EUMAS05)*, pages pp.344–354., Brussel, Belgium, December 2005. Koninklijke Vlaamse Academie Van Belgie Voor Wetenschappen en Kunsten.

[8] A. Robles, P. Noriega, F. Cantú, and R. Morales. Enabling intelligent organizations: An electronic institutions approach for controlling and executing problem solving methods. In A. Gelbukh, Álvaro Albornoz, and H. Terashima-Marín, editors, *Advances in Artificial Intelligence: 4th Mexican International Conference on Artificial Intelligence, Proceedings ISBN: 3-540-29896-7*, pages 275 – 286, Monterrey, NL, MEX, November 2005. Springer-Verlag GmbH. ISSN: 0302-9743.

APPENDIX

A. WORKFLOW SPECIFICATION

```
// Workflow definition for the input and
// maintenance of suppliers price lists
#define_tag SUPPLIER                    1
#define_tag SUPPLIER_NAME               2
#define_tag DUE_DATE                    3
#define_tag DISCOUNT                    4
#define_tag PROMOTION                   5
#define_tag CONTACT                     6
#define_tag GRID_FIELDS                 7
#define_tag END_FIELDS                  7
#define_tag TOTAL_COLUMNS               7
#business_rule BR_CONSISTENCY_CHK      3000
#business_rule BR_SELECT_SUPPLIER      3001
#business_rule BR_SELECT_ARTICLE       3100
#data_base_access DB_ARTICLE_LIST      3200
#data_base_access DB_SUPPLIERS         3300

//
// Actions on TAG value
//

process on TAG = 0 {
   ActivateFunctions(F1);
   DeactivateFields(SUPPLIER,CONTACT);
   DeactivateGrid("grid01");
   GridClearFields(GRID_FIELDS,GRID_FIELDS);
   InputFields(SUPPLIER,SUPPLIER);
}

process on TAG = SUPPLIER {
   ActivateFunctions(F2);
   DefineGrid("grid01");
   InputFields(DUE_DATE,CONTACT);
}

process on TAG = SUPPLIER and SupplierExists {
   ActivateFunctions(F1);
   DeactivateFields(SUPPLIER,CONTACT);
   GridDeactivateFields("grid01");
   GridStartInput(SUPPLIER,TOTAL_COLUMNS);
   InputFields(DUE_DATE,CONTACT);
}

process on TAG = PROMOTION {
   ActivateFunctions(F1|F2);
   // Read and show Supplier price list
   Interact(DataBaseServerAgent(DB_SUPPLIERS,SHOW));
   DefineGrid(Parameter);
   GridDisplayFields("grid01");
   GridStartInput(SUPPLIER,TOTAL_COLUMNS);
}

process on TAG = END_FIELDS {
   GotoField(DUE_DATE);
}

//
// Actions on EVENT
//
```

```
process on EVENT = F99 {
   Exit;
}

process on EVENT = F2 {
   ActivateFunctions(F3|F4|F5)
   DeactivateFields(SUPPLIER,CONTACT);
   DeactivateGrid(601);
}

process on EVENT = F3 {
   ActivateFunctions(F1);
   InitializeVariables();
   DefineGrid("grid01");
   DeactivateGrid("grid01");
   GridClearFields(GRID_FIELDS,GRID_FIELDS);
   InputFields(SUPPLIER,SUPPLIER);
}

process on EVENT = F4 {
   // Validate if all data is consistent, then
   // add the supplier's price list
   Interact(BusinessRuleServerAgent(BR_CONSISTENCY_CHK));
   Interact(DataBaseServerAgent(DB_SUPPLIERS,NEW));
   InitializeVariables();
   ActivateFunctions(F1);
   DefineGrid("grid01");
   DeactivateGrid("grid01");
   GridClearFields(GRID_FIELDS,GRID_FIELDS);
   InputFields(SUPPLIER,SUPPLIER);
}

process on EVENT = F5 {
   ActivateFunctions(F2);
   InputFields(DUE_DATE,CONTACT);
}

process on EVENT = F6 {
   // Delete supplier's price list
   Interact(DataBaseServerAgent(DB_SUPPLIERS,DELETE));
   InitializeVariables();
   ActivateFunctions(F1);
   DefineGrid("grid01");
   DeactivateGrid("grid01");
   GridClearFields(GRID_FIELDS,GRID_FIELDS);
   InputFields(SUPPLIER,SUPPLIER);
}

process on EVENT = F8 {
   GridStartInput(GRID_FIELDS,GRID_FIELDS);
   ActivateFunctions(F1|F2);
   InputFields(DUE_DATE,CONTACT);
}

process on EVENT = SELECT_SUPPLIER {
   // Show Supplier list and select one
   Interact(BusinessRuleServerAgent(BR_SELECT_SUPPLIER));
}

process on EVENT = SELECT_ARTICLE {
   // Show Article list and select one
   Interact(BusinessRuleServerAgent(BR_SELECT_ARTICLE));
}
```

Commercial Applications of Agents: Lessons, Experiences and Challenges

Roxana A. Belecheanu, Steve Munroe, Michael Luck, Terry Payne
School of Electronics and Computer Science
University of Southampton
Southampton, United Kingdom
{rab2, sjm, mml, trp}@ecs.soton.ac.uk

Tim Miller, Peter McBurney
Department of Computer Science
University of Liverpool
Liverpool, United Kingdom
{t.miller, p.j.mcburney}@csc.liv.ac.uk

Michal Pěchouček
Department of Cybernetics
Czech Technical University in Prague
166 27 Prague 6, Czech Republic
pechouc@labe.felk.cvut.cz

1. INTRODUCTION

As has been argued very eloquently and effectively, there is a *chasm* that needs to be crossed in the adoption of any new technology [8], and the marketing of such technologies must somehow try to bridge the gap that arises. Many currently see agent technologies in the middle of that chasm period of adoption, where addressing the visionaries as well as the pragmatists (or early adopters) who avoid risks but readily see the advantages of tested technologies [11], is necessary. As part of the needed marketing exercise, we believe that a catalogue of case studies of deployed applications with real quantified business benefit, can help convince those wavering pragmatists. Perhaps more importantly, the lessons that can be drawn from such case studies may also be used to guide the efforts of new commercial agent technology providers, both in developing the technology itself, and in understanding the concerns and constraints of early adopters.

As part of its industrial activity programme, AgentLink III[1] has been developing such a catalogue of case studies for exactly this purpose, working in collaboration with a range of companies to provide a selection of example applications, and draw out important issues arising from their development. In this paper, we summarise some of these case studies, and review the lessons learned, with the aim of identifying key problems. Due to space constraints, only very brief descriptions are provided in each case, but they serve to illustrate the nature of the application and the range of sectors and solutions; the full set of case studies is available from www.agentlink.org.

[1]http://www.agentlink.org

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

2. COMMERCIAL AGENT APPLICATIONS

To date, deployed applications of agent technologies have been concentrated in a small number of industrial sectors, and for particular, focused, applications. These have included: automated trading in online marketplaces, such as for financial products and commodities; simulation and training applications in defence domains; network management in utilities networks; user interface and local interaction management in telecommunication networks; schedule planning and optimisation in logistics and supply-chain management; control system management in industrial plants, such as steel works; and, simulation modelling to guide decision-makers in public policy domains, such as transport and medicine [5].

While some of these applications may be implemented as closed systems inside a single company or organisation (for example, agent-based simulation for delivery schedule decision-making), the domains to which agent technologies are most suited are those involving interaction between entities from more than one organisation. For example, automated purchase decisions along a supply-chain, require the participation of the companies active along that chain, so that implementing a successful agent-based application requires agreement and coordination from multiple companies. Moreover, to fully exploit the benefits of agent technology, other suitable domains are those involving a multitude of variables, with complex interdependencies, which change dynamically and thus cannot all be known at design time. This requires agent solutions because of their ability to dynamically adapt to changes in the environment and thus to offer real-time optimisation. For example, agent-based solutions successfully replaced conventional systems for network planning and transportation optimisation, because these were limited in their ability to cope with the increasing complexity, and especially with the dynamics, of a noticeably globalised transportation business [3].

In this section we describe a suite of agent applications provided by agent technology vendors for several commercial clients. In particular we describe the work of the following companies.

- Agent Oriented Software and their system used to aid

the Ministry of Defence in military training.

- Eurobios and the simulation model they developed to improve production schedules for a cardboard box manufacturer.
- Magenta and their approach to helping a shipping company improve their oil distribution shipping network.
- Whitestein's system for logistics management.
- Nutech and the solution they developed for an energy production and distribution company.
- Acklin's approach to solving information exchange problems for insurance companies.
- Rockwell Automation and their use of intelligent agents for chilled water systems for the US Navy.

2.1 Agent Oriented Software and Military Training

Simulation and modelling are extensively used in a wide range of military applications, from development, testing and acquisition of new systems and technologies to operation analysis and provision of training and mission rehearsal for combat situations. In particular, distributed simulation environments for use in training and analysis drive much of this work. For example, simulation exercises involve a few dozen real people with the remaining hundreds or even thousands of other battlefield entities being computer simulations. As the human element is a key success factor in combat situations, the value of computer models of combat are greatly affected by their ability to accurately represent the range and variability of expected human behaviour.

The Human Variability in Computer Generated Forces (HV-CGF) project, undertaken by Agent Oriented Software on behalf of the UK's Ministry of Defence (MoD), developed a framework for simulating behavioural changes of individuals and groups of military personnel when subjected to moderating influences such as caffeine and fatigue. The project built on the JACK Intelligent Agents toolkit, a commercial Java-based environment for developing and running multi-agent applications.

JACK was particularly appropriate for several reasons. It incorporates the Belief-Desire-Intention (BDI) reasoning model, proven to effectively model certain types of behaviour, like the application of standard operational procedures by trained staff. JACK also includes facilities for the creation of intelligent team behaviour through the advanced JACK Teams model, while the JACK Agent Language extends Java with constructs for agent characteristics, such as plans and events.

For the project, Agent Oriented Software built a layer on top of Jack, called the JACK Cognitive Architecture (CoJack), that provides cognitive modelling capabilities for enhancing agent functionality with psychological attributes. Using these attributes, cognitively-plausible agents can be developed to simulate human psychological parameters, such as the capacity of human memory to handle multiple concurrent tasks, the perception and speed of reaction to events from the battle-space, and the ability to choose from different problem-solving tactics. This layer can thus be used to develop various cognitive models.

The influence of internal and external moderators are modelled as an independent layer, through a set of *behaviour moderators*, one for each influencing factor that modifies the agent's psychological attributes. These were plugged directly into the JACK Cognitive Architecture.

Also part of the core architecture is the JACK Intelligent Agents Team-based Platform, which was implemented using JACK's team based agent capability, to model human variability at team level. It facilitates the design, configuration and execution of a collection of agents organised in military structures realistic to the UK army. Each team member is a rational agent able to execute actions such as doctrinal and non-doctrinal behaviour tactics, which are encoded as JACK agent graphical plans.

2.2 Eurobios and Manufacturing Management and Optimisation

SCA Packaging, a leading international manufacturer in the corrugated-box market, turned to Eurobios to provide an agent-based simulation model to explore different strategies for reducing stock levels without compromising delivery times, as well as evaluating the consequences of changes in its customer base. In order to operate successfully in a highly competitive market, the manufacturing company found itself under growing pressure to maintain a high quality of customer service and decrease delivery times, as well as increasing plant efficiency and reducing inventories of finished goods

Eurobios developed a solution for the management and optimisation of the company's manufacturing plant. In the plant, customer orders arrive simultaneously for a large variety of boxes, each with its own colour scheme and specific printing, and often to be delivered at very short notice. Plant managers have to deal with varying lead-times, varying seasonality and overlapping production pathways, while trying to avoid weekend shifts and overtime, keeping the warehouse stock levels down, and ensuring *on-time and in full delivery* [2]. In addition, the complexity of plant processes, combined with the difficulty of predicting customer behaviour and machine failure, makes managerial decisions difficult. Thus, the problems and the trade-offs arising in the management of the plant require a detailed understanding of the relationships between customer order patterns, factory capacity, machine speeds, order batching and warehouse size (to mention just a few of the relevant variables), which can only be achieved through the use of sophisticated simulation technology.

The simulation model developed by Eurobios combined agent technology with discrete event simulation [2]; agents represented interacting entities with behaviour rules and decision capabilities attached, while the simulation of the processes was achieved using events, occurring concurrently or consecutively.

2.3 Magenta and Vessel Scheduling

As in many other transportation domains, in the domain of shipping, a major problem is efficient cargo allocation and vessel scheduling. Commonly, such tasks are performed manually by human schedulers with no automated support. However, this causes two major problems, as follows.

1. The vast amount of information that must be considered to schedule large fleets of vessels in a dynamic

environment means that the complexity is high, potentially leading to costly mistakes. Reducing the frequency of such mistakes improves fleet utilisation, allowing increased profits while using the same resources.

2. There are many domain-specific constraints that impact on the scheduling process and, each time an employee involved in scheduling leaves the company, their expertise in relation to these domain-specific constraints leaves with them; reliance on human schedulers alone brings great dependencies causing the company to become inefficient, and requiring new employees to start building up expertise from scratch.

In the scheduling of ships, many factors must be taken into account. First, each vessel, cargo, and port has different constraints that may prevent certain schedules (for example, a vessel may be too big for a port, or a cargo too big for a vessel). Such constraints are *hard* in that they cannot be broken during the scheduling process. However, other *soft* constraints must also be considered. These are constraints that the scheduler can break only if there are no other reasonable solutions available such as, for example, reducing the amount of time that a vessel travels without cargo. In addition, the environment is dynamic, in that the cargo is changing constantly, and the vessels may fail, rendering them out of service for a period, and requiring a reschedule for their cargo over that time.

In addressing these concerns for a large shipping company, Magenta Technology developed the *Ocean i-Scheduler* vessel scheduling tool [4], which was built on top of Magenta's agent technology, using the multi-agent paradigm. The shipping company's original requirement for their application was simply an online system to allow direct booking of cargo to be transported. However, after discussions with Magenta and demonstrations of their agent technology, it was clear to the company that agents might enable the development of a much more useful system. A proof of concept application developed by Magenta convinced the customer to invest in the development of a full scale system. The customer was first impressed by the way that, using agent technology, the system's behaviour could be changed without having to shut it down or introduce new code. Second, they recognised how well the agent system itself could handle changes to cargo, vessels, and ports. Overall, the customer envisaged two benefits to implementing their system in this way.

1. The system would provide support to employees in the scheduling task, therefore reducing the possibility of costly mistakes.

2. The system would provide a formal model of the business knowledge needed to schedule shipping fleets, so that much less knowledge might be lost when scheduling employees leave the company.

In the development of the system, the client, together with Magenta Technology developers, used the agent paradigm also as a design tool. Each cargo, vessel, port, etc., was modelled as a real-world role. For example, the role of vessel was designed, taking into account the responsibilities and rights of such a vessel. Then, for each vessel in the client's fleet, a vessel agent was created. The developers claim that this led to a simplified design when compared to what might have been developed using traditional software engineering techniques.

2.4 Whitestein and Transportation Networks

In transportation, one of the keys factors affecting efficient operation is the optimisation of the use of available capacity. Strategic network planning and short-term route optimisation are used to allocate despatching plans but are normally performed manually, causing difficulties when plan deviations become necessary due to the occurrence of unexpected events.

The Living Systems Adaptive Transportation Networks (LS/ATN) application is a cost-based optimisation system for transport logistics [3]. Developed by Whitestein Technologies, LS/ATN is designed to provide automatic optimisation for large-scale transport companies, taking into account the many constraints on their vehicle fleets, cargo, and drivers. Although the agent solution accounts for only 20% of the entire system, agent technology plays a central role in the optimisation. Vehicle drivers send information specifying their location and proposed route, and the system determines if that vehicle can collect an additional load, or swap future loads with another vehicle in order to reduce cost. A negotiation is performed automatically by agents, with each agent representing one vehicle, using an auction-like protocol. The vehicle that can provide the cheapest delivery wins the auction, reducing the overall cost of cargo delivery and, in most cases, the combined distance travelled for all vehicles. The aim is to find a local optimum (that is, not European-wide), so that only vehicles travelling in close proximity to each other will be involved in negotiations.

2.5 NuTech Solutions and Plant Optimisation

Recent years have seen substantial change in the energy transformation market caused by deregulation, fluctuating energy prices, changes in customer demand and changes in production and delivery costs. In this context, a large energy supply company sought out NuTech Solutions to develop a system that would enable them to address a number of key concerns. Essentially, the company needed to optimise its operations so that it could schedule the production and distribution of liquid oxygen and liquid nitrogen from more than 40 plants to 10,000 customer sites.

The Supply Chain Production Optimiser system developed by NuTech combined the power of ant system optimisation and genetic algorithm search techniques, and is one of the largest and most successful applications of these techniques used in industry to date. The system takes into account the production schedules of all of the company's oil production plants and adapts them to projected energy prices, weather changes, client demand and desired inventory levels. Utilising data representing seven day power prices, customer demand projections, daily power costs and efficiency measures from every plant in the network, the system evaluates production costs based on forecast demand to determine which plants should produce which products. In addition, the system can model and take into consideration potential plant production capability, maintenance needs and power issues that may impact upon the generated schedules. All of this information can then be fed into the energy company's *operations control centre* to enable adjustments to be made to the operation of the plants to meet production requirements across the whole network.

2.6 Acklin and Information Management for Insurance

The process of handling transnational insurance claims is heavily bureaucratic, with most of the information processing done by humans, resulting in long times for settlement. European insurance companies use heterogeneous information systems, with different means of storing and using data, so that the data must be exchanged manually between companies when dealing with claims, which involve motor vehicle accidents in one country by drivers from another. Acklin, B.V. developed an agent-based system to automate the exchange of insurance information between human operators in the claim handling process [10] concerning trans-national vehicle accident claims. The main customer was a major Dutch insurance company, but other insurance companies involved in the claim handling process with this customer needed also to implement this system for interoperability reasons.

The KIR system developed by Acklin consists of a network of communicating agents to which the back-office of every insurance company can connect in order to exchange information. The communication between agents is done by email, as a means of message transport, because it was the only technology that was supported by all the insurance companies implementing this automation solution.

The agents were designed and implemented using the Five Capabilities model, also known as conceptual separation of concerns, which is composed of the following models:

The communication model handles all the interactions between agents and other systems, and it has methods for message transportation, representation and interpretation.

The competence model contains methods for the execution of the tasks for which the agent is designed.

The self model represents the knowledge that an agent has about its own capabilities (i.e. goals, tasks, jobs, states and competencies).

The planner model contains planning strategies that can be used to achieve the goals.

The environment model gives the agent a view on the world in which it operates (i.e. the other agents) and with which it interacts.

The advantage of this approach was that each model could be implemented independently, to reflect the type of agent; for example, for agent roles like payer and handler, the differences in capabilities were implemented only in the competence and self models.

The introduction of the KIR system resulted in a work pressure release of three people at the customer site and reduced the process of identification of client and claim information from 6 months to 2 minutes. The success of this system led the customer to request its implementation at European level.

2.7 Rockwell Automation and Chilled Water System

For many years now, Rockwell Automation has been investigating the use of multi-agent systems in manufacturing and various control and diagnostics applications [7]. Based on the Autonomous Cooperative System (ACS) proprietary agent platform [6], Rockwell Automation developed an intelligent application for distributed control of a shipboard Chilled Water System (CWS) for vessels of the US Navy [9].

In the CWS application, intelligent agents are deployed within commercial off-the-shelf (COTS) Programmable Logical Controllers (PLCs). Specifically, the set of functionalities of the standard Rockwell Automation ControlLogix and FlexLogix controllers was extended so that they can host a collection of various intelligent agents. The deployed agents carry out negotiations and collective decision-making aimed at dynamic reconfiguration of the Chilled Water System and fluid rerouting as a result of possible failures in the system.

The CWS application has been demonstrated on the Reduced Scale Advanced Demonstrator (RSAD) model, a Rockwell Automation developed reconfigurable fluid system test platform, which is a scaled-down version of a real ship. It features two chilling plants with 16 chilling targets on board the ship (for example, combat systems, communication systems, and radar and sonar equipment). Each water cooling plant is represented by an agent, as well as each chilling target(heat source), each valve, and some parts of the piping system.

3. LESSONS LEARNED

Having presented an overview of these agent-based applications, we now turn to discuss the main lessons learned from their development and deployment, and from the successful collaborations between the software companies and their industry clients. In this section, we present some of the challenges encountered and the ways in which they were overcome, in terms of selling the technology to the customer, knowledge acquisition and knowledge engineering, integration with legacy systems, validation, usability and operationalisation, and standards.

3.1 Selling Agent Technology

One of the important factors in securing a contract for an agent-based application was selling the agent technology to the right customer in the right industry sector. To achieve this, it was generally necessary to focus on industries who were forced to transform under competitive pressures and on companies who were looking to streamline some of their processes in order to reduce costs and become more efficient. Not only did agent technology need to address these demands, but it had also to deliver visible business or commercial value to the customer and sometimes even to their cooperating partners. In particular, success in securing such contracts often required helping companies to achieve a more profound transformation of how they operate and not just an incremental improvement.

Because of the disruptive nature of agent technology, a good relationship with customers was also important, as was having a history of successful implementations of agent-based systems, and experience of modelling of the particular problem domain. For example, the extensive experience of Eurobios with supply chain optimisation using agents was valuable in convincing managers at SCA Packaging of the feasibility of an agent-based solution. The alternative model was to find *pioneering* customers, or *early adopters* who understood the potential business value of the agent-based application and were willing to invest in it (such as the insurance company in the Acklin case study and the MoD),

even if they did not have competence in the specifics of the technology itself.

The agent metaphor itself was also useful in attracting customer interest, as it was found to be an intuitive way of modelling some business domains (for example, in the insurance case and the packaging plant) and easier to understand than analytical or mathematical models typically used for optimisation. With concepts such as roles and responsibilities, agent-oriented approaches to problem and system description are much closer to the ways in which managers conceive of business domains than are traditional software engineering descriptions. By mapping organisational functions onto agents using roles and capabilities, and by mapping business logic and process flows onto message exchange patterns between agents, it was easy to reason about and better understand company processes, especially when visualisation was used in early model-building stages. This aspect particularly appealed to senior management and process engineers, and made it easier to convince them to invest in the agent-based solution, even when the customer had no prior exposure to agents. The same cannot be said about using agents for cognitive modelling for the first time in a defence application, where building cognitive science concepts on top of agent concepts, and then linking back and interpreting simulation results according to cognitive principles was problematic. However, by successfully completing the project and demonstrating the feasibility of agent technology in modelling moderated behaviour in distributed simulations, further interest was raised within the defence community in applying agents to other types of simulations, thus pushing agent technology towards exploitation and development of new areas.

3.2 Knowledge Acquisition and Knowledge Engineering

In modelling and simulation applications for planning, such as the corrugated-box factory modelling of Eurobios, the cargo fleet scheduling of Magenta, or the production planning and routing optimisation of NuTech, knowledge acquisition and model building are often performed simultaneously and iteratively. To build a model that matches reality as accurately as possible requires gathering information about different operations and processes from documents and company databases, as well as eliciting and formalising knowledge from domain experts. For large scale and complex problems, the model is built incrementally, in several steps, from a simple basic model through to a detailed and complex one that approaches reality. The model and underlying knowledge are then validated by experts, after which more knowledge acquisition is undertaken to refine or correct the rules where necessary.

These tasks pose particular challenges but also offer substantial benefits both to software developers and customers. For business performance optimisation systems, for example, data acquisition often requires significant effort in companies where data is spread between different enterprise applications. This effort is even higher when there are no existing integration applications (such as enterprise resource planning, or ERP) in a company, and where the enterprise applications do not automatically generate the information necessary for optimisation. Such is the case of transaction-based applications, (as experienced by Eurobios at SCA Packaging, for example), which are not designed for storing intermediate states of information between transactions or for storing the events that model the business processes; neither are they able to store other types of information like the time of generating the data, or what-if analysis information, typically used in the modelling and optimisation task.

In terms of eliciting domain knowledge, specific challenges were encountered.

- First, the process of making explicit that knowledge which is complex, unformalised and developed through years of experience by people working in the field was one of the most challenging and time-consuming aspects of the acquisition task in some of the cases we considered.
- Second, clarifying domain terminology to the software developers was often necessary and was clearly very important, especially in the early stages.
- Third, the effort to obtain data of sufficient quality and detail was sometimes underestimated, and several iterations were necessary because the software engineers and business managers often had a different appreciation of what it is that constitutes good data.
- Additional problems came from the fact that domain experts often had little understanding of, or interest in, agent technology, and had limited time available for discussion (as in the case of the defence application).

Sometimes minor alterations to the documented processes were introduced on an *ad hoc* basis at an operational level without informing management. For example, changes to delivery dates for a particular customer were introduced by plant operators at SCA Packaging, based on their everyday understanding of customer order fluctuations. Consequently, there were often discrepancies between the results of simulations and real data from the domain (such as the warehouse stock levels in the case of the cardboard-box factory), which had to be corrected.

However, discovering such discrepancies was one of the benefits of the knowledge acquisition and engineering activities. Indeed, the impact was broader than just the software model, and reached beyond the realm of software development. Not only did different departments gain an insight into each other's work practices, but by encouraging them to talk to each other in the process of eliciting knowledge, some of the communication barriers between functional departments and between different management roles in the company were eliminated.

3.3 Integration with Legacy Systems

A key concern of many companies when deploying and integrating new IT systems is data security and the stability of the systems already in place (or legacy systems). For these reasons and because of the current early stage of agent technology adoption, agent-based applications for simulation and optimisation were used as tools to assist human decision-making and not operationally to control or automate processes. The only integration with legacy systems was therefore at a database level and not with any planning or scheduling applications in use at client companies. The only example of process automation with agents was the exchange of information in insurance claim handling, but no actual integration with insurance systems was present here

either. A contrasting case is the Co-JACK system developed by AOS, where the agent-based system was integrated with other simulation environments in order to take advantage of well developed user interfaces and high-fidelity graphical modelling of various geographical environments. Two main integration approaches were identified, as follows.

- The vertical approach, used by AOS, by which the Co-JACK agent system was built using a layered architecture, and the bottom layer of the architecture (the CGF Interconnection Layer) was used for integration with existing defence CGF systems (such as the OneSAF Test Bed).
- The horizontal integration approach, by which every agent interfaces with company databases, was used by Acklin. In order to minimise the interference between agents and the back-office of the insurance company, Acklin developed an interfacing component that maps execution commands from an agent to the back-office, and returned results from the back-office to the agent. Transferring data between agents and the interfacing component was done through a buffer where instructions and reports are read and written by agents. This solution enabled agents to have controlled, but not direct, access to the required functionality and data from the legacy system. The interfacing component was developed by the IT department of the insurance company, thus sharing the risk with the client and allowing the client to retain control over company systems and data.

3.4 Validation

Validation is crucial to ensure the accuracy, correctness and realism of any model and of its outputs. Due to the complexity of the processes being modelled, extensive tests are usually necessary for diagnosing problems and tracking system behaviour, not only for validation purposes but also to help the customer understand how the system works. However, the task is challenging, not least because agent-based systems may explore realms of behaviour which go beyond people's intuition and expectations and therefore results often seem surprising [1]. Testing and validation are even more problematic when the system must access company databases for data input in real-time and when this interaction occurs over multiple steps (for example, in the Eurobios case), because simulations become irreproducible, making diagnosis and debugging more difficult.

In addition, questions of how much time and effort to spend in system design and model building, as opposed to implementation and validation are common difficulties, and were experienced by Eurobios. On the one hand, spending considerable time in advance to ensure sufficient data accuracy, and designing the application to ensure reusability, quality and robustness was important. However, it was also found that only through simulations during the model calibration and validation stages did certain issues surface that would not have been discovered at design time. For reasons of complexity, there was a point after which spending more time in design brought no further value to the final system because there was a chance that what was being designed would later be invalidated.

In contrast, in the case of the Co-JACK system, experiences with model validation and overall system testing were different and more challenging, firstly because they had to address the problem of integration with the CGF system, and secondly because the realism of human behaviour modelling had to be proven without being able to test the system in an actual combat situation. Testing was therefore achieved by showing how changes in psychological variables within Co-JACK determine changes in the behaviour of the entities in the OneSAF system, thus testing the link with the CGF system. The agent-based model was validated by comparing the results of the simulation with those produced by other cognitive architectures that had previously been validated through experiments with humans.

3.5 Usability and operationalisation

Training and familiarisation play a vital role in the adoption of any new technology; this is true for agent technology, too. When an agent-based system is used to assist decision-makers (as with human schedulers in the Whitestein and Magenta applications), trusting the results of an optimisation or a simulation is essential, especially since decision-makers have a great deal of experience in their area. A period of familiarisation and day-to-day interaction with the system is therefore necessary for users to accept and have confidence in the recommendations of the software, and to rely on it when making critical decisions.

At this user-oriented level, usability was also an important aspect of operationalisation. To encourage day-to-day use in a seamless manner, customers may require that user interfaces are designed in a similar way to, or share some features with, standard software packages used in that particular industry. For example, the experience of Whitestein was that transportation management systems have specific interface designs and functionality, which the agent-based system for transport route optimisation had to provide.

In the Co-JACK application, demonstrations with non-expert users initially showed a fair amount of difficulty in the use of the system both in terms of writing accurate tactics for the model and tracing and understanding overall system behaviour. This was due to the complexity of inferences at the agent level, and of the interactions between agents. Thus, dealing with the combinatorial explosion of different outcomes in the battle-space, and how this explosion could most usefully be presented for decision support or training, were two of the objectives identified for further development during a subsequent project phase.

3.6 Agent Standards

While agent standards, such as FIPA[2], are developing and becoming more mature, it is interesting to note that in many of the projects outlined in this article, the developers did not use current agent standards for such elements as agent communication, and have developed their own languages and protocols. These design decisions reflect the closed nature of many of the systems. Where systems are open, in that they involve different organisations, they are not open in the sense that agents developed outside of these organisations can participate in the system. Even so, with these organisations developing their own interaction languages and protocols, this is a sign that the state-of-the-art in this area does not appeal to industry partners as much as many would hope, and that perhaps work is required in this field to make it more relevant and applicable to industry.

[2]http://www.fipa.org

4. CONCLUSIONS

This paper has reviewed and discussed some of the most important lessons and experiences of developing agent-based applications for niche markets, in the logistics, production, defence and insurance sectors. The experiences and challenges faced are largely determined by the nature of the business operations in the industry domain, the characteristics of the problem to be solved and the issues related to the organisational and technical infrastructure of the client company. Nevertheless, commonalities were observed across these applications, which led us to believe that sharing these lessons may also be useful to other companies considering agent-based implementations in the future. The common aspects can be summarised as follows.

- From a business perspective, technology adoption decisions should be strategic, based on long term benefits and supported by a clear business evaluation. A software developer wishing to sell agent technology to a new company must put forward a strong business case, identifying the commercial and technical value of an agent solution for that company, and the advantages of agent technology over other technologies already in use in the relevant industry, but without underestimating them.

- From a technical perspective, while the specific agent solutions considered here address specific performance requirements like dynamic adaptivity and real-time response, they also had to satisfy requirements common to any software implementation for industrial applications, such as accuracy of results, interoperability, robustness and user-friendliness. It is thus important that development and deployment is approached with traditional software engineering methods and standards, and traditional project and risk management techniques used for IT implementations.

More generally, it can be concluded that successful adoption of agent technology depends on identifying the specific key conceptual and technical aspects related to agents that can address real world problems, even if these problems require unsophisticated solutions, and introducing innovation only where necessary.

Acknowledgements

This paper is a personal view based on interviews conducted with relevant experts from each of the companies mentioned: Tony Delaney and Martyn Fletcher from Agent Oriented Software; Vince Darley from Eurobios; Nadezhda Yakounina of Magenta Technology; Christian Danneger and Klaus Dorer from Whitestein Technologies; Bryan Woody and David Davis from NuTech; Chris van Aart from Acklin and Y'All. The interviews were undertaken as part of the case study effort of AgentLink III, the European Commission's Coordination Action on Agent-Based Computing. We are grateful to all those who contributed. The full set of case studies is available from www.agentlink.org where more complete presentations are also given.

5. REFERENCES

[1] M. Buchanan. Special report: Supermodels to the rescue. *strategy+business*, Spring, 2005.

[2] V. Darley and D. Sanders. An agent-based model of a corrugated-box factory: the trade-off between finished goods stock and on-time-in-full delivery. In H. Coelho and B. Espinasse, editors, *Proceedings of the Fifth Workshop on Agent-Based Simulation*, 2004.

[3] K. Dorer and M. Calisti. An adaptive solution to dynamic transport optimiation. In M. Pechoucek, D. Steiner, and S. Thompson, editors, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems: Industry Track*, pages 45–51. ACM Press, 2005.

[4] J. Himoff, P. Skobelev, and M. Wooldridge. MAGENTA technology: Multi-agent systems for industrial logistics. In M. Pechoucek, D. Steiner, and S. Thompson, editors, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems: Industry Track*, pages 60–66. ACM Press, 2005.

[5] M. Luck, P. McBurney, O. Shehory, and S. Willmott, editors. *Agent Technology: Computing as Interaction. A Roadmap for Agent Based Computing*. University of Southampton on behalf of AgentLink III, 2005.

[6] F. Maturana, P. Šlechta, P. Vrba, P. Tichý, R. Staron, D. Carnahan, F. Discenzo, V. Mařk, and K. Hall. A specification to build distributed reconfigurable manufacturing systems using intelligent agents. In *The 3rd International Conference on Reconfigurable Manufacturing (CIRP 2005)*, Ann Arbor, MI, 2005.

[7] F. P. Maturana, R. J. Staron, and K. H. Hall. Methodologies and tools for intelligent agents in distributed control. *IEEE Intelligent Systems*, 20(1):42–49, 2005.

[8] G. Moore. *Crossing the Chasm*. HarperCollins, 1991.

[9] P. Tichý, P. Šlechta, F. P. Maturana, R. J. Staron, K. Hall, V. Mařk, and F. M. Discenzo. Multi-agent technology for robust control of shipboard chilled water system. In *Proceedings of International Federation of Automatic Control (IFAC) Conference on Control Applications in Marine Systems (CAMS 2004)*, Ancona, Italy, 2004.

[10] C. J. van Aart, K. van Marcke, R. F. Pels, and J. L. F. C. Smulders. International insurance traffic with software agents. In F. van Harmelen, editor, *Proceedings of the Fifteenth European Conference on Artificial Intelligence*. IOS Press, 2002.

[11] T. Wagner, L. Gasser, and M. Luck. Impact for agents. In M. Pechoucek, D. Steiner, and S. Thompson, editors, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems: Industry Track*, pages 93–99. ACM Press, 2005.

A Computer-Supported Cooperative Learning System with Multiagent Intelligence

Leen-Kiat Soh, Nobel Khandaker, Xuliu Liu, and Hong Jiang

Computer Science and Engineering
University of Nebraska-Lincoln
256 Avery Hall, Lincoln, NE 66588-0115

1-402-472-6738

{lksoh, knobel, xuliu, jiang}@cse.unl.edu

ABSTRACT

In this paper, we describe an innovative infrastructure to support student participation and collaboration and help the instructor manage large or distance classrooms using multiagent system intelligence. The system, called I-MINDS, has a host of intelligent agents for each classroom: a teacher agent ranks and categorizes real-time questions from the students and collects statistics on student participation, a number of group agents that each maintains a collaborative group and facilitate student discussions, and a student agent for each student that profiles a student and finds compatible students to form the student's "buddy group". Each agent is capable of machine learning, thus improving its performance and services over time. These agents also interact and collaborate among themselves to exchange information and form coalitions dynamically to better serve the users. We have pilot-tested I-MINDS in GIS lectures, deployed I-MINDS in an introductory computer science course (CS1)'s laboratory, and evaluated the impact of I-MINDS based on student assessment. The results showed that students using I-MINDS performed (and outperformed in some aspects) as well as students in traditional settings.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *multiagent systems, intelligent agents.*

General Terms

Algorithms, Design, Experimentation.

Keywords

Multiagent, Cooperative Learning, Distance Education.

1. INTRODUCTION

Students in large or distance classrooms often do not enjoy the same level and quality of student-to-student and student-to-instructor interactions during the class (synchronously) or after the class (asynchronously) as in smaller classrooms. Such interactions are, however, important to foster group-based learning and student affinity in a class. To improve such interactions, educators have turned to technology such as personal response systems, online chat rooms, teleconferencing tools, and computer-supported cooperative learning (CSCL) environments. However, these systems do not adapt to diverse student and instructor needs and behaviors. These systems are passive participants in the process and students and instructors often resort to other modes of interactions (e-mail, phone, or face-to-face meetings) in order to address individual problems. Further, the advent of Internet and multimedia technology has meant potentially drastic changes in the teaching and learning process from the traditional classroom setting to a more geographically distributed, virtual but still interactive one. Hence, there are both motivation and opportunity to develop intelligent CSCL systems.

The long-range goal of our research is to improve teaching and learning through technology, where intelligent agents actively support *cooperative learning* among students and support teachers in managing real-time classroom activities. A secondary goal is to bring the same quality of interaction and team building that invigorates the traditional classroom to distance education for both synchronous and asynchronous student learning. Note that "cooperative learning is instruction that involves students working in teams to accomplish a common goal" [4].

Our research has designed and developed an infrastructure called the Intelligent Multiagent Infrastructure for Distributed Systems in Education (or I-MINDS). I-MINDS consists of different intelligent agents working together to actively support student-student and student-instructor interactions. Specifically, there are agents that rank and categorize questions, and profile students to help the instructor manage a classroom in real-time; agents that find compatible agents to form "buddy groups" for students; and agents that form and maintain structured cooperative learning groups.

We have pilot tested I-MINDS and further deployed the software in the laboratory of an introductory computer science course at the Computer Science and Engineering Department of the University of Nebraska. Tests based on student assessments and control-treatment groups showed that I-MINDS, though still needing better graphical user interfaces and further development, could support and improve student performance and instruction in several pedagogical aspects.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005…$5.00.

In the following, we first present I-MINDS' architecture in Section 2. Section 3 discusses I-MINDS' agents as well as their roles and relationships. We then describe the suite of features that allow the agents to actively support classroom interactions in Section 4. Section 5 gives an overview of the implementation of I-MINDS. In Section 6, we discuss the results of the pilot test and the deployment. Finally, we conclude with some future directions.

2. ARCHITECTURE

I-MINDS is built using a loosely-coupled layered architecture [7] (Figure 1), which makes I-MINDS flexible and usable in heterogeneous environments.

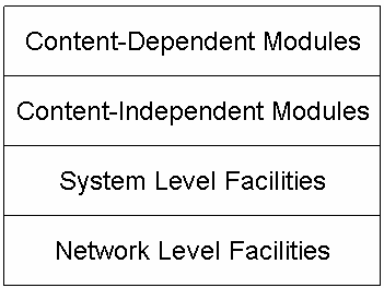


Figure 1: I-MINDS Agents and Topology

The bottom layer (i.e., the network layer) in I-MINDS provides the basic communication functionalities by using sockets. Since sockets are available in a variety of platforms across the network, this layer allows I-MINDS to communicate in heterogeneous situations.

The second layer provides system-level protocols and encapsulations with necessary abstraction to provide convenient communication and deployment functions to the upper layers. This level includes a Relational Database Management System (RDBMS)-type database (MySQL) and an audio/video server (Macromedia Flash© Communication Server). The database is used by the agents for fast storage and retrieval of information and the audio/video server helps establish audible and visual communication between the teacher and the students.

Finally the topmost two layers of I-MINDS contain the intelligent agents i.e., teacher agents, student agents, and group agents. Each intelligent agent has two sets of modules: content dependent and content independent. The content independent set provides the definitions and processes for general education-related services, while the content–dependent module handles specific course-related information and knowledge base, providing the required data and the heuristics used to gather, analyze, disseminate and process the generated data.

3. AGENTS and TOPOLOGY

Figure 2 shows an example of the topological infrastructure of I-MINDS [8]. The manager manages the ongoing classroom sessions. It manages system level information such as the list of ongoing classes, list of courses and teachers, the login names and passwords for the students for each of the classrooms, etc. I-MINDS is capable of holding multiple concurrent classroom sessions. In each I-MINDS classroom session, the teacher, students and student groups are assigned the teacher agent, student agents and group agents, respectively. These agents support the person/group that they serve.

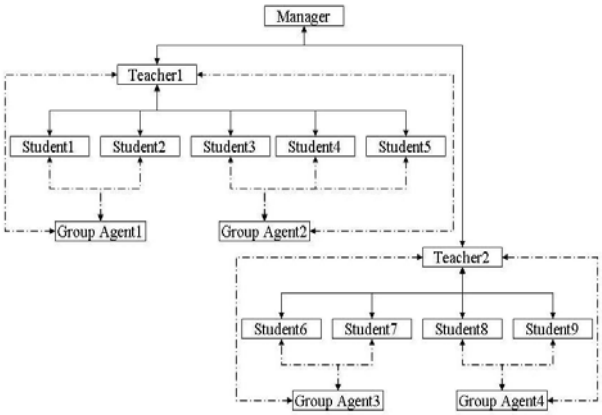


Figure 2: I-MINDS Agents and Topology

Intelligent agents are autonomous and can operate robustly in rapidly changing, unpredictable, or open environments. With these intelligent agents serving and catering to students' unique needs and behaviors, students will be able to participate in a computer-supported cooperative learning (CSCL) environment actively rather than listening to the lectures passively as in a traditional large or distance classroom. Currently, I-MINDS has three types of intelligent agents: (1) teacher agents, (2) student agents, and (3) group agents. A teacher agent, interacting with a teacher, is responsible for disseminating information streams to student agents, maintaining profiles for all students, assessing the progress and participation of different students, ranking and filtering of the questions asked by the students, and managing the progress of a classroom session. A student agent, on the other hand, mainly works as a personal helper to the student. The student agent manages the communication channels among students and between the teacher and the students. The student agent also presents the learning material to the student and forms coalitions with the other students for collaborative learning. The group agent forms and conducts structured cooperative learning such as the Jigsaw model [1], monitors and facilitates group activities.

3.1 Teacher Agent

In I-MINDS, a teacher agent has a graphical user interface (GUI) front-end, two sets of modules, and a database, as shown in Figure 3. A teacher agent interacts mainly with an instructor. It also sends instructions to the student agents, maintains student profiles, and receives questions from the student agents.

The GUI is equipped with various tools to create a classroom environment. For example, an instructor can capture and transmit desktop screenshots/slides to the class in real-time. An instructor can also communicate through the GUI with the students using text or audio/video using the message passing interface and video channel respectively [14].

The content-dependent module set of the teacher agent has a list of weighted keywords for a particular subject topic, rules and heuristics for ranking student questions for a particular course, and student profiles for a particular classroom. The content-independent module set of the teacher agent has a repository

mechanism to allow the storage and access of the knowledge bases used in the content-dependent module set. It also has reinforcement learning mechanism, associated with question processing,, to allow an instructor to implicitly "teach" the teacher agent how to rank questions better, as will be discussed in Section 4. There is also a Jigsaw support module that supports the structured cooperative learning Jigsaw model [1, 2], to be discussed as well in Section 4.

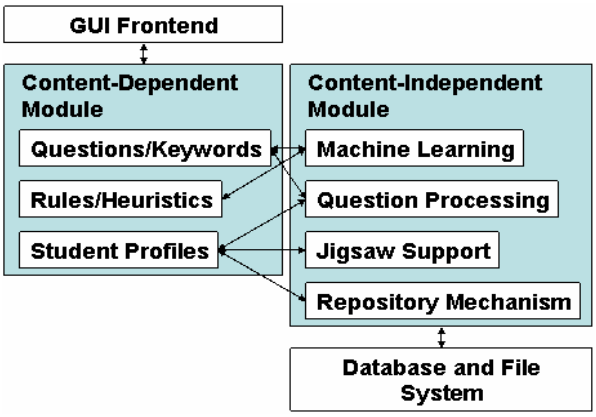


Figure 3: Teacher Agent

3.2 Student Agent

In an I-MINDS supported classroom, a student agent serves a unique student, and, similar to a teacher agent, it also has a GUI front-end, two module sets and a database as shown in Figure 4. A student agent primarily interacts with a student. It also exchanges information with the teacher agent and the group agents.

The student agent first allows for student login and course registration through communicating with the manager and the teacher agent, respectively. The student agent then acts as a medium: displaying the instructional materials received from the teacher agent to the student and relaying information from the student to the teacher agent.

In the content-dependent module, there are locally archived course materials and questions asked by the student and received from collaborative buddies [16]. Each student agent keeps a copy of profiles of all other students in the same class and synchronizes these profiles in real time with the teacher agent. The profiles are used as an important criterion by the student agent to automatically construct and refine the buddy groups. A student's buddy group is one that the student relies on to collaborate in forums and "idea sketching". This capability [15] can be turned on/off by the student.

When a student asks a question or performs some kind of collaborative activities (e.g., initiates a forum discussion or participates in sketching out ideas), his or her profile is changed and subsequently updated with the teacher agent and other student agents. The change might trigger other student agents to recruit the student to join their buddy groups. Thus, a buddy group is dynamic. Further, if a student is not responsive to a buddy group that he or she is a member of, he or she can be dropped from the group. Hence, more responsive students will be approached more frequently than the others. The student agent also adjusts its heuristic rules according to the current classroom environment. For example, in a course in which students are very active in collaborative learning in every classroom session, a buddy who has not been active for the entire session may be dropped.

Error! Objects cannot be created from editing field codes.

Figure 4: Student Agent

In the content-independent module, the tracking component tracks all activities and progresses of the student's learning and collaboration. For example, I-MINDS provides a digital forum and a digital white board for students to communicate and collaborate. The digital white board facilitates "idea sketching". All messages posted in the forum and activities on the digital white board are tracked and analyzed by the student agent. The tracking component also tracks the student's attendance. If the student misses part of a class session, the student agent contacts the teacher agent, finds the archived course materials for the missed time period according to the timestamps and the syllabus, and reminds the student about the missed lectures. All the tracking results are sent to the teacher agent and factored into the student's profile.

3.3 Group Agent

In I-MINDS, a group agent is activated when there are structured cooperative learning activities. Structured cooperative learning involves specified activities that explicitly require students to cooperate. Currently, I-MINDS implements the Jigsaw model [1], as will be discussed in Section 4. Since a group agent works behind the scene, it does not have a GUI front-end. Like the teacher and student agents, it has two content module sets and a database, as shown in Figure 5.

To support structured cooperative learning, the content-independent module set is capable of forming groups (or coalitions) of students based on their profiles, evaluating student perception of teamwork and other members through online surveys, tracking student participation, and improving its heuristics in monitoring and forming better groups. More will be discussed about the coalition formation module in Section 4.

The content-dependent module performs similar tasks as those discussed in previous sub-sections, archiving group activities (tasks and subtasks), and storing specific rules, heuristics and student profiles.

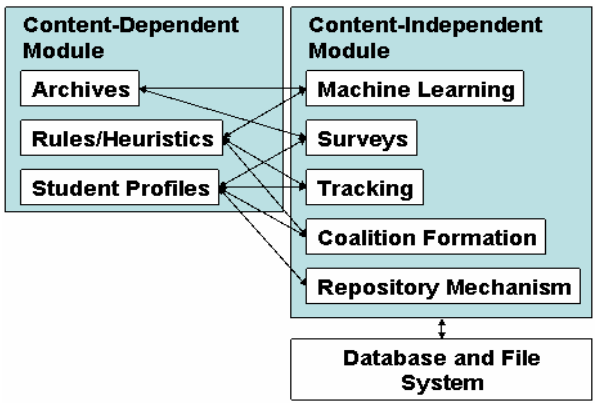


Figure 5: Group Agent

4. Intelligent and Multiagent Modules

In this section, we will discuss several key modules that makes our I-MINDS intelligent: automated question ranking that scores each question in real-time and ranks them, question grouping that classifies questions into similar groups based on their utterance classification and keywords, student profiling that keeps tracks of how each student performs in the classroom, cooperative student learning support implementing the Jigsaw process, and coalition formation that builds and refines the "buddy group" of each student.

4.1 Question Ranking

Question ranking is a key feature of the Question Processing module shown in Figure 2 that allows the teacher agent to manage large or distance classrooms for the instructor. When questions are asked, the teacher agent scores each question based on a set of keywords and heuristics. Keywords are subject-topic specific and weighted. Heuristics are course-specific. For example, a question asked by a student with a high-score profile will be given a high score; a question asked by a student who has never asked a question before will also be given a high score to encourage the student to ask more questions in the future; and a "why?" question is scored higher than a "what?" question. After scoring, the questions are ranked and displayed to the instructor. The instructor may choose to answer or discard a question. This is how the instructor's actions trigger the reinforcement learning mechanism of the module. When the instructor chooses a lowly-ranked question over a highly-ranked question to answer, the teacher agent takes this as an indication that it has ranked the question incorrectly. It thus traces back to the heuristics and keywords that have contributed to the high score of the question and lower their weights; likewise, it increases the weights of the heuristics and keywords that have scored the question low in the first place. Thus, by answering and discarding ranked questions, the instructor implicitly teaches the teacher agent how to better score and rank the questions. Further, the Question Ranking feature also parses and extracts keywords from the questions. Questions that are answered trigger the learning of these keywords. As a result, the teacher agent is able to add new keywords to the list of keywords, effectively building and refining important keywords for specific courses and subject topics. Details of this module can be found in [9].

4.2 Question Grouping

Question grouping is a key feature of the Question Processing module shown in Figure 2 that allows the teacher agent to address multiple similar questions together and profile students who ask similar questions. Future applications of this feature include automatically generated and answered Frequently Asked Questions (FAQs).

Question grouping is based on question classification and keyword matching. To achieve question classification, we adopt the *utterance classification* approach in the AutoTutor [3]. We use the ApplePie parser [11] and the utterance classifier program of AutoTutor [10] to classify a question into one of twelve classes such as Contribution, Discovery, and so on. This classification feature also allows the group agent to monitor the roles of group members in group discussions. Two questions are also considered similar if they share a high percentage of keywords.

4.3 Student Profiling

A student is profiled by I-MINDS' agents in several ways. First, student-instructor interactions are profiled mainly based on the quality of questions that a student asks. Based on the grouping of these questions, students can also be profiled relative to others in terms of compatibility. Second, student-student interactions are profiled through the monitoring of a buddy group. This profiling process is carried out by a student agent from the viewpoint of the student it serves. That is, the student agent keeps a profile of its student's buddies in terms of their responsiveness in collaborative activities (forums and digital whiteboard). These include the number of messages sent, type of messages, frequency of messages being sent, time spent on "idea sketching", the number of times a student is dropped from a buddy group, and so on. Third, group interactions are monitored by the group agents during structured cooperative learning. These include the number of messages sent, types of messages, self-reported teamwork capabilities, peer-based evaluations as a team member, and so on. The teacher agent collects the student-student and group profiles from the student and group agents. The group agent does make use of the student-student profiles to help form coalitions.

4.4 Coalition Formation Module

To support the Jigsaw learning model we have designed an auction-based learning-enabled coalition formation algorithm called VALCAM that combines automatically tracked collaboration activities with subjective peer-based evaluations of a student's performance in structured cooperative learning.

In VALCAM, the teacher agent acts as a coordinator or the coalition formation process and makes global decisions, such as, what should the least number of members in a coalition be, how long should the coalitions last, how the performance of each coalition should be evaluated, etc. The group agents then manage the coalitions. Each group agent monitors the performance and activities of the members of its assigned group. After the coalition has completed their tasks, the group agent also evaluates the performance of each student agent as a group member.

VALCAM algorithm is based on reinforcement learning of the student agents. In the beginning of every session the teacher agent chooses a few students and then initiates the group formation. During the group formation process, each student agent bids—an iterative Vickrey auction—to join its favorite group based on

virtual currency earned from previous Jigsaw activities. Once the groups are formed, the student agents work with their group members to complete tasks assigned by the instructor. Finally, when the assigned task is completed, the student agents are rewarded with virtual currency based on their individual performance and their performance as a group member through surveys of their peers' evaluation of the student. Students who performed well will receive more virtual currency, allowing them to more successfully bid for their favorite groups in future sessions. Details of this algorithm can be found in [5].

5. REALIZATION of I-MINDS

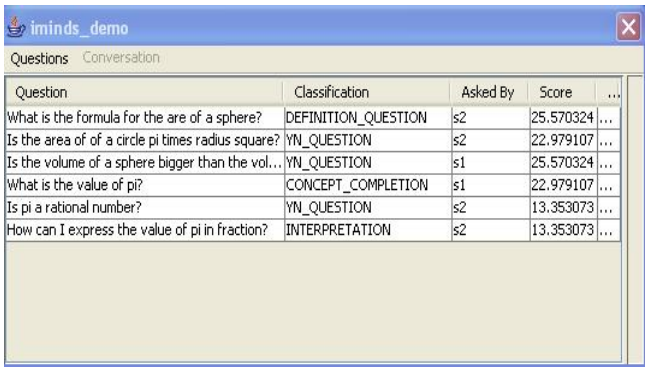
We have implemented I-MINDS in Java (SDK 1.4.2). We have used Java's socket functionalities to establish communication among agents, Java's swing class to create interfaces. Furthermore, we have used Java's JDBC technologies to connect to our MySQL database repositories to store and retrieve all data. We have also used the Macromedia Flash© Communication Server to establish audio/visual link between the teacher and the students. Figures 6 and 7 show the interfaces of the teacher and student agents.

Figure 6(a) shows the teacher agent interface. The basic buttons allow the teacher to send a page (basically a snapshot of what is on the desktop of his or her computer), modify a page with full editing capabilities (draw, write, etc.), create a blank page and build one from scratch, to bring up statistics on the students ("stuMng"), to log out, and to bring up or close the Q&A interface (QuestionList). It also has two menu groups. The "Jigsaw" menu group allows the instructor to oversee the Jigsaw process, such as creating a task and its subtasks, specifying the times allotted for each Jigsaw phase, and making sure that all phases are started and terminated correctly. The "Jigsaw Group" menu group allows the instructor to oversee each group's activities and monitor how group members are interacting (via the forums and digital whiteboards).

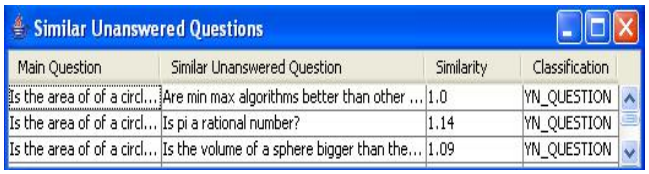
Figure 6(b) shows the popped up Q&A interface for the teacher agent. Each question is displayed with its type classification, the student who asked the question, and a score. The instructor then could sort the questions by clicking at a specific column heading. In Figure 6(b), the questions are sorted by their "scores", where the scores are computed using the design as discussed in Section 4.1. The instructor may also click on each question to bring up a detailed information window that displays the entire question and the profile of the student who asked the question. The instructor may also click on each question to "discard" or "answer" it, effectively "teaching" the teacher agent to learn about the inappropriate or appropriate questions. The Q&A interface also has a menu group "Questions" that allows the instructor to bring up "similar unanswered questions" of a question. Figure 6(c) shows the graphical user interface of this component. It lists the main question, the similar unanswered questions, the similarity scores, and the type classification. This is done through the question grouping design previously discussed in Section 4.2.



(a)



(b)

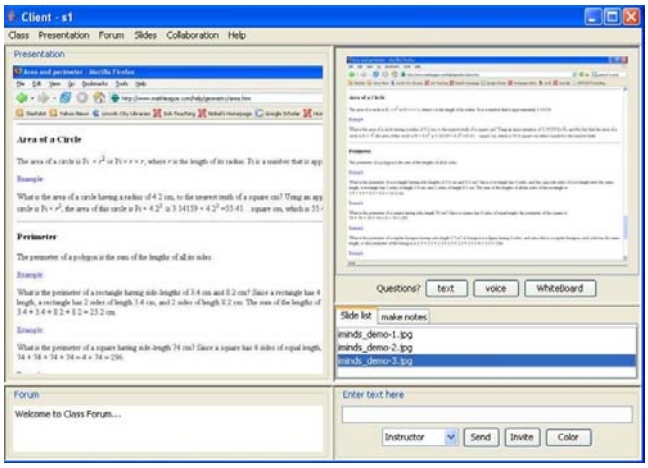


(c)

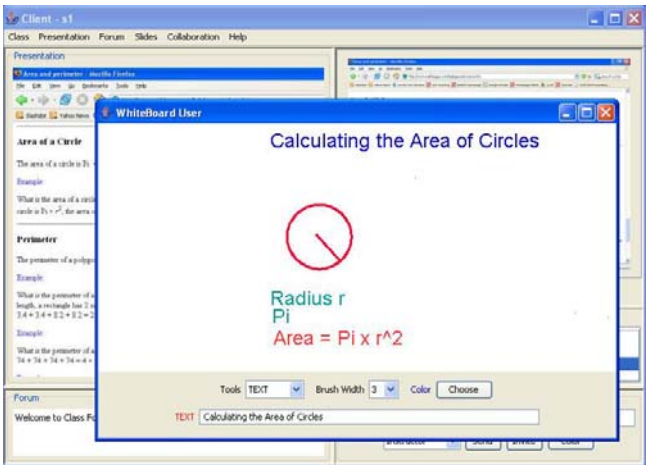
Figure 6: Teacher agent interface (a) the teacher agent interface, (b) the teacher agent's popped up Q&A interface, and (c) the teacher agent's popped up Q&A interface component that deals with question similarity

Figure 7(a) shows the full-feature student agent interface. It has four quadrants. The size of each quadrant is adjustable by moving the borders. The top-left quadrant is the "presentation quad" that displays any document (e.g., a presentation slide, a web page, or a text file, an image, or even a snapshot of the computer desktop) that the instructor sends to the students. The top-right quadrant is the "housekeeping quad" that allows a student to bring up a particular archived document, ask a textual or audio question to the instructor, and invoke a digital whiteboard. It also has a list of archived presentation pages. The bottom-right quadrant is the "text entry quad" that allows a student to type and send messages to various groups of recipients. It also allows a student to invite another student agent to be in the "buddy group" and change the color of the student's messages. Finally, the bottom-left quadrant is the "forum quad" that displays all the messages received by the student and the student agent, in color-coded texts.

Figure 7(b) shows the student agent interface with the invoked digital whiteboard. It allows students to draw and write on the board simultaneously through their student agents. Further, it has a "control token" that a student may grab such that he or she can draw and write on the board while the others cannot. This allows a student to be an active contributor while others in the buddy group "watch" or "listen" to what the student is going to contribute. The student may then release the token to allow for group contribution to resume.



(a)



(b)

Figure 7: Student agent interface: (a) the "quad"-based design, full-feature, (b) the invoked digital whiteboard on top of the quad design

6. Results

6.1 Pilot Study

To determine the potential impact of I-MINDS on student learning, a pilot study [12] was conducted in May 2003 where the tool was used by subjects in a controlled experiment to assess what impact it had on student learning of Global Information Systems (GIS) content. GIS technology can be used for scientific investigations, resource management, and development planning.

In this study, all sessions were taught by the same instructor. There were 20 undergraduate and graduate student participants, divided into two groups. The *control* group was conducted like a traditional classroom, with the instructor and the students in the same room. The *treatment* group was conducted as a distance classroom where the instructor and the students were in separate rooms and their only communication channel was through I-MINDS.

All students took a pretest that would serve as a baseline score. Then, each group was given two 1-hour sessions. Session 1 discussed basic GIS topics and session 2 discussed advanced GIS topics. After each session, each student was required to take a post-test. The instructional content was the same for both groups.

Results from the two testing sessions were encouraging. Here we outline the results as reported previously in [12].

Although there was a slight difference in the means of the experiment and control groups for Test 2, this difference was not statistically significant ($p > .05$), and *the amount that the treatment group improved from the pretest to the posttest was nearly twice that of the control group.*

Comments from the university professor who used I-MINDS in teaching both of the content lessons were also encouraging. He indicated that the teaching tool was very easy to learn and use. He also said that the tool could enhance distance learning, especially by making it possible for building an archive of information that could be accessed "on-demand" by students. The instructor also noted that questions asked of him via I-MINDS tended to have higher quality, reflecting a deeper understanding, and demand a richer response than those questions posed during the control sessions.

Details of this pilot study can be found in [12].

6.2 Deployment with Jigsaw

In Spring 2005, we deployed I-MINDS in the laboratory of an introductory computer science course at the University of Nebraska. The course is CS1, the first core course for computer science and computer engineering majors in the Computer Science and Engineering Department. The course had three 1-hour weekly lectures and one 2-hour weekly laboratory sessions. In each lab session, students were given specific lab activities to experiment with Java and practice hands-on to solve CS problems. In Spring 2005, there were three different lab sections, each with about 20 students.

Our study utilized two sections. In the control section, students worked in Jigsaw cooperative learning groups without using I-MINDS. Students were allowed to move around in the room to join their Jigsaw groups to carry out face-to-face discussions. In the treatment section, students worked in Jigsaw cooperative learning groups using I-MINDS. Students were told to stay at their computers and were only allowed to communicate via I-MINDS. With this setup, we essentially simulated a distance classroom environment.

For each lab, the students were given a lab handout with a list of activities—thus, a lab is a task and its activities are the subtasks. We conducted the study for three weeks, covering topics in debugging and testing, Unified Modeling Language (UML), and recursion.

The students of both the control group and treatment group were required to complete the tasks and subtasks in the four Jigsaw Phases. Table 1 shows the four Jigsaw phases with the allocated amount of time.

In each section, the instructor announced the main groups. In the control section, this was done manually. In the treatment section, I-MINDS automatically performed this using its VALCAM algorithm (Section 4.5). Once the main groups were formed, the teacher agent formed the focus groups by randomly selecting students from the main group. After the focus groups were formed, every focus group was assigned one subtask randomly.

After the subtask assignment the focused exploration phase was started. In this phase the three Jigsaw Phases were carried out in order. During these three phases the student agents and the group agents monitored and controlled the activities of the students and the student groups respectively. After the three Jigsaw Phases were executed, all the students filled out the Peer Rating Questionnaire and Team Based Efficacy Questionnaire and took a 10-minute post-test. This 10-minute post-test score was used as the measure of student performance in terms of understanding the topic of the lab. Details of this process can be found in [13].

Table 1: Time allocations of Jigsaw phases

Jigsaw Phase	Time (min)
Introduction	15
Focused Exploration	30
Reporting and Reshaping	20
Integration and Evaluation	20

Table 2 shows the post-test scores of the three sessions for the "Control" (i.e., Jigsaw without I-MINDS) and "Treatment" (i.e., Jigsaw with I-MINDS) sections. The results indicate that students using I-MINDS for the Jigsaw activities were able to obtain comparable and even better post-test scores. We had initially hypothesized that the students in the control section would perform better than the students in the treatment section simply because the students in the former would have a chance to discuss face-to-face and were able to use paper and pencil to draw and share the same computer screen during the stages 2 and 3 of the Jigsaw process. On the other hand, I-MINDS, still lacking user-friendly GUI, had been expected to hinder such free, natural interactions among students, leading to ineffective collaborations. Thus, the result had not been expected. Students in the Treatment section performed better than the students in the Control section in the later two sessions. Further, students in the Treatment section also achieved better standard deviation—meaning that these students' post-test scores were more tightly clustered than those of the Control section. Upon closer analysis, we suspect that the act of typing and communicating through the forum and digital whiteboard of I-MINDS forced the students to articulate explicitly their thoughts and focused their attention to the tasks at hand. This in turn improved student understanding of the subject matter.

Table 2: Student performance for the Control (Jigsaw only) and Treatment (Jigsaw w. I-MINDS) sections

Peer Rating Questionnaire Results				
Week	Control Section		Treatment Section	
	Mean	Std. Dev	Mean	Std. Dev
1	42.02	2.68	35.60	5.97
2	39.06	7.05	33.30	8.65
3	39.90	4.80	38.59	6.34

The results were very encouraging: without face-to-face interactions, students carried out their I-MINDS-supported Jigsaw tasks and performed relatively well in individual post-tests.

Next, we look at the surveys in terms of peer rating and team-based efficacy. These surveys were conducted in both sections after each lab session. Tables 3 and 4 show the results. The *peer rating* average is the average peer rating scores that each student gave to his or her group members. As evidenced in the scores, students in the Control section rated their peers better than the students in the Treatment section. This is possibly due to the face-to-face interaction. After all, students interacting through I-MINDS could not enjoy the advantages of face-to-face interactions such as facial expressions, the spontaneous free-flowing of ideas, and more immediate feedback in their discussions. This observation indicates that I-MINDS still lacks sufficient graphical user interface and multimedia capabilities to fully capture real-time characteristics.

The team-based efficacy was collected after each lab based on a set of questions. It measured how a student viewed how well its group had performed. Once again, students in the Control section approved of their team-based activities more than the students in the Treatment section. Since there was no difference in the scores in tasks for the Control and Treatment sections, and that students in the Treatment section had more similar post-test scores, that means this perception had been likely biased by the ease-of-use (or the lack thereof) of I-MINDS.

Table 3: Peer Rating Questionnaire Results

Peer Rating Questionnaire Results				
Week	Control Section		Treatment Section	
	Mean	Std. Dev	Mean	Std. Dev
1	42.02	2.68	32.45	5.78
2	39.06	7.05	37.72	4.60
3	39.90	4.80	34.63	8.08

Table 4: Team Based Efficacy Questionnaire Results

Team Based Efficacy Questionnaire Results				
Week	Control Section		Treatment Section	
	Mean	Std. Dev	Mean	Std. Dev
1	29.87	2.51	27.72	5.08
2	31.86	3.48	29.20	2.78
3	30.08	3.02	28.25	4.02

Looking at the data tracked by the I-MINDS agents and comparing them to post-test scores, we observe the following. First, students rate their peers subjectively: average peer rating that a student gave to his or her peers is negatively correlated to his or her post test score with a correlation of -0.63. Second, students who do well in the post-tests tend to send long messages with a correlation of 0.51.

7. CONCLUSION

We have described a computer-supported cooperative learning system called I-MINDS that uses multiagent intelligence. We have discussed the structure and functionalities of the different agents in I-MINDS. Then we have described the intelligent and multiagent modules that allow the teacher, group, and student agents to collaborate to actively support student-student and student-instructor interactions. To demonstrate the effectiveness of I-MINDS, we have presented the results of our pilot study and also the results of incorporating Jigsaw Framework in I-MINDS in real-time classroom situations.

The results of the pilot study shows that the instructor and the students can both use I-MINDS effectively as a framework for delivering distance education.

Furthermore, the results of our CS1 lab deployment shows, although face-to-face activities were perceived to be better by the students, I-MINDS-supported activities were able to produce better or comparable post-test scores and were able to yield more similar scores among the students.

Future work includes continued deployment of I-MINDS in the classroom, improvement of the GUI front-end for the teacher and student agents to better support collaborative, real-time programming, and extension of I-MINDS to work more seamlessly with existing Interactive Design Environments (IDEs). Additional research includes the refinement of the VALCAM algorithm to more fairly synthesize subjective peer-based evaluation and agent-tracked empirical data in determining the amount of virtual currency rewarded to each student.

8. ACKNOWLEDGMENTS

The I-MINDS project was first developed with a seed grant from the National Center for Information Technology in Education (NCITE). It was later sponsored partially through an NSF SBIR grant DMI-044129. The authors would like to thank Phanivas Vemuri, Suresh Namala, and Xuesong Zhang for their programming work.

9. REFERENCES

[1] Aronson, E., Blaney, N., Sikes, J., Stephan C., and Snapp, M. *The Jigsaw Classroom*. Beverly Hills, CA: Sage, 1978.

[2] Clarke, J. *Pieces of the Puzzle: The Jigsaw Method*. In *Handbook of Cooperative Learning Methods*. Westport, CT: Greenwood Press, 1994, 34-50.

[3] Graesser, A. C., Wiemer-Hastings, K. , Wiemer-Hastings, P., Kreuz, R. and Tutoring Research Group. Autotutor: A simulation of a human tutor. In *J. Cognitive Systems Research*, 1999, vol. 1, 35–51.

[4] Johnson, D. W., Johnson, R. T. and Smith, K. A. Cooperative learning: increasing college faculty instructional productivity. In *ASHE-ERIC Higher Education Report No.* 4, George Washington University, 1991.

[5] Khandaker, N. VALCAM - an auction based learning enabled multiagent coalition formation algorithm for real-world applications. M.S. Thesis, Univ. of Nebraska, NE, 2005.

[6] Khandaker, N. Question classification and question similarity in I-MINDS. In *Independent Research Report*, Dept. Computer Science & Engineering, Univ. Nebraska, Lincoln, NE, 2004.

[7] Liu, X., Zhang, X., Al-Jaroodi, J., Vemuri, P., Jiang, H. and Soh, L.-K. I-MINDS: An Application of Multiagent System Intelligence to On-Line Education. In *Proc. IEEE-SMC*, 2003, 4864-4871.

[8] Liu, X., Zhang, X., Soh, L.-K., Al-Jaroodi, J. and Jiang, H. A Distributed, Multiagent Infrastructure for Real-Time, Virtual Classrooms. In *Proc. ICCE*, 2003, 640-647.

[9] Namala, S. An intelligent module for I-MINDS. M.S. Project Report, CSE Dept. Univ. of Nebraska, NE, 2004.

[10] Olney, A., Louwerse, M. Mathews, E., Marineau, J., Hite-Mitchell, H. and Graesser, A. Utterance classification in Autotutor. In *Proc. HLT-NAACL Workshop*, 2003, 1-8.

[11] Sekine, S. and Grisman, R. A corpus-based probabilistic grammar with only two non-terminals. In *Proc. IWPT*, 1995, 216-223.

[12] Soh, L.-K., Jiang, H. and Ansorge, C. Agent-Based Cooperative Learning: A Proof-of-Concept Experiment. In *Proc. SIGCSE*, 2004, 368-372.

[13] Soh, L-K., Khandaker, N., Liu X. and Jiang, H. Computer-Supported structured cooperative learning. To appear in *ICCE'2005*, 2005.

[14] Vemuri, P. Multimedia Support for I-MINDS. M.S. Project Report. Dept. Computer Science & Engineering, Univ. Nebraska, Lincoln, NE, 2003.

[15] Ye, Y. and Churchill, E. *Agent Suported Cooperative Work*. Boston, MA:Kluwer Academic Publishers, 2003.

[16] Zhang, X. I-MINDS: An Intelligent Multiagent System Supported Teaching and Cooperative Learning Environment. M.S. Thesis, Dept. Computer Science & Engineering, Univ. Nebraska, Lincoln, NE, 2004.

Deploying a Personalized Time Management Agent

Pauline Berry, Ken Conley, Melinda Gervasio, Bart Peintner, Tomás Uribe, Neil Yorke-Smith
{berry,conley,gervasio,peintner,uribe,nysmith}@AI.SRI.COM
Artificial Intelligence Center, SRI International, Menlo Park, CA 94025 USA

ABSTRACT

We report on our ongoing practical experience in designing, implementing, and deploying PTIME, a personalized agent for time management and meeting scheduling in an open, multi-agent environment. In developing PTIME as part of a larger assistive agent called CALO, we have faced numerous challenges, including usability, multi-agent coordination, scalable constraint reasoning, robust execution, and unobtrusive learning. Our research advances basic solutions to the fundamental problems; however, integrating PTIME into a deployed system has raised other important issues for the successful adoption of new technology. As a personal assistant, PTIME must integrate easily into a user's real environment, support her normal workflow, respect her authority and privacy, provide natural user interfaces, and handle the issues that arise with deploying such a system in an open environment.

Categories and Subject Descriptors

H.4.1 [**INFORMATION SYSTEMS APPLICATIONS**]: Office Automation—*Time management*; I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence—*Intelligent agents*; K.6.3 [**MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS**]: Software Management

General Terms

Design, Human Factors

Keywords

scheduling, personal assistants, user interfaces, deployment

1. INTRODUCTION

Over the past two-and-a-half years, we have been developing the Personalized Time Manager (PTIME), an agent with the goal of helping a human user manage her temporal commitments in a consistent, integrated manner over an extended period of time. While automated meeting scheduling has been well studied in the research community [8, 25, 23, 17], the resulting systems have rarely been adopted. We can point to two main reasons. The first is a failure to take into account the intensely personal nature of the human time management problem. Most individuals, even the busiest knowledge workers, are reluctant to relinquish control over the management of their own time. Further, people have different preferences and practices regarding how they schedule their time, how they negotiate appointments with others, and how much information they are willing to share when doing so. To be successful, a time management solution must thus address these basic issues of authority, privacy, and preferences. The second reason is that adoption of such a system is contingent on its practical usability—including important user interface issues—within a real-world deployment. While we do not claim to have fully resolved either one of these obstacles, this paper describes our experience and progress in overcoming these challenges.

Each PTIME agent is an autonomous entity that works with its user, other PTIME agents, and other users, to schedule meetings and commitments in its user's calendar. It can negotiate with other agents on behalf of its user, provide her with useful information about upcoming events while protecting her privacy, and learn her preferences. To achieve these goals, PTIME integrates commercially available calendaring tools and user interface technology with state-of-the-art algorithms for constraint satisfaction (to solve the underlying scheduling problems and find preferred schedules), goal-directed process management, and preference learning.

PTIME is part of the CALO project, a large-scale effort to build an adaptive cognitive assistant situated in the office environment [15]. A CALO agent is designed to support its user in various ways, including project and task management, information collection and organization, and meeting preparation and summarization. In October 2005, the CALO project started its third year of research and development. In Year 1, the theme was *integration*—assembling a large collection of independently developed pieces of software into a single system. CALO 1.0 was highly distributed, with the CALO modules implemented on top of the Open Agent Architecture (OAA) [4] and with integration primarily at the level of communication. CALO 1.0 required multiple computers to run and provided only the most rudimentary desktop functions. In Year 2, the theme was *usability*—integrating the applications within a common architecture that could run on a single laptop computer. CALO 2.0 was reborn within the IRIS desktop environment [5], which featured a semantic knowledge base, based on the CALO ontology, and shared by a suite of common desktop applications, including email, calendar, web, file system, and chat. Through the shared knowledge base, IRIS supported the sharing of information between the disparate applications, enabling CALO to better control and coordinate its assistive behavior. With CALO 2.0, users could read email, file documents, schedule meetings, organize bookmarks,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

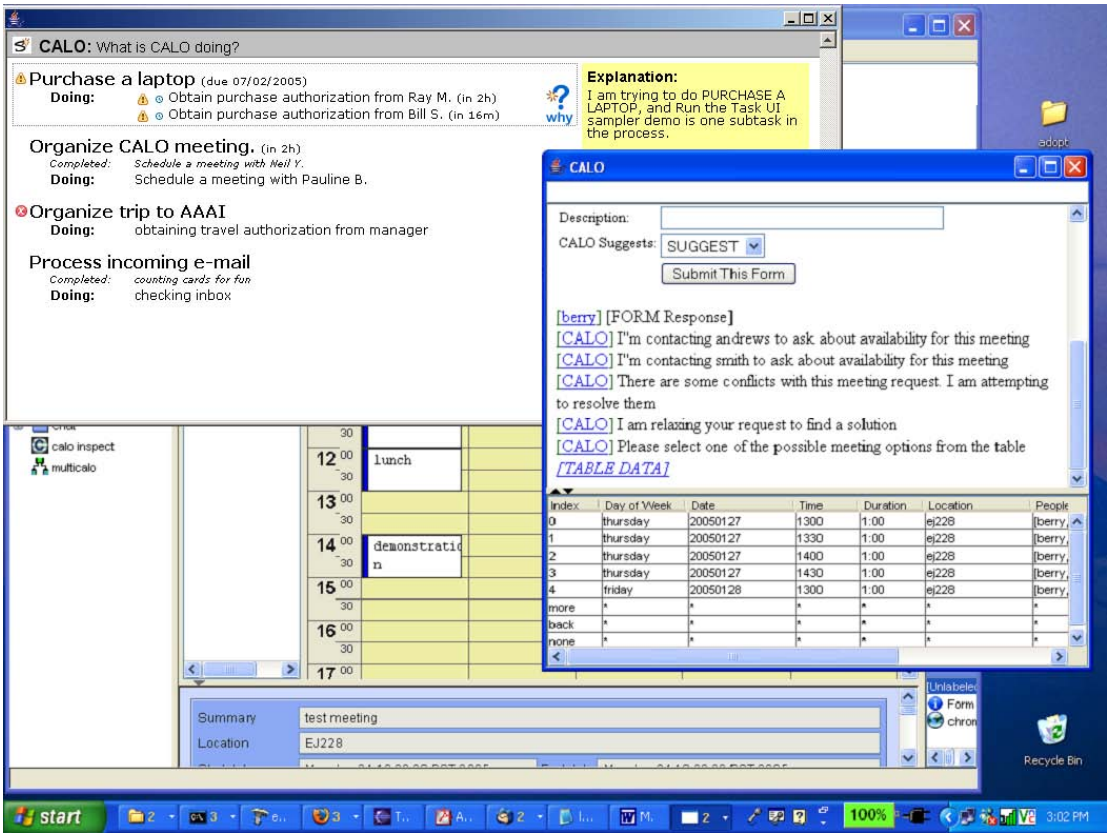


Figure 1: Screenshot of Year 2 PTIME interface

and perform various other everyday office activities. However, there remained obstacles to its widespread use, such as cumbersome user interfaces and computational inefficiency. The theme for Year 3 is *usefulness*—providing users with a CALO that is not only usable but can also truly help make busy office workers more efficient and effective in their everyday lives.

This paper describes lessons learned from the development and deployment of PTIME within the CALO project. We begin by presenting the PTIME system, its local agent architecture, and its deployment in a multi-agent setting (Section 2). Next, we present the challenges we faced and the solutions we implemented to successfully field a PTIME agent in CALO 1.0 and 2.0 (Section 3). Then we discuss other issues that arose post-deployment, through actual use by real users, and we present the modifications we made to address these issues prior to the final, annual release (Section 4). We then discuss our plans for CALO 3.0, including the modifications to existing features and implementation of new features designed to address known issues as well as new challenges (Section 5). We conclude with a discussion of related work and a summary of lessons learned (Sections 6 and 7).

2. PERSONALIZED TIME MANAGEMENT

Despite the proliferation of calendar tools to organize, display, and track commitments, most individuals still expend a considerable amount of effort managing their schedules.

The research community has primarily focused on automated meeting scheduling: finding feasible timeslots for meetings given a set of requirements on participants, times, and locations. Work in this area can be generally divided into *open* and *closed* scheduling systems [8]. In an open system, individuals are autonomous, responsible for creating and maintaining their own calendars and meeting schedules. They generally operate in an unbounded environment, without any obligation to a single organization. In a closed system, by contrast, a consistent and complete global calendar is maintained. While closed systems are more commonly studied and marketed because they provide a controlled and known setting, they are rarely adopted in a uniform way because users seldom live in a completely closed environment. However, open scheduling systems pose challenges of their own. An individual may not wish to share all of her schedule, or may choose not to participate in a meeting. Different individuals will have different scheduling preferences. Without centralized control to balance the constraints and preferences of all involved parties, how does an open system respect individual preferences without devolving into completely selfish agents that result in greatly suboptimal schedules? Further, while users may not wish to cede all control to an autonomous assistant, they may let their agent automate some processes.

CALO exists in such an open, unbounded environment, where these issues of privacy, authority, adjustable autonomy, cross-organizational scheduling, and uncertain availability of participants abound. PTIME is an autonomous agent integrated within a larger CALO agent [15]. Each CALO/PTIME agent is itself composed of a set of internal agent components linked within the OAA agent architecture; we will refer to these as CALO/PTIME components.

2.1 The Distributed Scheduling Task

The distributed scheduling task that PTIME addresses fits into the continuum between fully competitive and fully cooperative agents. PTIME agents in collaboration with their users are autono-

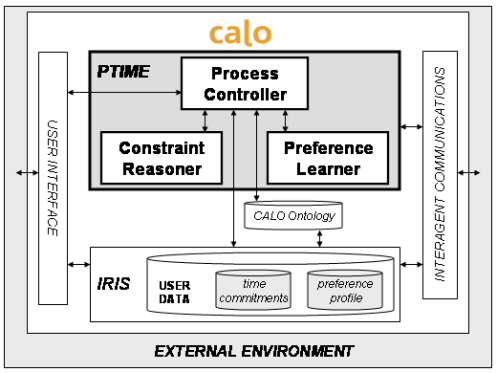


Figure 2: Architecture of Year 2 PTIME agent

mous, self-interested entities that manage their user's own calendars. They are in effect single-calendar schedulers dependent on other agents to coordinate shared calendar entities. The scheduling task is viewed as a shared goal of the user and the agent. For a PTIME agent, global utility is thus secondary to maximizing the local utility of its user, which typically does include a component valuing cooperation with others [22]. This is in contrast to most distributed scheduling systems [25], which aim to maximize some notion of global schedule quality (e.g., [16]).

Thus, PTIME supports two forms of collaboration: between itself and its user and between itself and the PTIME agents of other users. The collaborative scheduling process is separated from the scheduling algorithms, to enable interaction with the user and other PTIME agents. This interaction forms the framework for learning and adjustable autonomy.

The collaboration between PTIME and its user is illustrated in Figure 1, which is a screenshot of the PTIME interface for the deployed CALO 2.0 system. The collaboration is seen in the "CALO Chat" window in the foreground (right). In the top-left window, CALO is reporting on its activities in response to a user request for explanation. In the background, the calendar view of the IRIS desktop environment can be partially seen. In addition to a traditional form-based input mechanism, CALO has a restricted natural language (NL) input that allows the user to interact by means of spoken or typed input, for instance "meet with Alice about integration next week". CALO also has multi-modal output, including leaving spoken messages on the user's cellphone.

2.2 Local Agent Architecture

Figure 2 depicts the architecture of PTIME in relation to the relevant components in IRIS and CALO. We limit our discussion to the three PTIME components this paper primarily involves; for a more complete discussion of the various components of PTIME, see [3].

- At the heart of PTIME is its *Process Controller*, a SPARK agent that captures possible interactions between the user and other agents in the form of structured decision points. SPARK [19] is a Belief-Desire-Intention agent framework that supports both goal-directed and reactive behavior. The Process Controller manages PTIME's processes, tasking and coordinating the activities of the Constraint Reasoner and Preference Learner, and managing interactions with the rest of CALO and the external environment. Alternative processes reflect different scheduling strategies, preferred by users or organizations, or applicable in different contexts. For example, different processes would correspond to scheduling a set of interviews with available interviewers versus allowing interviewers to sign themselves up. The SPARK framework supports agent advisability [20], and users can give advice to PTIME regarding preferred actions as well as explicit scheduling preferences. SPARK also supports adjustable autonomy, and users can explicitly or gradually delegate certain decisions to PTIME for autonomous execution.

- The basic scheduling process employs a unified representation for temporal and nontemporal constraints, and preferences. Unlike standard calendaring systems, as part of CALO, PTIME has access to rich information about the user's other activities. Thus, even when scheduling simple meetings, the Constraint Reasoner considers such issues such as workload and task deadlines, to ensure, for instance, that the user is not over-committed. The Constraint Reasoner can also explore alternative conflict resolution options using relaxation, event bumping, and explanation techniques [14].

- A core design goal of CALO is personalization. PTIME evolves by learning preferences through its *Preference Learner*. The Preference Learner in PTIME is an unobtrusive, online learner where the user's selections from suggested alternatives provide feedback to the learning algorithm. Learning has so far focused on temporal scheduling preferences over when meetings should be scheduled (e.g., time of day, day of week) and over the nature of the resulting calendar (e.g., degree of fragmentation). PTIME learns these preferences for under-constrained problems. We have showed how active learning with support vector machines (SVM) can be used to improve performance [11]. Learning a user's preferences enables PTIME to suggest increasingly desirable schedules to the user, thus providing more effective assistance.

The components of a PTIME agent utilize a variety of programming languages: the Process Controller is implemented in SPARK, the Constraint Reasoner in SICStus Prolog [27], and the Preference Learner in Java, with its underlying SVM learner being SVMLight [13], which is implemented in C. The different components, deployed as OAA agents, communicate with each other through the mechanisms provided by the OAA architecture [4].

3. DEPLOYMENT CHALLENGES

Deployment of the PTIME calendaring system required overcoming several multi-agent integration issues. We discuss them in this section, together with some of the deployment constraints imposed by the CALO personal assistant system.

3.1 Calendar Integration

Most fundamentally, a calendar scheduling system must be able to update and maintain a user's calendar. PTIME uses the *iCalendar* specification [7] for compatibility with other calendar tools on the market. An initial implementation used Microsoft Outlook, which was instrumented so that PTIME could query information from the calendar; add, delete, or modify calendar events; and listen to calendar actions performed by the user. Initial experiments with test users indicated this was a workable integration. One problem with this implementation, however, was that availability information was inaccessible when one or more users did not have their desktop environments connected.

A second integration was required with the introduction of the CALO user environment, IRIS [5], which includes a calendar application for accessing a centralized commercial calendar server.

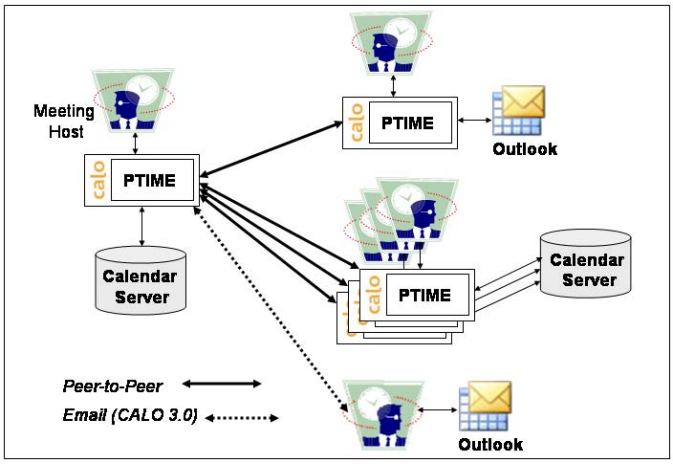


Figure 3: Communication modes between PTIME agents

The IRIS calendar application allowed PTIME to query user calendar information using the Glow API [1], but many users preferred to continue using their familiar calendaring tools, such as Outlook or Mozilla, that do not store their data on the centralized calendar server. There were also data synchronization issues, since PTIME could query the server for a participant's calendar status even when that user's CALO was unconnected.

One particularly challenging architectural constraint was that we could not make modifications to the calendaring user interface in IRIS. A core CALO 2.0 design decision was to keep "CALO actions" completely separate from the regular operations of the IRIS applications. Instrumentation of existing dialogs and development of external user interfaces were allowed. The result of this architectural constraint was two separate interfaces for adding meetings to the calendar: the standard "Add meeting here" Outlook-style dialog within the IRIS calendar application, and a separate CALO interface for scheduling through PTIME.

3.2 Multi-Agent Integration

A PTIME agent must communicate with other people's calendars to gain availability information, and negotiate with other agents to resolve conflicts and commit to calendar changes. Calendars are private; we assume that direct interrogation of other users' calendars is restricted to free/busy timeslots. Other calendars cannot be updated directly without adequate permission. PTIME agents communicate using peer-to-peer messaging or using email (Figure 3). The initial deployments (CALO 1.0 and 2.0) relied only on peer-to-peer communication via message passing. The next deployment will also offer email communication peer-to-peer as well as with the user. Thus, PTIME will be able to schedule with any online participant, whether or not she is running CALO. The protocol adopted for inter-agent negotiation is the following:

- Use direct agent-to-agent communication: resulting action would be governed by the destination's own agent.
- Agent not answering: for availability requests, ask the calendar server for a recent (possibly outdated) free/busy estimate.
- Otherwise: send an email query that could be answered by the human user or the other (PTIME) agent.

This communication protocol exists in a deployed and unbounded environment where not all participants in a meeting may be using PTIME, the shared calendar server, or indeed any automated calendar. In addition, we cannot assume that the participant's contact details exist in the user's address book. During the initial deployment of the system, a simple name resolution system asked the user for clarification of a participant's email address. However, we encountered a problem when a person had multiple email addresses and thus PTIME would duplicate messages. Although we did work around this problem, for current and future deployment we have a two-pronged solution:

- When a CALO agent (of which PTIME is a component) is initialized it sends information to a common name server. This name server stores the contact information for other agents, and this information may be queried by any agent.
- The CALO agent also provides a name resolution capability that allows a participant's name to be resolved to a known person on the name server, in the personal CALO address book, or an email address typed in by the user.

3.3 Knowledge Integration and Advisability

In addition to personal and shared 'contact' information, a personalized time management assistant must also have access to a user's organizational information, institutional rules, and personal preferences. For example, a PTIME agent may schedule a meeting with the user's superior or subordinate using different strategies. PTIME can store locally some of this information, such as scheduling preferences, but it must access other information, such as organizational relations, from the wider CALO knowledge base. PTIME adopts the CALO-wide ontology and query standards.

A foundational requirement for deployment was user–PTIME interaction. Not only must PTIME serve the user's needs in a mixed-initiative fashion [3], but it must also become a more effective assistant over time. It must build trust through its interaction with the user. Thus, PTIME permits the user to provide advice [20] that adjusts the level of autonomy, and permits her to override decisions made by the agent. A specific problem with these simple mechanisms to adjust autonomy was that the users were reluctant to change settings if the system did not provide an adequate explanation of their effects. For example, the user might advise the agent about preferred days of the weeks for meetings, but PTIME did not have the ability to illustrate to the user the resulting preference profile or to explain decisions with respect to those preferences.

4. LESSONS FROM TESTING AND USE

The CALO program is evaluated annually through a SAT-style test administered on a number of CALOs that assist users over some designated period of time. In Year 2, the test involved fifteen users, including some of the authors of this paper, using their CALOs over the period of a week to perform a variety of everyday office tasks, including tending to email, scheduling and preparing for meetings, editing documents, and performing web searches.

In preparation for the test, CALO underwent a standard software development cycle involving a series of releases each featuring increased and more robust functionality. As part of this process, CALO underwent quality assurance testing, including regression and unit tests. Each release was vetted through "dry runs" with real users. The process provided valuable feedback to the research and development teams, validating some design decisions while raising new problems, allowing time to address them before the actual test.

4.1 Different Modes of Use

PTIME was initially designed to schedule meetings on behalf of the user, including accepting or declining meetings scheduled by

other users. However, users turned out to desire PTIME support for at least two other usage modes as well.

First, users often simply wanted to add fully-specified events (e.g., seminars, weekly group meetings) to their calendars regardless of overlaps with existing events. PTIME's standard scheduling interface could be used to add such meetings, resulting in a single alternative being presented for acceptance by the user, with the option to reschedule in the case of conflicts. However, with this solution, what should have been a one-step procedure turned into a multi-step process, because of the verifications required from the user. The offer to reschedule in case of a conflict was also inappropriate for these types of events, which were already scheduled.

The second situation involved scheduling a complete agenda or related set of meetings centered on another person, for example, scheduling an interview for a job candidate. Here, the user's preferences are of no special import, since the schedule is for someone else. There may also be constraints and preferences not just on individual meetings but between meetings: for example, minimizing the free time between meetings so that the visitor is not left idle. We handled this scenario by implementing a separate set of SPARK processes for visitor scheduling, including separate facilities for obtaining the scheduling constraints from the user, and presenting agenda options. This worked reasonably well, and we will continue to do this for CALO 3.0.

This second usage mode also revealed inefficiencies in the Constraint Reasoner. While efficient enough to find alternatives for scheduling one or a few new meetings at a time, the Constraint Reasoner was often unable to efficiently find a solution in the significantly larger search space generated by visitor scheduling scenarios, consisting mostly of temporal constraints. Our solution to the search inefficiency problem was less than desirable: we modified the constraints formulated by the agenda scheduling problem to reduce the size of the search space sufficiently so that the Constraint Reasoner could find a solution if one existed. Clearly, this is only a stopgap solution since it forces users into unnatural and inexact problem formulations. We have solved this problem for CALO 3.0 by using constraint-solving techniques specifically designed for temporal constraints (discussed in Section 5.3).

4.2 Visibility into the Scheduling Process

Scheduling a meeting involving multiple participants can be a somewhat involved process, as it requires gathering availability information for all participants, finding valid solutions or—possibly—relaxing unsatisfiable constraints, before presenting the user with candidate solutions. Early on, it became clear that we needed to give users some method of tracking the scheduling process. Thus, as it undertakes the various tasks involved in scheduling a meeting, PTIME prints simple status messages in the CALO chat window to inform the user of its progress, as seen in Figure 1.

This feedback proved valuable in letting the user track PTIME's progress, particularly when unexpected situations arose. For example, obtaining another person's availability information typically takes a few seconds. In the case of delayed results, whether due to the other person wanting to personally approve all requests or due to inter-CALO communication failure, PTIME could notify the user accordingly, letting her take action if desired.

A related issue is CALO's ability to explain its scheduling decisions. Often, users want to know why CALO is unable to schedule a meeting. We addressed this by providing a minimal set of unsatisfiable constraints [14] when informing the user that a particular meeting request was unsatisfiable. This explanation mechanism also turned out to be particularly useful when debugging the system, exposing, for example, the Constraint Reasoner's flawed expectation that all events have non-zero durations. Similarly, when CALO suggests relaxed alternatives, users often want to know what constraints have been left unsatisfied so that they can make well-informed decisions. We will be extending the explanation mechanisms in CALO 3.0 to cover this situation as well, and linking PTIME's explanations with the broader CALO-wide explanation capability seen in the left-most window in Figure 1.

4.3 Qualitatively Different Schedules

Since our learned schedule evaluation was based on simple weighted sums, the best schedules tended to be clustered together. One of the factors we investigated in our active learning experiments was the presentation of qualitatively different schedules to the user, motivated by the intuition that users would be interested in seeing a variety of schedules and that the resulting feedback would also improve learning. However, as our experiments showed, the greedy approach of showing the best solutions according to the current learned model resulted in the fastest learning [11]. As a result, the fielded system used this approach to select candidates to present.

While most successful for learning, this scheme often resulted in highly similar schedules being presented to the user. For example, it was not uncommon for PTIME to present alternative candidates that began at 9:00, 9:30, 10:00, 10:30, and so on. This was particularly problematic early on, when PTIME had not yet sufficiently learned the user's preferences and was thus unable to present any good schedules within its default presentation set of 15 candidates. In our experiments with synthetic users we had relied on the user's ability to override bad solutions by scheduling the meeting manually. However, user interface limitations (see Section 3.1) prevented such a solution: users could schedule the meeting manually through the regular IRIS calendar interface, but in bypassing PTIME, the learning algorithm could receive no feedback on its suggestions. We believe a second issue is that our synthetic users did not fully represent the different kinds of preferences real users have; in particular, preferences between criteria. We are currently developing a richer, multi-criteria evaluation approach (Section 5.4) that we expect will lead to better adaptation.

4.4 User Interface Issues

Our primary focus for CALO 2.0 was scheduling functionality: encoding the procedures controlling user interaction and the scheduling process, implementing the constraint solver to generate solutions, and developing the learning module to provide personalized scheduling assistance. As a result, PTIME's user interface was functional but not particularly effective. Besides being hard to use, it prevented us from exposing all of PTIME's functionality.

First, PTIME provided a single dialog form through which users could specify a meeting request. This was a particularly poor fit to other scheduling situations. Moreover, it was also a poor fit to basic meeting scheduling, as it required users to specify constraints differently to their normal way of specifying meeting requests.

Second, while users could explicitly state general scheduling preferences to bootstrap the learning process, the mechanism was both cumbersome and unable to handle fine-grained preferences.

Third, PTIME presented its candidate schedules in a table (as seen in Figure 1), completely separate from the IRIS calendar interface. This was primarily due to the architectural constraint of keeping CALO actions distinct from the regular operations of the desktop applications. However, it turned out to be problematic in this case as it not only confused users, but also presented the candidate schedules almost completely devoid of context. Without knowing their existing schedules, users often found it difficult to select an appropriate candidate schedule.

5. ONGOING/FUTURE DEPLOYMENT

The third annual release of the CALO agent system, CALO 3.0, will deploy a new version of PTIME that will address the lessons learned from CALO 2.0 and add new capabilities to the agent. The ultimate goal is to advance the capabilities and interface to a point where people other than PTIME developers and CALO testers find value in using their CALO agent on a daily basis. Below we outline five improvements designed to bring us closer to this goal. CALO 3.0 will be evaluated similarly to CALO 2.0; the increased usability will facilitate the next round of user studies, which, along with daily use, will surely uncover new lessons about developing and deploying user-assistive agents in unbounded environments.

5.1 Multiple Modes of Use

Recall that the PTIME user interface in CALO 2.0 enabled the interactions necessary for PTIME's advanced functions, at the expense of encumbering the user with extra steps for the simple, more common, scheduling scenarios. We are addressing this issue in CALO 3.0 by defining distinct use cases and implementing a different SPARK process for each case, including different input and output facilities in the user interface.

The new version of PTIME will allow users to simply add meetings to their calendars, with or without checking for overlap and communicating meeting information to participants. In the case of overlap checking, PTIME's course of action is still an open issue. Ideally, if the user is looking at the appropriate portion of her calendar, then an overlap should be obvious and the user would not want to be bothered by superfluous confirmation requests or warnings. Instead, a simple highlighting may occur. In the more complex case of multiple participants, we must determine whether PTIME should check for participant conflicts and whether PTIME should pass on the commitment to the participants. PTIME may do background checks for such changes in the user's calendar and warn the user about conflicts with or rejection by other participants.

5.2 Multi-User Scheduling

Our informal user studies confirm that other participants' constraints and preferences often influence the meeting organizer's decisions [22]. When deciding which of many possible candidate schedules to present to the user, PTIME must reason about how each candidate schedule will affect others. This requires PTIME to retrieve from each affected party that person's schedule (as in CALO 2.0), preferences, and probable effects of the proposed schedule. Moreover, reasoning about other participants is required even when communication with another user's agent (CALO) is not available—and hence, should a shared calendar server be in use, even when the information it contains is unreliable (recall Section 3.1). Once a PTIME user selects a schedule from those presented, PTIME must negotiate with the agents of the other parties to resolve conflicts within each schedule. The goal is to remove the burden of much of this reasoning and negotiation from the user, while keeping her informed about the state of the process.

To determine the effect of a particular schedule on others, we are currently integrating a distributed querying mechanism, based on the ADOPT protocol [16], that will obtain a cost reflecting the rescheduling effort required as a result of a proposed schedule. The cost for users who do not respond (i.e., communication with their CALO fails) is estimated using the most recent knowledge about their schedules and their negotiation history.

When the user organizing a new meeting selects a candidate schedule, PTIME invites all other meeting participants, with a justification for the request. All conflicts are then addressed by other users, who will either resolve the conflicts with other CALOs or choose to not attend. Negotiation with the meeting organizer to counter-propose an alternate time is planned for CALO 4.0.

5.3 Improved Constraint Reasoning

Section 4.1 described some of the limitations of the constraint reasoning in the CALO 2.0 system. The current technology will be replaced with a Constraint Reasoner efficient for both finite-domain and temporal constraints [26, 18]. Increased performance improves the user's interaction experience with CALO, and allows PTIME to present more varied candidate schedules (Section 4.3). For instance, in the scenario of scheduling a complete agenda (e.g., for an interview visit) described earlier, preliminary experiments show that computation time can be reduced by orders of magnitude, putting larger multi-event scheduling problems within reach.

Further, the new technology will allow representation of soft constraints, enabling the user to specify, for example, preferred times for meetings, the relative importance of each participant, and events that may partially or fully overlap. In the case where many schedules satisfy all hard constraints, the soft constraints allow computation of schedules that best satisfy the user's preferences. In the case where no schedule satisfies all hard constraints, the soft constraints provide instruction for how to best relax constraints. For example, knowing the relative importance of each meeting participant would guide the Constraint Reasoner to first try omitting an unimportant participant before relaxing other constraints.

5.4 Schedule Evaluation Criteria

For meeting requests other than the simple "Add meeting here" type, PTIME presents multiple candidate schedules to its user. As stated earlier, in CALO 2.0 the candidates were ranked according to the learned schedule evaluation function, a simple weighted sum over schedule features, and a subset was selected for presentation by an active learning mechanism. Based on the results from actual use, informal user studies, and interviews of subjects from the CALO 2.0 test, we determined that this approach was inadequate for a number of reasons. The most important is its inability to express tradeoffs users have in practice: for instance, Pareto optimal schedules that balance two criteria [2].

Based on those same studies, we have identified key criteria that contribute to the user's evaluation of a schedule. These features include satisfaction of the user's general preferences on schedules, specific meeting-request constraints (both hard and soft constraints), perturbation from the previous schedule, and, as noted, the goodness of the schedule according to others. We will use these criteria in a multi-criteria evaluation framework [2] in which the relative importance of each criterion is learned.

When deciding which solution candidates to present, PTIME has multiple objectives: present desirable schedules that the user is likely to accept; present qualitatively different schedules to help the user explore the space of possibilities; and support faster learning of preferences over criteria. The second point is particularly important if the user has not yet formulated what is a 'good' solution to a specific meeting request. Only after the user has elicited in her own mind what is 'good' can we address the other two points, presenting desirable schedules and learning preferences. We are now exploring search techniques that can produce not only a single optimal solution, but also a good set of qualitatively different solutions to meet these objectives. Based on an example critiquing approach, in which solutions are presented to the user to prompt her decision process, we will present schedules designed to maximally stimulate expression of the user's preferences [28]. In conjunction, we are exploring how to vary the diversity of the solutions chosen for presentation based on a measure of entropy of the solution set.



Figure 4: Concept of future PTIME interface

5.5 Collaborative Interface

User feedback from CALO 2.0 reinforces the point that an intuitive and visually appealing user interface (UI) not only enhances human–agent collaboration, but strongly influences whether users are willing to adopt the underlying agent technology. To encourage adoption of PTIME in CALO 3.0, we will redesign the UI to improve the two main tasks: defining a meeting request and viewing/selecting candidate schedules for that request. This new UI is being designed and developed in collaboration with human-computer interaction professionals.

When defining a meeting request, PTIME's interface will allow more expressive, albeit restricted, natural language input. Although specifying preferences such as "prefer Tuesday at 2pm", "Bob is optional", and "can overlap" is possible with dialog form widgets, to include the appropriate widgets for all possible constraints and preferences would result in an overly complex interface, and such interfaces have been found to elicit untrue preferences [28].[1]

When candidate schedules are presented to the user, PTIME will display a collaborative, graphical interface to help the user rapidly understand the key differences among the set of schedules, refine or relax her input constraints, explore alternative solutions, and select a preferred candidate. This will replace the generic text-based table used in CALO 2.0.

Figure 4 shows a mockup of the future interface, including the visualization for presenting candidate schedules. "Overlaps with Coffee" (lower left-corner) indicates to the user that some hard constraints were not fully satisfied; red underlines indicate portions of the user's schedule that had to be rearranged. In a separate editing mode, PTIME will show what rearrangements are compatible with other users' schedules. The top half of the interface displays past actions of both the user and CALO to provide additional context.

[1]Users often fill out all the widgets on the form, even those marked as optional: simply asking whether the user has a preference for a feature can lead to the user having a preference for it.

6. RELATED WORK

While both commercial (e.g., Outlook/Outlook Exchange Server, Sun Java System Calendar Server) and open source (e.g., Zimbra, UWCalendar) calendaring systems abound to support centralized solutions within an institution, as closed systems [8] they suffer from the issues discussed earlier, including resistance to adoption due to loss of control and privacy. Unlike PTIME, they offer a one-size-fits-all solution that does not readily support different calendars, different modes of interaction, and different user preferences—often exhibited in the necessity of an institution-wide mandate to standardize on a particular tool. Also, unlike PTIME, which is part of a much larger cognitive assistant (CALO), these tools are dedicated calendar scheduling solutions that cannot utilize other types of information—such as workloads and deadlines—to better manage the user's schedule. Similarly, tools such as OfficeTracker and GroupReady focus on visualization and facilitating a scheduling process actually performed by a human.

In the applied research community, scheduling agents have been developed for fragments of the tasks with which a PTIME agent assists a user (e.g., [12]). Examples of other scheduling systems that operate in open environments include RCal [23] (arguably the first agent work for this scheduling task), CMRadar [17], and work in preference-aware constraint programming, such as [10].

The CMU scheduling agents RCal [23] and CMRadar [17] both distribute the scheduling problem in a similar way to that of PTIME, and the more recent CMRadar embodies a spirit similar to PTIME. A CMRadar agent helps its user schedule meetings by parsing out meeting information from email. It generates schedule options using a constraint-based scheduler that takes into account user preferences; presents candidate schedules with a graphical visualization; and enters into multi-agent negotiation. CMRadar is designed as a standalone scheduler and is tightly integrated into Microsoft Outlook, while PTIME is integrated into CALO and primarily uses the IRIS interface to access external calendar servers. PTIME is also able to support other individual calendar tools, and schedule meetings with non-CALO-related individuals. In a further architectural difference, CMRadar relies solely on email for inter-agent communication, while PTIME uses multiple modalities, with direct inter-CALO communication as its primary modality. CMRadar uses a passive learning approach to learn user preferences [21], which contrasts with PTIME's online, active learning.

Another mixed-initiative, multi-agent scheduling system is currently deployed in a closed environment to schedule nightly maintenance operations for the Hong Kong metro system [6]. The system provides core scheduling abilities and visualizations that facilitate both the scheduling of operations and the negotiation of resources between managers. Like PTIME, the UI for this system allows a window into the workings of the automated scheduler.

The CALO 3.0 UI offers multi-schedule comparison similar to RhaiCAL [9] and several non–agent-based commercial calendaring tools (e.g., GroupReady), although PTIME's visualization is part of a larger collaborative user–agent dialogue paradigm. Nonetheless, the independent development of such UIs underlines key interaction challenges, which include how an agent can ask a user to check and correct its understanding of NL input, and how it can iterate with a user to approve, reject, or modify proposed actions. Explanation in the context of collaborative and semi-autonomous assistive agents has been explored in frameworks such as Collagen and the (prototype) deployed systems built with it [24].

7. CONCLUSION

The CALO project presents a combined challenge, being both a research effort into intelligent, adaptive, personal assistants and a large-scale deployable software system. This paper has described the PTIME system, the Personalized Time Manager of a CALO agent. PTIME is designed to assist a human user in managing her time commitments, a large part of which involves managing the communication and negotiation intrinsic in scheduling meetings. We have presented the motivations behind our design decisions, the deployment challenges we faced, the issues that arose with actual usage, and our current work on addressing them.

Our experience in developing PTIME has led to the following lessons that now inform our current research and development:

- *Scheduling technology is only one part of a personalized time management solution*. To be useful, an assistive agent must address issues of privacy and authority while providing support in solving the user's most important problems. To be usable, the agent must be deployed in the real world, an open environment not fully under the user's or agent's control.
- *Multiple modes of communication and operation are critical to successful scheduling*. An open system must be able to function within the real world. For a scheduling system, this means the ability to communicate and negotiate with other agents, be they CALOs with complementary PTIME components, CALOs or human users with email, or human users with voicemail. Crucially, it also means being able to function in spite of incomplete or uncertain information.
- *A well-designed user interface is key not only to making a system usable but also to promoting the use of new functionality*. Design flaws of the initial user interface hindered adoption and masked the full set of PTIME capabilities. Strong usability allows users to explore advanced capabilities and leads to more frequent use of the overall technology.
- *A personal assistant must build trust*. Until users understand the benefits of a new technology, they will be reluctant to abandon their tried-and-true practices. An assistant can facilitate this transition by providing users with visibility into its reasoning process—for example, by providing progress reports, and explanations, especially of problems or failures.
- *An assistive agent must aim to support, rather than replace, the user's natural processes*. Since users are often unwilling to relinquish full control of their time management, an assistive agent must be designed primarily to facilitate rather than automate the user's scheduling processes. This means implementing a process that closely follows the user's own practices (including her evaluation criteria for a candidate schedule) and making explicit the various decision points to support adjustable autonomy.

Acknowledgments. We thank all the members of the CALO project, especially our collaborators at SRI, University of Michigan, and USC. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010.

8. REFERENCES

[1] D. Azuma. Glow API version 1.0 reference. July 2000.

[2] P. Berry, M. Gervasio, B. Peintner, T. Uribe, and N. Yorke-Smith. Multi-criteria evaluation in user-centric distributed scheduling agents. In *Proc. of the AAAI 2006 Spring Symposium on Distributed Plan and Schedule Management*, Mar. 2006.

[3] P. Berry, M. Gervasio, T. E. Uribe, and N. Yorke-Smith. Mixed-initiative issues for a personalized time management assistant. In *Proc. of ICAPS'05 Workshop on Mixed-Initiative Planning and Scheduling*, pages 12–17, Monterey, CA, June 2005.

[4] A. Cheyer and D. Martin. The Open Agent Architecture. *J. Autonomous Agents and Multi-Agent Systems*, 4(1):143–148, 2001.

[5] A. Cheyer, J. Park, and R. Giuli. IRIS: Integrate. Relate. Infer. Share. In *Proc. of Fourth Intl. Semantic Web Conference Workshop on the Semantic Desktop*, Galway, Ireland, Nov. 2005.

[6] A. H. W. Chun, D. W. M. Yeung, G. P. S. Lam, D. Lai, R. Keefe, J. Lam, and H. Chan. Scheduling engineering works for the MTR Corporation in Hong Kong. In *Proc. of AAAI-05*, 2005.

[7] F. Dawson, D. Stenerson, and E. H. Durfee. RFC:2445. *Internet Engineering Task Force, Network Working Group*, 1998.

[8] E. Ephrati, G. Zlotkin, and J. Rosenschein. A non-manipulable meeting scheduling system. In *Proc. of the Thirteenth Intl. Distributed Artificial Intelligence Workshop*, Seattle, WA, 1994.

[9] A. Faulring and B. A. Myers. Enabling rich human-agent interaction for a calendar scheduling agent. In *Proc. of CHI-05*, 2005.

[10] M. S. Franzin, F. Rossi, E. C. Freuder, and R. Wallace. Multi-agent constraint systems with preferences: Efficiency, solution quality, and privacy loss. *Computational Intelligence*, 20:264–286, May 2004.

[11] M. T. Gervasio, M. D. Moffitt, M. E. Pollack, J. M. Taylor, and T. E. Uribe. Active preference learning for personalized calendar scheduling assistance. In *Proc. of IUI'05*, San Diego, CA, Jan. 2005.

[12] N. R. Jennings and A. J. Jackson. Agent-based meeting scheduling: A design and implementation. *IEE Electronics Letters Journal*, 31(5):350–352, 1995.

[13] T. Joachims. Making large-scale SVM learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*. MIT Press, 1999.

[14] E. Junker. QuickXplain: Preferred explanations and relaxations for over-constrained problems. In *Proc. of AAAI-04*, 2004.

[15] W. Mark and R. Perrault. CALO: Cognitive Assistant that Learns and Organizes. `www.ai.sri.com/project/CALO`, 2005.

[16] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1–2):149–180, 2005.

[17] P. J. Modi, M. M. Veloso, S. F. Smith, and J. Oh. CMRadar: A personal assistant agent for calendar management. In *Proc. of Agent-Oriented Information Systems (AOIS 2004)*, LNCS 3508, pages 169–181, Riga, Latvia, June 2004.

[18] M. D. Moffitt, B. Peintner, and M. E. Pollack. Augmenting disjunctive temporal problems with finite-domain constraints. In *Proc. of AAAI-05*, pages 1187–1192, Pittsburgh, PA, July 2005.

[19] D. Morley and K. Myers. The SPARK agent framework. In *Proc. of AAMAS'04*, pages 714–721, New York, NY, July 2004.

[20] K. L. Myers and D. N. Morley. Human directability of agents. In *Proc. of First Intl. Conf. on Knowledge Capture*, Victoria, BC, 2001.

[21] J. Oh and S. F. Smith. Learning calendar scheduling preferences in hierarchical organizations. In *CP'04 Workshop on Preferences and Soft Constraints (Soft'04)*, Toronto, Canada, Sept. 2004.

[22] L. Palen. Social, individual and technological issues for groupware calendar systems. In *Proc. of CHI-99*, pages 17–24, 1999.

[23] R. Payne, R. Singh, and K. Sycara. Rcal: A case study on semantic web agents. In *Proc. of AAMAS'02*, pages 802–803, 2002.

[24] C. Rich and C. Sidner. COLLAGEN: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, 8(3/4):315–350, 1998.

[25] S. Sandip and E. Durfee. A formal study of distributed meeting scheduling. *J. Group Decision and Negotiation*, 7:265–298, 1998.

[26] H. M. Sheini, B. Peintner, K. A. Sakallah, and M. E. Pollack. On solving soft temporal constraints using SAT techniques. In *Proc. of CP'05*, pages 607–621, Sitges, Spain, Oct. 2005.

[27] SICS. *SICStus Prolog User Manual Version 3.12*, Oct. 2005.

[28] P. Viappiani, B. Faltings, V. S. Zuber, and P. Pu. Stimulating preference expression using suggestions. In *AAAI 2005 Fall Symposium on Mixed-Initiative Problem-Solving Assistants*, Arlington, VA, Nov. 2005.

Utility-based Collaboration among Autonomous Agents for Resource Allocation in Data Centers

Rajarshi Das
IBM Thomas J. Watson Research Center
Hawthorne, New York 10532
rajarshi@us.ibm.com

Ian Whalley
IBM Thomas J. Watson Research Center
Hawthorne, New York 10532
inw@us.ibm.com

Jeffrey O. Kephart
IBM Thomas J. Watson Research Center
Hawthorne, New York 10532
kephart@us.ibm.com

ABSTRACT

Autonomic computing, a proposed solution to the looming complexity crisis in IT, is a realm in which software agents and multi-agent systems can play a critically important role. Conversely, given its importance to a multi-billion dollar industry, it is fair to say that autonomic computing is a *killer app* for agents. Two years ago, we introduced Unity, an agent-based autonomic data center prototype that demonstrated the virtues of agency in autonomic computing applications. We discuss the road to commercialization of Unity, which entails infusing agent concepts into well-established lines of software and middleware, and discuss experiments that establish the commercial viability of utility-based resource allocation. Furthermore we examine the practicality of framing resource allocation in data centers as a collaboration between two agents, each of which is based on a commercially available product.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Autonomic computing, resource allocation, data centers, utility functions

1. INTRODUCTION

The complexity of modern-day corporate computing infrastructures makes them very difficult for administrators to install, configure, maintain, and tune. A typical medium- to large-scale enterprise may employ hundreds of applications from half a dozen different vendors running on a combination of mainframes, Linux, and Windows systems. Often, software compatibility issues necessitate running different versions of one vendor's middleware or database management systems within the same IT (Information Technology) environment. The data flows among the applications are often so complex that visualizations of them are too dense with lines to discern the interdependencies, let alone comprehend them.

IT complexity has several unfortunate consequences, of which we cite just three. First, it significantly raises the cost of making a change once the infrastructure is in place. Even after taking many precautionary measures, administrators frequently make mistakes in adding a new component or in upgrading the middleware on one of the applications with resulting costs that range from thousands to hundreds of thousands of dollars per hour. The high cost and risk of change makes IT systems ponderous—in contradistinction to the increasing call by customers for On Demand computing, which promises flexible and resilient IT systems that support nimble response to new business threats and opportunities. Second, the complexity of individual system components and their interdependencies makes it difficult for administrators to localize, diagnose, and fix problems. Often, the supposed fix does not work, or creates new problems. Third, administrators are finding it ever more difficult to keep pace with the flood of new parameters that get introduced with each new release of products. New books on database management systems from IBM, Oracle, or Microsoft are literally out of date before they are printed. Under such circumstances, it is very difficult to extract the best performance from the system, and system administrators dare not tweak knobs too much for fear of degrading performance or creating other problems.

Most major IT vendors agree that the rapidly growing complexity of IT is a looming crisis that can be averted only if systems become more self-managing. IBM coined the term "autonomic computing" in 2001 to describe its vision of a future in which large-scale IT systems would manage their own behavior in accordance with high-level objectives specified by administrators[4]. Other vendors like Hewlett Packard and Microsoft may use somewhat different terminology, but they agree generally on the importance and urgency of making systems more self-managing.

In short, realizing the vision of autonomic computing is of fundamental importance to the future of the multi-billion dollar IT industry. Moreover, there is every reason to believe that software agent and artificial intelligence technology will play a strong, multi-faceted role in achieving autonomic computing [4]. Inverting the last statement, autonomic computing appears destined to be the long-sought *killer app* for software agents and multi-agent systems [3].

In previous work[10, 13], we discussed and illustrated how an agent-oriented architecture can provide an excellent basis for autonomic computing. Individual self-managing compo-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...$5.00.

nents can be framed as agents that are responsible for managing their own behavior in accordance with agreements established with other agents. The self-management of the system as a whole (including self-optimization, self-healing, self-configuration, and self-protection) arises from more than just the individual self-managing capabilities of the agents; it is an emergent property of the mutual interactions that occur within the multi-agent system. We demonstrated the advantages of an agent-oriented approach in the context of a simple prototype data center called Unity. Middle agents [9] such as registries and sentinels were used to attain a measure of self-configuration and self-healing. Utility functions expressed in terms of service-level attributes such as response time, used in conjunction with models and optimization techniques, drove efficient allocation of server resources across multiple large-scale applications.

Unity proved to be no exception to the general rule that the flow of ideas from research prototypes to real commercial products is fraught with barriers that are not just technical in nature. One cannot ignore the strong legacy of a successful product line and abandoning it in favor of productizing a multi-agent system designed from scratch. To do so would not just be difficult for the vendor; it would risk alienating an existing customer base that is reluctant to accept radical departures from legacy solutions. The only practical course is to infuse agency gradually into existing products, demonstrating value incrementally at each step along the road towards full agency.

Therefore, during the last year, we have worked with colleagues at IBM Research and with IBM product development teams to implement some of the Unity ideas in two commercially-available products: WebSphere Extended Deployment and Tivoli Intelligent Orchestrator. WebSphere Extended Deployment (henceforth denoted WXD) is an IBM middleware offering that is used to manage routing, CPU and memory allocation, and software module placement in a large-scale multi-tiered application environment given a fixed set of server resources. Tivoli Intelligent Orchestrator comprises a couple of management components, including the Global Resource Manager (henceforth denoted TIO-GRM). TIO-GRM decides how to allocate server resources across multiple application environments, and communicates its decisions to another management component, the Tivoli Provisioning Manager, which provisions, configures, and redeploys server resources in accordance with the re-allocation decisions made by TIO-GRM.

The remainder of this paper is structured as follows. Section 2 sets the stage for our current work by providing a historical perspective of our past efforts in getting WXD and TIO-GRM to interact with each other. In section 3, we discuss WXD and TIO-GRM and our modifications to them for utility-based collaboration in greater detail. Then, in section 4, we describe two experiments designed to show that, in practical usage scenarios, our scheme allocates server resources in accordance with an administrator's service-level objectives in the face of dynamically changing workloads and objectives. Finally, in section 5, we summarize our main points and provide thoughts on future directions.

2. BACKGROUND

Conceptually, the capabilities of WXD and TIO-GRM are complementary, with WXD managing to service-level objectives as well as possible within fixed resource constraints, and TIO-GRM operating one level above, deciding how to allocate resources across multiple independent large-scale application environments as outlined in Figure 1.[1]

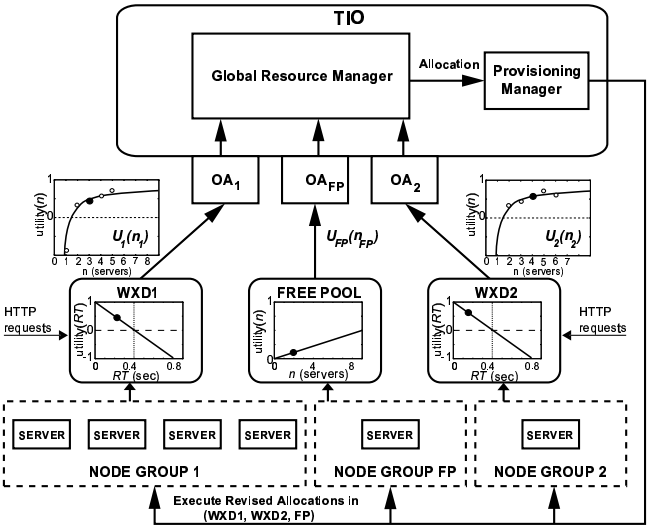


Figure 1: WebSphere Extended Deployment (WXD1 and WXD2), Tivoli Intelligent Orchestrator (TIO), the Objective Analyzers (OAs), and their mutual interactions.

Yet the products as originally conceived did not dovetail very well. This is not surprising, as WXD and TIO-GRM were developed completely independently of one another (indeed, the TIO-GRM technology was obtained by IBM via acquisition) with very different assumptions about environments, span of control, and the nature of the objectives to which they would manage. Therefore there was a great desire on the part of IBM to get these two independently-developed software entities, each capable within their own realm of expertise, to collaborate effectively to achieve system-wide performance optimization.

In recent work [1], we reported early progress in getting WXD and TIO-GRM to interact fruitfully with minimal changes to the existing products. One particularly daunting challenge was the mismatch between the way objectives were expressed by WXD and TIO-GRM. WXD drives its internal behavior using a simple form of service-level utility function expressed in terms of average response time of the various service classes. In other words, an administrator creates a function that describes the value of achieving a given average response time target. This is entirely consistent with the utility-based approach used by Unity. In order to have adopted the Unity approach, WXD should then have computed a resource-level utility $U(n)$ expressing the value of having a given number of server resources at its disposal, and sent this curve to TIO-GRM. TIO-GRM would then have combined similar estimates from other application environments (managed by WXD, or possibly some similar entity) to compute a best overall server allocation that maximized total system utility in each allocation-decision period.

However, the hitch is that TIO-GRM does not view the

[1]An application environment might contain a set of applications running on behalf of one customer within a data center, for example.

world in terms of utility. Rather, TIO-GRM expects to receive "Probability of Breach" (PoB) curves, which describe the estimated probability for the application environment to breach its contract as a function of the number of servers with which it is provided. The resource-level utility curves required by Unity and the Probability of Breach curves expected by TIO-GRM are similar in form, but completely incompatible. WXD is similarly incompatible with TIO-GRM—there is no *breach* of contract in WXD; there is only lower or higher utility.

The approach used in [1] was for WXD to output "speed" numbers that hinted at how urgently it needed additional resources. Then, an *Objective Analyzer*—an extension of WXD designed to mediate interactions with TIO—did a highly heuristic transformation of the speed numbers into a probability of breach. The resulting system was shown to work in a qualitative sense, with more resources being supplied to a single application environment when the load was increased. However, it was difficult to quantify the extent to which WXD was receiving the right amount of resource.

In [12], a follow-on to [1], the form of communication between WXD and TIO-GRM was improved. With help from our colleagues at IBM Research, WXD was instrumented to produce from its internal queuing models the information needed to construct the resource-level utility curves $U(n)$. This achievement required WXD's internal model to operate in a what-if mode, by which it would express the utility it thought it would experience if there were n servers. This is a fairly sophisticated analysis, as it entails some computation of how internal control parameters would be changed if more or less resource were made available. However, a problem remained. Although WXD was now articulating its resource-level utility curve $U(n)$, TIO-GRM was incapable of consuming it: it still expected to receive PoB curves. Accordingly, we developed a heuristic for converting from $U(n)$ to PoB(n) by mapping extrema and midpoints of these curves to one another on the basis of intended behavior. While this scheme was somewhat better at conveying the true intent of the WXD objectives, it was still difficult to quantify the degree to which this was so.

In this paper, we bring our product implementation significant steps closer to the original agent-based vision of the Unity prototype. As we shall describe, we have worked with IBM product developers to create an experimental version of TIO-GRM that accepts utility input directly, eliminating the need for heuristic conversions from utility to PoB. Now TIO-GRM can allocate resources in a manner that truly reflects the intent of the WXD administrator. Thus utility information—and information derived from it via models and optimization—serves as a clean, consistent, universal means for expressing objectives and managing the system to those objectives automatically. Furthermore, we go beyond the original Unity prototype in that we use utility to represent the relative importance of retaining servers in the free pool, a point that will be further explored in section 4.

3. THE COLLABORATING AGENTS

In this section we discuss the two main collaborating agents, based on WXD and TIO-GRM—both of which are commercially available IBM products. Specifically, we use WebSphere Extended Deployment v6.0, which includes the extensions to WXD that supply utility estimates, and we use an experimental version of Tivoli Intelligent Orchestrator that is based upon TIO v3.1, but extended to support utility functions. As will be discussed, each of these agents is itself composed of multiple autonomous interacting entities that have agent-like properties.

3.1 WebSphere Extended Deployment

WebSphere Extended Deployment (WXD) is an advanced middleware application server environment. While a full enumeration of the capabilities of WXD is beyond the scope of this paper, we will discuss here those features of the environment that are most relevant to the paper.

WXD provides the ability for administrators to set service-level objectives (in the form of performance targets) for the web applications installed within the environment. It provides a load balancing reverse proxy (called the "On Demand Router", or ODR) that directs requests to individual application servers such that the specified service-level objectives are met. WXD also starts and stops instances of individual application servers as the extant load changes.

A WXD installation consists of a central management server (called the "Deployment Manager"), one or more ODRs, and several backend nodes (physical machines) that run the applications themselves. Administratively, the backend nodes are grouped into one or more "Node Groups". Each node can be in multiple Node Groups. The user's applications are installed into "Dynamic Clusters", which are in turn backed by the Node Groups—each application is installed into one Dynamic Cluster (although each Dynamic Cluster can host many applications), and each Dynamic Cluster is backed by one Node Group.

Each backend node can run multiple instances of WebSphere Application Server. WXD makes decisions at runtime concerning which instances to start and when. The user's applications are hosted in these individual instances.

The ODR determines the appropriate application for each inbound request by examining the URI. Then, from the set of servers running the application, it selects a server to process the request, taking into account session persistence, etc. The server handles the request in the usual way, and sends the response to the ODR, which relays it to the client.

Clearly, WXD would not be able to meet the service-level objectives if it merely mediated incoming requests. Sometimes, an application is flooded with so many incoming requests that it cannot meet its service-level objectives. Consequently, WXD is also able to start and stop server instances. Thus when the load for a given application is high, WXD can start additional instances of the WebSphere Application server created for the Dynamic Cluster containing the application.

3.1.1 WXD as agents

Rather than having all the management functionality and logic in WXD in a single place, it is distributed through the system as separate components. These components can be regarded as independent, yet inter-connected, agents. The performance management aspects of WXD are handled by three such agents.

The first such agent is the placement control agent—this determines which application server instances should be running at any given time on each node in the system. The approach taken to solving the placement problem is that discussed in [5].

The second such agent is the flow manager—this manages

the various service classes, trading one off against the other to obtain the highest overall system utility. The flow manager takes the current placement (which servers are available for each Dynamic Cluster) as a given, and calculates the concurrency limits for the various Dynamic Clusters that the ODR uses for request assignment. The algorithms used by the flow manager are essentially those described in [6].

The flow manager is particularly relevant to this paper, as it is the component that the TIO-GRM asks for system utilities. The flow manager is not only capable of considering the system as it currently stands, but also of considering possible future allocation changes. The value to WXD of these possible future allocations (in addition to the value of the current allocation) is expressed in terms of a utility function $U(n)$, where n is the node count.

The third such agent adjusts the load balancing weights used in request routing; its function is not pertinent to this paper, and it will not be mentioned further.

The three performance agents are inter-connected, in that information calculated by each agent is used by the other agents. For example, the flow manager agent computes "demand constraints" which the placement control agents uses as input to the placement decision.

3.1.2 The WXD Objective Analyzer

As discussed later in Section 3.2, TIO-GRM provides a mechanism by which components called "Objective Analyzers" (OAs) can be plugged in to provide performance information for specialized systems that TIO-GRM is managing.

WXD provides such an OA, in order that higher-level management and administration functionality can be carried out automatically. As TIO-GRM provides a flexible scripting and automation environment for system management, it can be used to carry out higher-level reallocations that WXD cannot do on its own—specifically, TIO-GRM is able to move nodes between Node Groups. In order to do this, as was discussed above, properties of the nodes other than those controlled by WXD need to be changed—for example, database clients might need to be installed or reconfigured so that a node is capable of running the applications that will be deployed upon it once it is a member of a certain Node Group or set of Node Groups.

A hierarchy of management can therefore be seen between WXD and TIO-GRM—WXD handles management within each Node Group, and is aware of low-level performance properties of the applications, whereas TIO-GRM handles management between Node Groups, and between WXD and other systems in the data center.

The OA that currently ships with WXD v6.0 obtains the $U(n)$ function discussed above from WXD for each Node Group (see Section 3.3 for more information on the communications between the various agents), and uses mechanisms discussed in [12] to convert those utility curves to probability of breach curves PoB(n). Some information is lost in the conversion because WXD's utility values are fit to negative exponential curves with a small number of parameters, which are then mapped very nonlinearly to parameters describing sigmoids that represent the PoB curves required by TIO-GRM. In addition, as is discussed in Section 4, the PoB values are themselves subjected to a nonlinear transformation to "fitness" values after being passed to TIO, which can distort and magnify any inaccuracies introduced by the OA conversion.

In order to support direct use of utility values by TIO, the commercial OA was modified to produce an experimental OA in which the mapping of $U(n)$ to PoB(n) was removed. This removal introduced a slight problem that is not an issue for the commercial OA. TIO-GRM requires input curves PoB(n) or $U(n)$ that cover a broad range of n (from 0 or 1 to some upper bound on the total number of servers). However, WXD does not compute $U(n)$ for all n; for a variety of reasons it computes $U(n)$ only for a small range of n that does not differ from the current allocation n_t by more than 2 in either direction. In the commercial OA, the partial range is not a problem because $U(n)$ is used as input to a curve-fitting procedure that yields a complete PoB(n) for all n. In the experimental OA, the problem of missing data values $U(n)$ for $n < n_t - 2$ and $n > n_t + 2$ is solved by extrapolating them from the $U(n)$ in the limited range computed by WXD. The resulting utility curve $U(n)$ for *all* n is input to TIO-GRM.

3.2 TIO Global Resource Manager

Tivoli Intelligent Orchestrator (TIO) is an automated resource management solution for corporate and Internet data centers. It allows automatic policy-based management of computing resources by automating three key data center processes: infrastructure provisioning, capacity management, and service level management. TIO allows system administrators to create, customize, assemble and store for further use a large variety of workflows that automate various data center processes from configuring and allocating servers, to installing, configuring and patching software.

The TIO component of interest for this paper is the TIO Global Resource Manager (denoted TIO-GRM), which is charged with making resource allocation decisions that are optimal and sufficiently stable. An optimization component in TIO-GRM performs a tree search over the possible combinations of server allocations and proposes an allocation with the highest overall system "fitness", taken to be the sum of the individual fitnesses of each application environment that is requesting resources from TIO.

In the commercially available version of TIO-GRM, the fitness $F(n)$ for an individual application environment is a nonlinear transformation of its PoB(n). While TIO-GRM supports any transformation function, the default transform assigns the highest fitness values to PoB values that are in middle range. Typically, this helps prevent underprovisioning and overprovisioning. However, since the fitness is defined strictly within TIO-GRM, there is no way to ensure that the fitness function reflects the intent of the adminstrator as expressed in the utility functions defined in WXD. Moreover, even if the administrator had access both to WXD and to TIO-GRM, the doubly compounded nonlinear transformations (from utility to PoB in the OA, and from PoB to fitness within TIO-GRM) would make it excessively difficult for an administrator to construct a fitness function that properly reflected his objectives.

Therefore, to allow the utility-based objectives defined in WXD to flow transparently to TIO-GRM, the commercially available TIO-GRM was extended to support utility functions directly. This was achieved by setting $F(n) = \text{PoB}(n) = U(n)$. In other words, aside from the extrapolation provided by the OA, the fitness function is no longer an independent, carefully engineered function that is separately defined within TIO-GRM. It is determined com-

pletely by (and essentially equal to) the utility estimates from WXD. The optimization component in the TIO-GRM is quite generic, and therefore required no changes; as always, it simply maximized the sum of the fitness functions of the individual application environments, which in this case is equivalent to maximizing the overall system utility.

3.3 Inter-agent communication

There are two distinct communication paths between WXD and TIO-GRM. The first path conveys utilities estimated by WXD to TIO-GRM. Periodically, when TIO-GRM desires utility estimates, it calls the OA (which is in the same JVM). In turn, the OA makes a JMX call[2] to WXD requesting from WXD's flow manager the current utility estimates.

The second communication path executes the resource allocation decisions made by TIO-GRM by adding or removing servers in the WXD environments and the free pool. There are two methods, each of which permit TIO to execute OS-level commands on the machines it wishes to modify. The first method involves use of the "Tivoli Common Agent" (TCA), a client-side agent used by various IBM management software. The second method, which was used in our experiments, does not require TCA, but uses SSH (Secure Shell) to execute the commands instead.

When TIO decides to add a node to a Node Group, it does so by first executing commands on the node to add that node to the WXD system ("federating the node"), and then executing commands on the WXD Deployment Manager to move the node into the Node Group. Conversely, when TIO decides to remove a node from a Node Group, it does so by first executing commands on the WXD Deployment Manager to take the node out of the Node Group, and then executing commands on the node to remove that node from the WXD system (this process is known as "unfederating the node").

4. EXPERIMENTS

In this section, we describe two experiments, each of which is based on what we anticipate will be a common usage pattern for the TIO-GRM and WXD agents in deployed systems. In different ways, the two experiments demonstrate efficacious collaboration between these agents to achieve service-level objectives specified by administrators—even in the face of dynamically changing workloads and dynamically changing objectives. First, we describe an experimental setup common to both experiments, and then we discuss each experiment and its results.

4.1 Experimental Setup

The following setup, which applies to both of the experiments, is illustrated in Figure 1. An installation of TIO-GRM was configured to manage two independent instances of WXD, each running its own application with time-varying workload on a single dedicated Node Group contained in a single dedicated Dynamic Cluster. Additionally, TIO-GRM was configured to manage a free pool of homogeneous servers from which the two WXD instances could be supplied. For each WXD instance, the Deployment Manager and the ODR were installed on separate dedicated servers, amounting to four servers in all. (i.e. four servers were used in all for the WXD management).

[2] [2] describes WebSphere's use of JMX.

We employed the same web-based application for both instances of WXD. Each application consisted of a single service class. For each service class, WXD permits an administrator to establish a simple utility function expressing the value of attaining a given average response time[3]. The utility function is elicited via a simple template: the administrator inputs a response time target RT_0 and selects one of seven importance levels ranging from very low to very high. These two parameters are mapped to a simple utility function $U(\mathrm{RT})$, in which the utility drops monotonically from 1 at $\mathrm{RT} = 0$ to 0 at the response time target RT_0. For response times greater than RT_0, the utility is negative and linearly decreasing with a slope governed by the importance level. In all of our experiments, the response target was $\mathrm{RT}_0 = 0.4$ sec, and the importance level was selected to be "very high", which translated to a constant slope for all response times. (For lower values of importance, the absolute value of the slope is reduced to reflect a lower penalty for response times above the target.) Thus the utility was given simply by $U(\mathrm{RT}) = \frac{\mathrm{RT}_0 - \mathrm{RT}}{\mathrm{RT}_0}$ for all possible values of RT, with $\mathrm{RT}_0 = 0.4$ sec.

The demand in each application, quantified by the number of clients sending requests to the ODR, was generated using a single closed-loop load generator [7] with an adjustable number of clients. In principle, each request required the same amount of CPU cycles. The think time for each client in the closed-loop load generator was drawn from an exponential distribution with a mean of 0.125 sec.

A total of six homogeneous server machines were available for our experiments. Each server machine was an IBM eServer xSeries 335 machine with two Intel Xeon 3.06GHz processors (with 512KB of Level 2 cache) configured with hyperthreading enabled. Each machine had 2.5GB of RAM. The properties of the application and the server machines were such that each machine could serve up to 30-40 clients and still meet RT_0 of 0.4 sec.

TIO was installed on a separate machine and the Global Resource Manager was configured to allocate server machines to the two WXD instances based on *resource utility* information $U(n)$ provided by the two OAs every 30 seconds. The free pool was placed on the same footing as the two WXD instances by ascribing a utility u_f to each server in the free pool, i.e. $U_f(n) = nu_f$. In our experiments, we typically set $u_f = 0.05$, although as reported below we also tried setting it to higher ($u_f = 0.4$) and lower ($u_f = 0.01$) values. While it would have been more elegant to represent the third pool by a third (very simple) OA, for the sake of expediency the free-pool utility was computed inside our modified version of TIO-GRM. From discussion with developers and TIO customers, it is clear that there is a good, solid grounding for the use of a free-pool utility. A server that is idling in the free pool may be less costly than one employed by WXD because some hardware contracts specify a lower rate for idle servers, or because multi-server applica-

[3]WebSphere Extended Deployment supports performance measures other than average response time, such as percentile response time. In these cases, utility functions are defined similarly to what is described here for average response time. We have conducted a similar series of experiments on percentile response time, and we have also experimented with heterogeneous applications and multiple service classes, but for reasons of clarity and space we focus only on the average response time results in this paper.

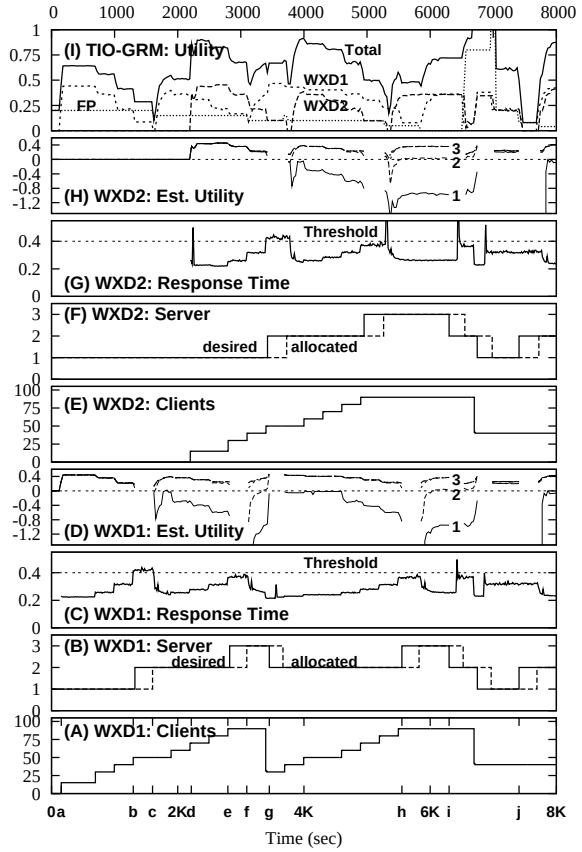


Figure 2: Summary of Experiment 1 in nine time series plots.

tion or middleware software licenses may take into account the number of servers on which the software is installed. An idle machine costs less in terms of power and money to run. Moreover, an idle server is advantageous because, if there is a sudden urgent need, it can be employed more quickly than one that is in currently in service, as there is no need for a potentially lengthy de-installation. Thus in many cases a data center administrator would like to ascribe a utility value to the size of the server free pool.

TIO-GRM guaranteed that at least one server machine would always be made available to each instance of WXD. Thus, based on the current level of load, each WXD was essentially competing with the other WXD instance and the free pool for the remaining four server machines. TIO-GRM's overall goal was to find the allocation that maximized the total system utility, which was taken to be the sum of the utilities derived from the two WXD instances and the free pool.

In all of our experiments, TIO was run in the *automatic* mode, allowing it to execute the resource allocation decisions with no manual intervention.

4.2 Experiment 1

Results from an experiment with TIO-GRM managing two WXD instances, denoted henceforth as WXD1 and WXD2, are summarized in Figure 2. The figure shows nine time series plots over a period of 8000 seconds (2 hours and 20 minutes). From bottom to top, they are:

(A) the number of clients requesting service in the application installed on WXD1,
(B) the number of servers scheduled to be allocated and the number of servers actually allocated to WXD1,
(C) the average response time achieved by WXD1,
(D) the utility values $U(n)$ reported by WXD1 to TIO-GRM,
(E) the number of clients requesting service in the application installed on WXD2,
(F) the number of servers scheduled to be allocated and the number of servers actually allocated to WXD2,
(G) the average response time achieved by WXD2,
(H) the utility values $U(n)$ reported by WXD2 to TIO-GRM, and
(I) the total actual system utility obtained by TIO-GRM from WXD1, WXD2, and the free pool (FP).

Notable times are indicated by labeled letters at the bottom of the figure.

At time 0, each WXD had one allocated server machine as per TIO-GRM's guarantee, and the number of clients for each application was zero. Hence all utility values reported by WXD1 and WXD2 were initially zero. The total system utility obtained by TIO-GRM was derived solely from the free pool, and since there were four machines in the free pool, this value was $0.05 \times 4 = 0.2$.

At time **a**, the number of clients in the application running on WXD1 suddenly jumped to 15. WXD1 already had a server in its Node Group, and after a small transient period the server provided an average response time of about 0.23 sec. The actual utility reported by WXD1 to TIO-GRM was 0.44. Note that WXD1 also estimated that there was virtually no increase in utility $U(n)$ with increasing n.[4] Because of the utility associated with the free pool, TIO-GRM correctly decided not to allocate any additional servers to WXD1.

From time **a** to **b**, the number of clients in WXD1's application increased in a series of steps. With each such step, it is possible to associate a corresponding increase in response time and a decrease in the utility values. At time **b**, when the number of clients increased to 50, the average response time sharply jumped to over 0.4 sec. This drop in performance also caused a drop in WXD1's utility, and since $U(2)$ was slightly more that $U(1)$, TIO-GRM began the process of allocating an additional server to WXD1. The allocation entails copying application software and other necessary infrastructure information to the new server machine, a process that may take more than 5 minutes.[5] However, once the server is finally made available to WXD1 at time **c**, there is a drop in response time and a corresponding increase in utility reported by WXD1 to TIO-GRM. The utility of the free pool on the other hand drops to 0.15. Interestingly, even after the server has been allocated to WXD1, there is a substantial transient period before the response time and utilities reached their corresponding near-asymptotic values.

Beyond time **c**, the number of clients associated with WXD1 continues to climb to 90, putting an increasing strain on the two servers in WXD1. In response to further decrease in the reported utility, TIO-GRM decides to allocate the third server to WXD1 at time **e**. A similar series of

[4]This is because the mean queue length was zero for this level of load on a single server. Thus additional servers would not have had any effect.

[5]In its current implementation, the OA stops reporting the estimated utility values to the TIO-GRM while an allocation (or de-allocation) process is in progress. Hence, we see the gaps in the utility values in Figures 2(D) and (H).

events ensues, and once the new server is online at time **f**, the response time drops accordingly.

At time **g**, the trend of increasing clients is reversed, with the client count dropping suddenly to 30. All of the estimated utility values in WXD1 rise dramatically, as a result of which TIO-GRM determines to take away one server from WXD1. In this case, once the server is returned to the free pool, XD1 is still able to provide a response time well below the threshold of 0.4 seconds.

While TIO-GRM and WXD1 were involved in this exchange of server machines, note that back at time **d**, the WXD2 application took on some clients for the first time, and the number of clients began to increase with the passage of time. Accordingly, beyond time **d**, when WXD2's performance begins to sag noticeably, TIO-GRM granted an additional server to WXD2. The allocation of a server to WXD2 and the de-allocation of a server from WXD1 overlap near time **g**, underscoring TIO-GRM's robust ability to handle multiple allocation processes at the same time.

Later, at time **h**, we stress-tested our experimental setup with very high load on WXD1 and WXD2. Soon after TIO-GRM's decision to allocate the last remaining server in the free pool to WXD1 at time **h**, both WXD1 and WXD2 make use of three servers each, enabling them to successfully meet their response time targets and obtain relatively high utility.

To illustrate the ability of the system to respond dynamically to changes in objectives, and to demonstrate the naturalness of the free pool utility, the free pool utility is increased at time **i** from its default value of 0.05 to 0.4. As discussed above, this may represent a change in the licensing structure, or in the perceived advantage of retaining machines in the free pool. Fig. 2 demonstrates that TIO-GRM is able to take this change in stride with no need for stopping and restarting. In this case, TIO-GRM decided to pull one server away from environments WXD1 and WXD2. While this action causes the response times of WXD1 and WXD2 to rise noticeably, and their utilities to fall somewhat, this is more than compensated by the increased utility of the two extra servers in the free pool.

Later, when the load on both WXD1 and WXD2 has dropped, a complementary series of steps is triggered at time **j**, when the system administrator chooses to decrease the value of each individual server in the free pool to 0.01. As a result, TIO-GRM now relinquishes an additional server from the free pool to each WXD.

This experiment demonstrates effective utility-based collaboration among three autonomous agents, resulting in robust, dynamic allocation and de-allocation of server machines that faithfully reflects service-level objectives that may themselves be dynamic in nature.

4.3 Experiment 2

The purpose of the second experiment was to explore the effectiveness and reliability of the collaborative agents in a more dynamic setting with more realistically time-varying workloads. The experimental setup was substantially similar to that of Experiment 1. The main difference was the use of a time-series model of measured web traffic developed by Squillante et al. [8], which provides a realistic emulation of stochastic bursty time-varying demand. The number of clients in the closed-loop load generator was reset every minute according to this model with hard lower and upper thresholds of 5 and 125 clients respectively. The time

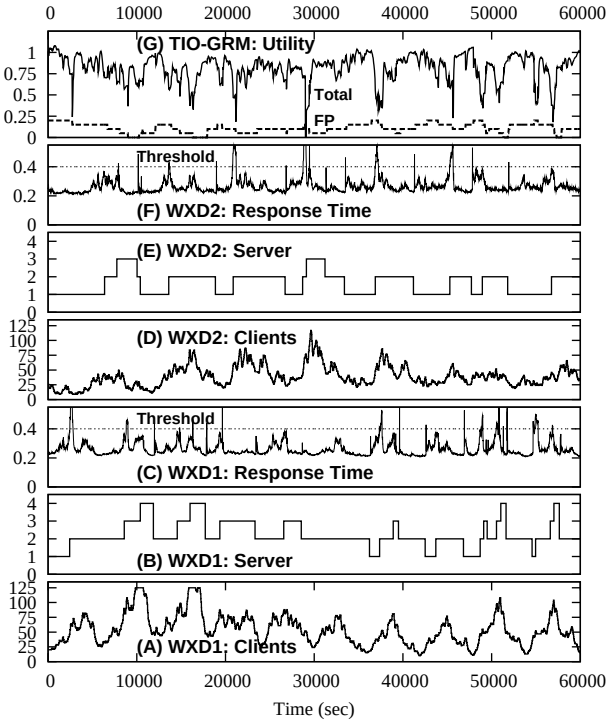


Figure 3: Summary of Experiment 2 in seven time series plots.

series of the workload on each application was generated independently, with the application on WXD1 experiencing slightly higher average load conditions than the application on WXD2.

TIO-GRM was again set in the "automatic" mode, and the experiment ran uninterrupted for a period of over 60000 seconds (16 hours and 40 minutes).

Results from Experiment 2 with TIO-GRM managing WXD1 and WXD2 are summarized in Figure 3. The figure shows seven time series plots; from bottom to top, they are:

(A) the number of clients requesting service in the application installed on WXD1,
(B) the number of servers set to be allocated on WXD1,
(C) the average response time achieved by WXD1,
(D) the number of clients requesting service in the application installed on WXD2,
(E) the number of servers set to be allocated on WXD2,
(F) the average response time achieved by WXD2, and
(G) the total actual system utility obtained by TIO-GRM from WXD1, WXD2, and the free pool (FP).

The results show that, over the course of over 16 hours, all the agents collaborated successfully. Through a harmonious exchange of servers, each instance of WXD was generally able to meet the response time threshold. Dynamic systems with delayed feedback often suffer from nonlinear effects such as hysteresis under time-varying conditions. The XD-TIO-GRM system studied here experiences long transients and allocation delays that can last for over 5 minutes—conditions which under some circumstances can induce thrashing behavior. However, for the time-varying workloads experienced by WXD1 and WXD2 in this experiment, rapid repeated allocation and de-allocation of servers by TIO-GRM was not observed in spite of the allocation delays.

5. CONCLUSIONS

In this paper and in prior work [1, 12], we have argued and demonstrated that software agents and multi-agent systems have a centrally important role to play in autonomic computing, and that, given that it may be the only answer to solving a major problem in the multi-billion dollar IT industry, autonomic computing is a *killer app* for software agents and multi-agent systems. One important proof point in that argument was Unity, an autonomic data center prototype that used an agent-oriented architecture to configure, heal, and optimize itself.

This paper focused on the road to productization for the self-optimizing portion of Unity, which entailed the elicitation, usage, transformation and propagation of utility functions. Direct productization of Unity was not feasible, given the considerable legacy of IBM products that occupy the space of automated provisioning and workload management. Instead, we set about to gradually infuse the most interesting and important elements of Unity into existing products. This paper describes an important breakthrough in that process, in which we have succeeded in implementing in two commercial products the main capabilities that are needed to produce and consume what-if utility information. Two experiments demonstrated effective utility-based collaboration between two products that formerly viewed the world in very different and incompatible terms, resulting in effective cooperative interactions between workload management and provisioning. Indeed, we were even able to take a step beyond the original prototype, showing that management of the free pool (which had not been present in the prototype, and which had been managed heuristically in the product) could be done elegantly, using the same utility-based methods that are used for managing application environments.

Of course, our work is hardly finished. WXD and TIO-GRM still lack features that one would expect in full-fledged agent systems. To cite just one desirable feature, it would be beneficial for the agents to exchange what-if information using more general, flexible semantics. For example, TIO-GRM might find it advantageous to ask for more detailed information on estimated utilities for individual service classes, rather than a single overall utility. We have begun to work on more flexible languages that allow agents to specify and describe hypotheticals. We are also exploring various uses for reinforcement learning techniques in this context [11]. Looking further beyond, we would like to incorporate in these products the other self-managing capabilities in the original Unity that gave rise to its self-healing and self-configuring properties. And looking beyond still further, there is practically infinite scope for employing agent concepts, architectures and technologies—not just for workload management and provisioning, but within autonomic computing systems quite generally.

6. ACKNOWLEDGMENTS

We thank our IBM Research colleagues Giovanni Pacifici, Malgorzata Steinder, Mike Spreitzer, and Asser Tantawi for much discussion and consultation on WebSphere Extended Deployment, without which this project would have been impossible. We are also indebted to U. Gopalakrishnan of IBM India for making several essential enhancements to the workload generator, giving us the flexibility we needed.

7. ADDITIONAL AUTHORS

Additional authors: Hoi Chan (IBM Thomas J. Watson Research Center, email:hychan@us.ibm.com) and Paul Vytas (IBM Software Group, Tivoli, email:pvytas@ca.ibm.com).

8. REFERENCES

[1] D. M. Chess, G. Pacifici, M. Spreitzer, M. Steinder, A. Tantawi, and I. Whalley. Experience with collaborating managers: Node group manager and provisioning manager. In *Proceedings of the Second International Conference on Autonomic Computing*, 2005.

[2] R. Cundiff. System administration for websphere application server v5. http://www-128.ibm.com/developerworks/websphere/techjournal/0302_cundiff/cundiff.html, 2003.

[3] M. Fisher, J. Muller, M. Schroeder, G. Staniford, and G. Wagner. Methodological foundations for agentbased systems. *Knowledge Engineering Review*, 12(3):323–329, 1997.

[4] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–52, 2003.

[5] T. Kimbrel, M. Steinder, M. Sviridenko, and A. Tantawi. Dynamic application placement under service and memory constraints. In *4th International Workshop on Efficient and Experimental Algorithms*, Santorini Island, Greece, May 2005.

[6] R. Levy, J. Nagarajarao, G. Pacifici, M. Spreitzer, A. Tantawi, and A. Youssef. Performance management for cluster based web services. In *8th IFIP/IEEE International Symposium on Integrated Network Management (IM 2003)*, 2003.

[7] D. Menasce and V. A. F. Almeida. *Capacity Planning for Web Performance: Metrics, Models, and Methods.* Prentice Hall, 1998.

[8] M. S. Squillante, D. D. Yao, and L. Zhang. Internet traffic: Periodicity, tail behavior and performance implications. In *System Performance Evaluation: Methodologies and Applications*, 1999.

[9] K. Sycara. Multi-agent infrastructure, agent discovery, and middle agents for web services and interoperation. In *Multi-agents systems and applications*, pages 17–49. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[10] G. Tesauro, D. M. Chess, W. E. Walsh, R. Das, A. Segal, I. Whalley, J. O. Kephart, and S. R. White. A multi-agent systems approach to autonomic computing. In *AAMAS*, pages 464–471. IEEE Computer Society, 2004.

[11] G. Tesauro, R. Das, W. E. Walsh, and J. O. Kephart. Utility-function-driven resource allocation in autonomic systems. In *Second International Conference on Autonomic Computing*, 2005.

[12] I. Whalley, A. Tantawi, M. Steinder, M. Spreitzer, G. Pacifici, R. Das, and D. M. Chess. Experience with collaborating managers: Node group manager and provisioning manager. To appear.

[13] S. R. White, J. E. Hanson, I. Whalley, D. M. Chess, and J. O. Kephart. An architectural approach to autonomic computing. In *First International Conference on Autonomic Computing*, 2004.

ACTIVE: Agile Coordinator Testbed Integrated Virtual Environment

G. Emami, J. Cheng, D. Cornwell, M. Feldhousen, C. Long, V. Malhotra, I. Starnes
Global InfoTek, Inc.
1920 Association Drive
Reston, VA 20191-1545
1-703-652-1600
{gemami, jcheng, dcornwel, mfeldhousen, clong, vmalhotra, istarnes} @globalinfotek.com

L. Kerschberg*, A. Brodsky**
George Mason University, *KRM, Inc. and **Adaptive Decisions, Inc.
ISE Department, MS 4A4
Fairfax, VA 22030-4444
1-703-993-1640
{kersch, brodsky}@gmu.edu

X. Zhang
University of Mass-Dartmouth
Computer and Information Science
285 Old Westport Rd.
North Dartmouth, MA 02747-2300
1-508-999-8294
x2zhang@umassd.edu

ABSTRACT

This paper describes the specification, design and development of ACTIVE, a testbed for the testing and simulation of large-scale agent-based systems. ACTIVE is being developed as part of DARPA's Coordinators program. The goal of the ACTIVE testbed is to support the simulation of large collections of distributed cooperating agents in solving constrained scheduling problems.

Categories and Subject Descriptors

I.2.11 {**Artificial Intelligence**}:Distributed Artificial Intelligence.

I.2.5 {**Artificial Intelligence**}:Programming Languages and Software.

I.2.8 {**Artificial Intelligence**}:Problem Solving, Control Methods and Search.

General Terms

Algorithms, Measurement. Performance. Design, Reliability, Experimentation, Languages, and Verification.

Keywords

Agent testbed, Scalable testbed, Test Harness, CTAEMS, GMASS, TAEMS, MASS.

1. INTRODUCTION

The Coordinators Program seeks to research, develop, and deploy agent-based systems in support of emergency responders in disaster relief scenarios as well as war-fighters in combat situations. The Coordinators Program is managed by Dr. Tom Wagner of the Defense Research Projects Agency (DARPA) — The purpose of this program is to support fielded units in adapting mission plans to changing situations. The problem involves distributed coordination over large interconnected mission structures that change dynamically. Once deployed, the units are physically distributed. Authority and information are also distributed. The Coordinators Program will use agents to interact and modify plans according to constraints and select the best options for modifying plans. It is a complex task involving a great deal of low level information exchange and analysis that is done better and more quickly by machines than humans and will free humans for high level tasks and decision making

The goal of the ACTIVE system is to provide the infrastructure to support the configuration, simulation, execution, monitoring and evaluation of teams of distributed agents that cooperate in solving large-scale scheduling problems that cannot be solved in a centralized fashion. The distributed nature of the problem, the sheer number of agents involved and the complexity of the tasks together with their constraints, invites new approaches to solving this problem

It is essential to automate the tests, in order to support the number of software agents needed to support these personnel. Large scale testing will be required. Hundreds or even thousands of tests must be run automatically and unattended. The Coordinators agents must be tested with numerous input scenarios and the tests must be repeatable and the results quantifiable. We at Global InfoTek, Incorporated are designing, building and testing the ACTIVE testbed to support three agent-based projects, each of which is specifying, designing, implementing and testing their own Coordinators agents. The performer teams are from Honeywell, Information Sciences Institute at USC, and SRI International.

The ACTIVE Testbed is designed to support both testing and evaluating the results of the tests. A Scenario Generator has been developed to generate scenarios based on input parameters. Developing the scenario manually would not be feasible given the scale of the testing. The Centralized Scheduler provides a baseline set of solutions for each input scenario that will be used to test the Coordinators agents. Chronos provides time management for the Centralized Scheduler.

The ambitious goals of the Coordinators Program, the size of the agent communities being fielded by the three teams, and the complexity of the problem space in which the agents operate, requires that the simulation testbed be both scalable and robust. Although we are aware of other testbeds for agent-based systems,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'06, May 8–12, 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005…$5.00.

we feel that our approach is better in that it is modular, service-oriented, scalable, and able to operate "24/7" so as to support the vast number to tests needed to evaluate the performance of each performer team's solutions to the agent-based team scheduling and coordination problems.

1.1 Related Work

Several agent-based testbeds have been proposed and built. Examples are the ART Testbed (Agent Reputation and Trust) at the University of Texas at Austin [2], JADE [1] at Telecom Italia in Naples, the Autonomous Multi-Agent Testbed at the University of Washington [5], and the AgentCities Testbed [7].

The ART Testbed [2] has an international team of researchers whose focus is to coordinate game design, game specification, testbed development and competition administration. The competition testbed is designed to fulfill two roles: 1) provide a forum in which competitors can compare results using objective metrics, and 2) provide a suite of tools with flexible parameters to allow researchers to perform customizable and easily-repeatable experiments. The architecture consists of multiple agents connected to a simulation engine which accesses a database, a game setup interface and game monitor interface. The focus is to measure agent trust and reputation in the game-playing environment. This is different from the ACTIVE Testbed environment.

The Java Agent DEvelopment Framework (JADE) [1] has a large following in the FIPA agent community. It is designed to enable agent-based system development through middleware services. The framework provides for scalable messaging among agents.

The Autonomous Multiagent Testbed [5] at the University of Washington supports seven miniature robots, a global vision system and a custom-designed infrared communication structure. The wireless aspect of the testbed gives the agents real autonomy.

The AgentCities testbed [7] supports FIPA agents and is intended to handle a very general service-oriented architecture for the dynamic search, discovery and composition of Semantic Web Services. This testbed also runs in 24/7 mode. As we shall explain in this paper, the ACTIVE Testbed under development at Global InfoTek to support the DARPA Coordinators Program has unique features not found in other testbeds.

1.2 Organization of Paper

This paper is organized as follows. Section 2 provides an overview of the ACTIVE testbed architecture and the layered services approach. Also discussed in Section 2 are the subsystems for the testbed, infrastructure and benchmarking. Section 3 gives a brief description of the Testbed Lab. Section 4 outlines our future work for the ACTIVE system.

2. ACTIVE TESTBED ARCHITECTURE AND SYSTEM

Global InfoTek (GITI) is a company dedicated to providing intelligent systems and advanced middleware solutions to government and industry. It has a long history of successful projects with DARPA and other government agencies. The Coordinators program requires an industrial-strength testbed in which to load, initiate, run, monitor, and evaluate the agent-based schedules, execution traces, and overall quality of the performer team solutions.

2.1 The ACTIVE Testbed Architecture

The ACTIVE testbed is a robust, scalable and lightweight experimental framework to support scientific evaluation of Coordinators agents. It is a system-agnostic framework, which supports any Coordinators agent interface, and requires minimal effort to integrate with the testbed. It also provides agent-based tools to orchestrate Coordinators agents and supporting testbed services, rapidly set up experiments, and quickly reinitialize the baseline system to support repeatable experiments.

Figure 2 provides an overview of the ACTIVE services for testing and evaluation for the Coordinators Program. The Testbed consists of three service layers: Test Harness, Command and Control (C&C), and Evaluation/Experimentation utilities. The CoABS Grid [6] middleware is used for agent communication. The Command and Control module manages the complex, distributed environment for the Coordinators agents and controls and monitors experiments. Evaluation/Utility Services include a Scenario Generator and a Centralized Scheduler as well as collecting and archiving test outputs for analysis and evaluation.

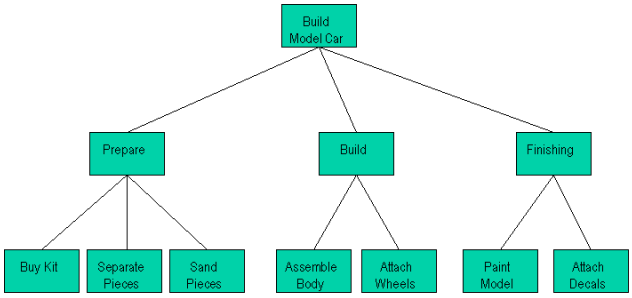


Figure 1. Hierarchical Task Structure

2.2 Test Harness

Figure 3 shows the ACTIVE layered services architecture, and Figure 4 shows the calling relationships among the services. Figure 5 presents an event-sequence diagram showing how the ACTIVE services interact in support of a test. The Test Harness layer encapsulates the agent tasking language, CTAEMS (Coordinators Task Analysis, Environment Modeling and Simulation) and the simulation engine, GMASS (GITI Multi Agent System Simulator). It also provides the interface for Coordinators agents to interact with the testbed services and with other Coordinators agents.

TAEMS (Task Analysis, Environment Modeling and Simulation) is a language designed to model tasks for software agents. TAEMS models the quantitative behavior of tasks, task activities and interactions between tasks [4]. CTAEMS was created by

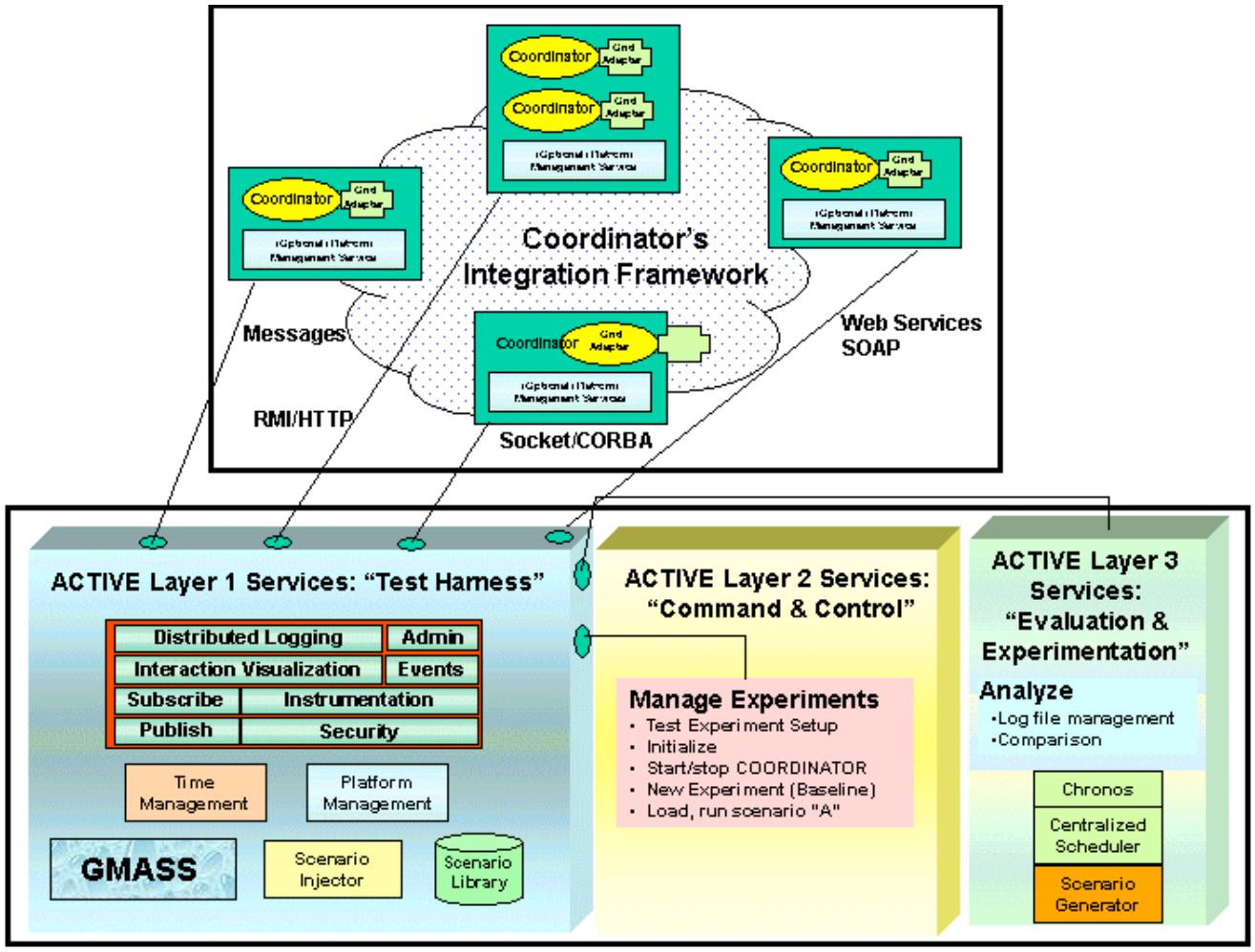


Figure 2. ACTIVE Services for Testing and Evaluation

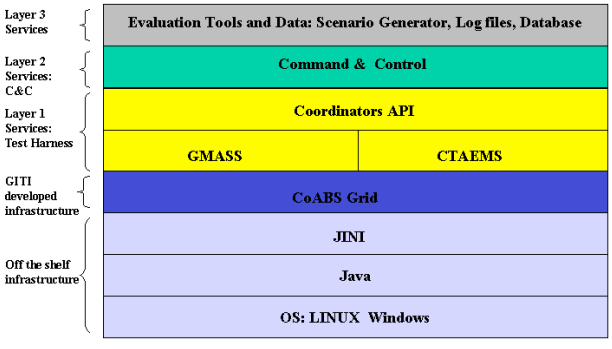


Figure 3. ACTIVE Layered Services Architecture

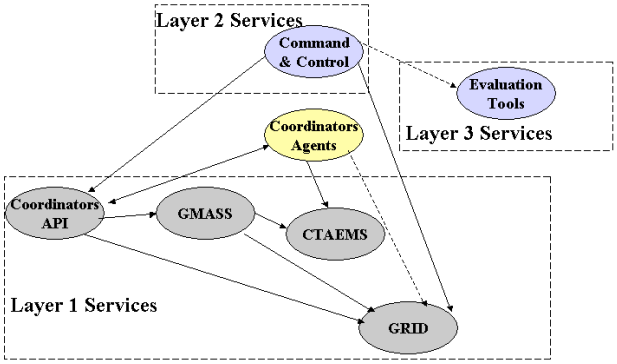


Figure 4. Relationships Among Services

modifying TAEMS to meet the unique needs of the Coordinators Program. The primary modifications are:

- Removal of GUI infrastructure, preprocessor, and commitments
- Changes to core data structures, virtual nodes and parser
- Addition of exogenous events

There are three types of CTAEMS Nodes: method, task, and task group. Methods are the leaf nodes of the graph and represent primitive executable actions. These actions are grouped together as tasks. A task can contain one or more methods and represents a goal that can be accomplished by its method(s). A task group represents a higher-level goal that is accomplished by the tasks in its structure. Figure 1 shows a simplified example of a task group for building a model car.

GMASS (GITI Multi Agent System Simulator) provides the simulation engine for testing coordination and negotiation among teams of Coordinators agents. GMASS was created by modifying MASS (Multi Agent System Simulator) to meet the unique needs of the Coordinators project. The primary modifications are:

- New Time Management model
- Rewrite of core event handling
- Communication changed from sockets to CoABS Grid

The simulator itself runs as a single process. All Coordinators agents involved in the simulation are connected to the simulator through the CoABS Grid [6]. This allows the simulation to monitor all Coordinators agent communication with the simulator and with other Coordinators agents. Each Coordinators agent has its own subjective view of the world and its own goals. The simulator has the objective view of the world containing information that the Coordinators agents do not see. These views are created by the Scenario Generator based on input parameters.

GMASS simulates the Coordinators agent's method execution. The actions of the simulator are directed by a queue containing a time-ordered list of events. The simulator receives two basic types of messages from agents: Method Abort and Method Start. Each message causes a method to be added to or deleted from the event queue. At each pulse the simulator selects the correct events and realizes their effects.

A CTAEMS Scenario for the Coordinators program contains a hierarchy of tasks assigned to each Coordinators agent along with deadlines and constraints. There are two types of views of a CTAEMS scenario, objective and subjective. The objective view contains the entire correct scenario. The subjective view does not include everything that is contained in the objective view. Agents only have access to a subjective view for their planning and interacting. The subjective view is different from the objective view because agents may have an imprecise model of what will actually happen, performance-wise when the method is executed. The objective view is created by the Scenario Generator. The objective views of the methods are managed by the simulator; the subjective views are created by the simulator and passed to the agents.

When an agent requests that the simulator start a method, the simulator retrieves the method from the objective CTAEMS structure. The first step of the simulator is to calculate the values of the quality, cost and duration that will be produced by this execution. A random number generator is used to calculate these values. Any event that occurs before a queued event is performed can affect the results of that event's execution

The simulator controls time simulation for the agents. A single unit of time is called a *pulse*. The simulator periodically sends a pulse to each of the agents. MASS manages time so that a pulse is complete after all agents have completed an action. The pulse time is not proportional to clock time or CPU time, but can be of varied length. A pulse ends when all agents have reported in. The MASS time management scheme has been altered for GMASS so that the pulse time is based on clock time and is the same length throughout a run. The length of a pulse is controlled by a setting in a configuration file.

GMASS time management provides a guarantee window that ensures that requests sent to the simulator within a time window in a given pulse will take effect within that pulse. Coordinators agents can run at any time they please and exchange messages within or outside the guarantee window. The simulator will process such requests within the given pulse period provided it can do so without delaying the next pulse. It will send a failure message to the requesting agent if it cannot do so.

The Coordinators API provides a single programming interface for accessing the simulator, representing the agent subjective view, communicating among agents, and communicating between agents and the simulator. The API is both a convenience for developers of Coordinators agents and a tool for observing and instrumenting the Coordinators agents. The Coordinators API serves as a wrapper for GMASS and CTAEMS.

2.2.1 Agent Startup

All Coordinators agents are required to implement the Coordinator interface, which allows them to receive messages from GMASS and from other agents. It also provides the capability to be started by the ACTIVE testbed.

The methods used during startup are initialize and begin. Initialize tells the agent to start any processes and perform all initialization that it can without having scenario-specific information. Begin notifies the agent that the simulation is starting. It passes the CTAEMS task structure and other information necessary for the agent to begin the simulation.

2.2.2 Agent to GMASS Communication

Agents send requests to GMASS using method calls defined in the Simulation interface. When an agent is started by the Testbed, it is given an object that implements this interface. The Simulation interface allows agents to start methods and abort methods in GMASS.

2.2.3 GMASS to Agent Communication

The Coordinator interface allows agents to receive messages from GMASS (with receiveMassMessage) and from other agents (with receiveMessage). The API defines different classes to represent different types of messages and events that agents may receive from GMASS. Agents are not permitted to communicate with one another until their begin methods have been called.

2.2.4 Agent Shutdown

When a simulation ends, agents will receive a disconnect message from the simulator. Upon receipt, the agent should shut down any external processes, close open files, etc. Once the agent is ready to exit, it should notify the Simulator by calling Simulation.bye. If

another simulation is to be run, the command and control infrastructure will initialize and restart a new instance of the agent.

2.3 Command and Control

The primary requirement for the C&C system is to provide completely automated unattended testing on a large scale of up to thousands of tests. In addition, tests are to be run simultaneously on multiple subnets.

C&C provides services to manage the large-scale experiments using the Test Harness and collecting data for evaluation. Features include automation and remote control, installation and configuration, test definition and management, preflight check, control of test results, instrumentation, logging and data capture. Services include experiment setup, initialization, load and run a scenario.

C&C monitors the full execution of a trial, which is one execution of a test or group of tests. A test refers to a test specification that can be repeated many times and run for different teams. A trial refers to a specific test run at one particular time for one particular team.

The test lab configuration consists of a simulation host that runs the simulation engine, a C&C host that runs the C&C software and logging database, and one or more client workstations that run the Coordinators agents.

2.3.1 Preparation for Testing

There are several steps which must take place prior to testing:

- Create the .xml file with input parameters for generating the CTAEMS scenario file.
- Run the Scenario Generator to create the CTAEMS file containing the objective view.
- Create an initial schedule for the scenario by running the Centralized Scheduler on the objective view.
- Run a utility to extract the simulator parameters and other info from the Scenario Generator input and output files.

2.3.2 Configuration and Initialization

In order to minimize time and effort required to prepare test hosts, C&C requires very little software on each test system. Only Java, Ant, and Bash Shell are required on each machine. All other software is pushed out by C&C just prior to running each trial. This software is prepared once for each type of system and is called a "mold." A mold consists of the baseline home directory and all files required for a Coordinators agent or the simulator to run. This includes software, data, configuration files, scripts, and utilities. For each trial, the software is essentially reinstalled. This ensures that the software is always up to date and that each test is run from a consistent clean initial state. When the software to be tested is modified or updated, it only needs to be delivered and installed once - to the C&C host. Installation on all other systems will be done automatically at the time of testing by the C&C services.

The techniques used by C&C ensure that test results are reliable and not invalidated by files or processes left behind by previous tests or by failure to keep software up to date and consistent between systems. C&C also supports distribution of other software packages to all machines in the lab, or specific subsets of machines. This facilitates the process of updating machine software, and will be used to ensure software consistency between all machines in the lab.

When C&C is invoked, it loads the Test Definition and other configuration information. For each trial, C&C verifies that all required hosts are available and running. Each trial will require one simulation host and one client workstation for each Coordinators agent to be run. C&C then baselines each host, by clearing the home directory and copying over a "mold" for each type of system. .

2.3.3 Monitoring Network Status

C&C monitors the network and all nodes to verify that all agents are running, GMASS is properly registered with the CoABS Grid, and that the test has started. If a required service is not running, C&C will attempt to restart it.

2.3.4 System Preflight Check

C&C's System Preflight Check utility verifies that the test environment is operating properly prior to execution of a test or experiment. It checks network connectivity and verifies that all required processes are running on the systems involved in the test. It checks for sufficient available disk space and system memory to execute the test and store logs.

2.3.5 Execution of a Test

The test definition files consists of a CTAEMS scenario file and a properties file. The properties file contains simulator parameters, the names of other configuration files, and the name of the CTAEMS file, generated by the scenario generator, contains the scenario (both the objective and subjective views). These files will be queued for processing by placing them in a specified directory. The CTAEMS file will be passed to the simulator by C&C and the simulator will provide the Coordinators agents with the subjective views only.

C&C will begin each trial by starting the CoABS Grid and then starting the simulator.

The C&C will use the Test Harness API to communicate with the Agent Conductor to initialize and start up all the s agents.

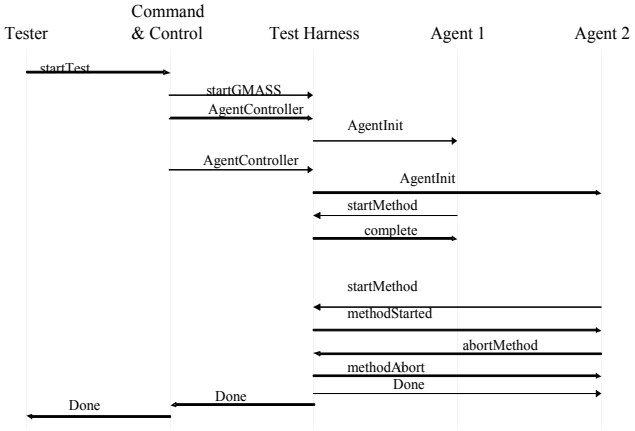


Figure 5. Single Test Event Sequence Diagram

2.3.6 Data Collection

Data is collected into logs during execution of each test. In order to minimize network traffic, logs are written to the local systems during the execution of the tests. After the test run is completed,

the logs are collected, compressed, and moved to the C&C host. The original CTAEMS scenario files, property files and other configuration files are stored with the log files for use in evaluation. In the event of a failed test, the logs are still moved to the C&C host for analysis..

2.4 Evaluation

The ACTIVE Evaluation layer includes tools used to support evaluation of performance of Coordinators agents.

2.4.1 Scenario Generator

The Scenario Generator automatically generates scenarios based on the input parameters. The scenarios are expressed as CTAEMS structures. A scenario consists of 1 to 10 concurrent problems linked by a top-level SUM task. The input parameters determine the number of problems in the scenario and specifications for window or deadline tightness and overlap, number of agents, nonlocal events, and uncertainty. Input parameters can be specified exactly, or can be randomly generated within a specified range. From the input parameters, the Scenario Generator produces a template file that includes all parameters and their default values and a CTAEMS text file that describes the scenario generated according to the input parameters.

2.4.2 Centralized Scheduler

The Centralized Scheduler is a program that solves a centralized coordination problem. It computes a set of solutions via an exhaustive search (or heuristic search for large problems) for experimentation and evaluation. The Scheduler operates on the same scenarios as the Coordinators agents. However, the Scheduler does not operate under the same time constraints as the Coordinators agents – it has unlimited time to create solutions. The results from the Centralized Scheduler will be used as a baseline for evaluating Coordinators agents.

2.4.3 Chronos

Chronos provides the time management for the Centralized Scheduler. The Centralized Scheduler can be used to determine whether a feasible initial schedule exists for a static scenario. However, in general, scenarios are dynamic – during the course of a scenario execution, several types of "unexpected" events may occur, such as method failures, new task arrivals, and deadline changes. Chronos integrates the centralized scheduler and GMASS, resulting in a centralized agent capable of handling these exogenous events. As with the distributed Coordinators agents, Chronos cannot see the future – dynamics are not known ahead of time, and past decisions (actions or non-actions) cannot be reversed. However, unlike the distributed agent teams, Chronos is not subject to time constraints between clicks, and Chronos is given the objective (rather than subjective) scenario view.

2.5 Infrastructure

2.5.1 CoABS GRID

The CoABS Grid [6] is middleware that integrates heterogeneous agent-based systems, object-based applications, and legacy systems. It includes a method-based application programming interface to register services, advertise their capabilities, discover services based on their capabilities, and send messages among those services. The Grid also provides a logging service, to log message traffic and other information, a security service to provide authentication, encryption, and secure communication; and event notification when agents register, deregister, or change their advertised attributes.

The Grid is built using Jini™ Network Technology developed by Sun Microsystems. Jini is a specification for service discovery and provides the robust and dynamic nature of the CoABS Grid. The Grid provides helper utility classes that are local to an agent and that shield the Grid user from the complexity of Jini. It also provides Jini services to facilitate agent communication and logging. Although on the one hand, the Grid shields the user from Jini, on the other, it exposes the underlying Jini classes, so that the programmer can use Jini directly, if desired.

2.5.2 Labmin

Labmin, the GITI Lab Manager, is a set of tools for managing a dynamic test lab environment. Labmin was developed with open source software and utilities. The primary utilities are SSH, partimage, Python, GRUB bootloader, and the PXE framework.

Labmin provides for remote reboot, restart, and shutdown. Systems can be turned on using the PXE framework. Running machines can be rebooted or shut down using secure channel commands. Remote change of operating system is also supported by this utility. Individual partitions can be re-imaged either from a network store, or a file on the local drive. This feature can be used for a return to baseline to ensure identical initial states for test runs.

For system integrity, Labmin commands are authorized only after they have been authenticated using public key cryptography. All Labmin communication is encrypted using asymmetric key algorithms such as RSA and DSA.

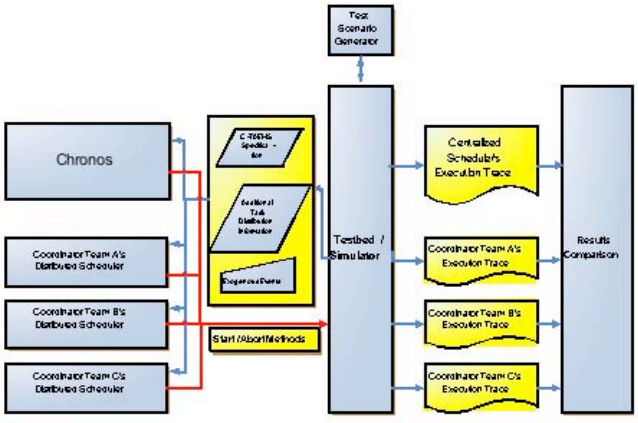


Figure 6. Centralized Scheduler in Context

2.6 Benchmarking of Coordinator Teams

Benchmarking of the Coordinators teams is done with the help of Chronos. The conceptual benchmarking scheme is depicted in Figure 2. Multiple scenarios are prepared for evaluation by the evaluation team. Each scenario contains (1) CTAEMS specification of the requested distributed tasks/methods, (2) information on specific probabilistic outcomes of methods in terms of duration and quality, and (3) exogenous events. The input (1) is given as input in the beginning of each test to both the Coordinators teams and Chronos. The inputs (2) and (3) are released to the Coordinators teams later, in real time, during the testing process. Chronos utilizes the centralized scheduler for

scheduling, which runs off-line. However, similar to the Coordinators teams, Chronos is not allowed to use inputs (2) and (3) to schedule tasks/methods that occur before the information (2) and (3) becomes available on the simulated timeline. The Coordinators teams and Chronos submit start/abort method instructions to the Testbed simulator. The simulator executes the instructions which results in producing an execution trail for each of the Coordinators teams and Chronos. Each of the execution trails are assigned the total Quality measurement according to the CTAEMS specification. The Quality measurement of the Chronos execution trail serves as the benchmark for the Coordinators teams. Chronos has the advantage of knowing the *objective view* in the CTAEMS specification, whereas the Coordinator teams know, at least initially, only the *subjective views*, i.e., parts of the CTAEMS specifications relevant to individual agents, and their environment. Also, the Coordinators teams need to produce decisions on starting/aborting tasks/method within strict time-limits, dictated by real-time constraints (enforced by the simulator), whereas Chronos has more time to make its scheduling decisions. However, due the volume of computations necessary in the testing process, the Centralized Scheduler is using a simple combinatorial algorithm, that sacrifices optimality for computational time. Thus, the quality of execution trails of Chronos may either outperform or under-perform the Coordinators teams' execution trails. In either case, the quality of execution trails produced by Chronos is used as a benchmark.

3. TESTBED LAB

The current Testbed Lab is configured with a separate subnet for each Coordinators Performer team, each of which contains 11 workstations. One of the machines on each subnet is a reserved host used for simulation execution and contains a 200 GB disk that will be used as a software and data repository. The 10 remaining machines are reserved for Coordinators agent execution. The current scenarios do contain not more than 10 agents, each agent on a Coordinators team can execute on its own machine. Figure 7 shows a diagram of the Lab Design. These machine clusters give high system/agent efficiency, enabling all three teams to execute scenarios independently in parallel, while the subnet topology ensures inter-team security. The ACTIVE Testbed software can support additional subnets, which may be required in later phases of the Coordinators program.

A server outside the individual subnets will serve as the C&C host to run the C&C services and host archived log data for evaluation. The C&C host is used to monitor the progress and operation of each cluster, manage the clusters, and manage large-scale experimentation and evaluation.

4. FUTURE WORK

The ACTIVE testbed will be advanced to support the continuing development of Coordinators agents. CTAEMs will be extended to include more non-local effects such as Disables and Hinders. The capability may be added to modify the task structure by adding exogenous events, removing nodes from the task structure, and reparenting nodes in the task structure. Currently the communications model for the agents is completely reliable; it may be changed in future to be unreliable in order to more realistically model the environment. GMASS may be modified to support a different timing model. All software components will be enhanced as needed to support testing on a larger scale.

5. ACKNOWLEDGMENTS

This work is supported by the Defense Research Projects Agency in the Coordinators Program.

We also thank the performer teams for their helpful comments and suggestions.

6. REFERENCES

[1] Cortese, E., Quarta, F. and Vitaglione, G. Scalability and Performance of JADE Message Transport System, Telecom Italia LAB, Naples, Italy, 2005.

[2] Fullam, K.T., Klos, T.B., Muller, G., Sabater, J., Schlosser, A., Topol, Z., Barber, K.S., Rosenschein, J.S., Vercouter, L. and Voss, M., A Specification of the Agent Reputation and Trust (ART) Testbed: Experimentation and Competition for Trust in Agent Societies. in *Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005)*, (Utrecht, 2005), 512-518.

[3] Horling, B., Lesser, V., Vincent, R., Wagner, T., Raja, A., Zhang, S., Decker, K. and Garvey, A. The TAEMS White Paper, http://mas.cs.umass.edu/research/taems/white/, UMASS, 2005.

[4] Horling, B. and Vincent, R. TAEMS/MASS Technical Overview, http://mas.cs.umass.edu/research/taems/Coordinators05-Taems-MASS.ppt, UMASS, 2005.

[5] Hoyt, S., McKennoch, S. and Bushnell, L.G. An Autonomous Multi-Agent Testbed using Infrared Wireless Communication and Localization, University of Washington, Seattle, WA, 2005, 6.

[6] Kahn, M.L. and Della Torre Cicalese, C., CoABS Grid Scalability Experiments. in *2nd Int;l Workshop on Infrastructure for Agents, MAS and Scalable MAS, Autonomous Agents 2001*, (Montreal, Canada, 2001).

[7] Willmott, S.N., Thompson, S., Constantinescu, I., Bonnefoy, D., Charlton, P., Zhang, T. and Dale, J., Agent Based Dynamic Service Synthesis in Large-Scale Open Environments: A Report from the Trenches. in *AAMAS'04*, (New York, NY, 2004).

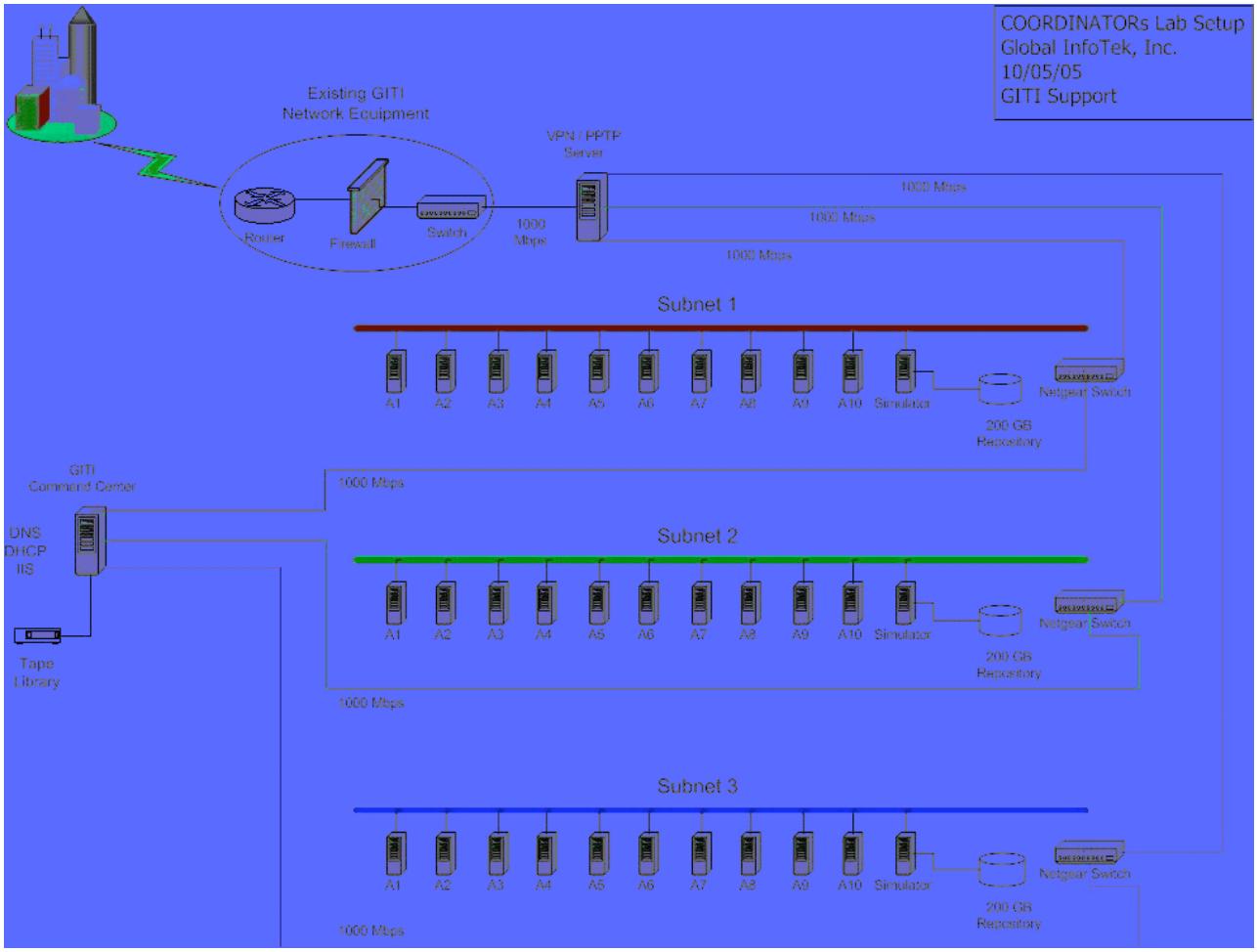


Figure 7. Testbed Lab Setup

Author Index

A

Abdallah, Sherief 850
Afsharchi, Mohsen 923
Agha, Gul 1297
Agno, Andrew 1449
Agogino, Adrian 58, 1079
Agotnes, Thomas 153, 177, 232
Aihara, Kazuyuki 789
Airiau, Stéphane 609, 1263
Akashi, Osamu 601
Albright, Cory 1467
Alchieri, João 97
Aldea, Arantza 114
Aldewereld, Huib 223
Alechina, Natasha 217, 1288
Alvarado, Matias 114
Álvares, Luís Otávio 807
Amamiya, Makoto 1323
Amato, Christopher 341
Amgoud, Leila 347, 414
Amours, Sophie D 132
Ando, Yasushi 73
Asada, Minoru 3
Ashlagi, Itai 819
Au, Tsz-Chiu 561

B

Bada, Gustavo Aranda 1282
Balan, Gabriel 616, 804
Balasubramanian, Vidhya 1469
Balbo, Sandrine 939
Balch, Tucker 795
Bañares-Alcántara, Rene 114
Banerjee, Bikramjit 798
Banerjee, Dipyaman 1249
Barber, K Suzanne 1241, 1317
Basso, Eduardo 810
Baumer, Eric 969
Bazzan, Ana 108, 126, 1471
Becker, Marcel 1441
Becker, Raphen 529
Beetz, Michael 42, 482
Belecheanu, Roxana A. 1549
Benedicenti, Luigi 1430
Benfield, Steve S. 10
Benford, Steve 996
Benisch, Michael 1405
Berbers, Yolande 1421
Bernstein, Daniel S 341
Berry, Pauline M. 1467, 1564
Bertoli, Piergiorgio 217
Beun, Robbert-Jan 899
Bezek, Andraz 457
Bhat, Sooraj 338
Birukou, Aliaksandr 265
Bisson, Robert 993
Blanzieri, Enrico 265
Blikstein, Paulo 1457
Boella, Guido 229, 688
Boer, Frank S de 1271, 1279
Boes, Andrew 1445
Bohte, Sander 728
Bölöni, Ladislau 411
Bonarini, Andrea 783
Bordini, Rafael 1288, 1291
Bosse, Tibor 117, 1317
Botelho, Luis 65, 813
Botti, Vicent 1107, 1282
Bouchard, Bruno 320
Boucké, Nelis 842
Bousquet, Francois 81
Bouzouane, Abdenour 320
Bowling, Michael 713
Bowring, Emma 1413, 1424, 1467
Bratko, Ivan 457
Braun, Peter 585
Brodsky, A. 1580
Brueckner, Sven 111, 993
Brussel, Hendrik Van 1451
Brzostowski, Jakub 355
Buskirk, Chris van 1441
Byde, Andrew 1199

C

Cai, Kai 1143
Campos, André 97
Candale, Teddy 1211, 1266
Canuto, Anne 97
Cavallo, Ruggiero 882
Cerquides, Jesús 893
Chaib-draa, Brahim 132, 736, 1369, 1372
Chakraborty, Doran 691
Chalmers, Stuart 1459
Chang, Chee Fon 377
Chavez, Christina 1332
Chechetka, Anton 1427
Chen, Huajun 1257
Chen, Wei 662
Cheng, J. 1580
Chevaleyre, Yann 362, 866
Child, Christopher 792
Chopra, Amit 1326, 1345
Choren, Ricardo 1332
Cleaver, Timothy 1055
Clement, Bradley J. 1483
Cohen, Dave 676
Combettes, Stéphanie 1384
Conde, Toni 89

Conitzer, Vincent 521, 537
Conley, Ken 1467, 1564
Cooper, Brian 338
Cornwell, D. 1580

D

Damiano, Rossana 688
Das, Rajarshi 1572
Dasgupta, Prithviraj 479, 609
Dash, Rajdeep K. 1208, 1353, 1463
Dastani, Mehdi 1110, 1271, 1279, 1285
David, Esther 834
Davin, John 1433
Day, Robert 1183
Debenham, John 1225
Decker, Keith 335, 662
Denzinger, Jörg 923
Desai, Nirmit 1326
Diggelen, Jurriaan van 899
Dignum, Frank 223, 899, 1110
Dignum, Virginia 1110
Dinont, Cédric 593
Diuk, Carlos 313
Dix, Jürgen 1009
Dodier, Robert 676
Dolgov, Dmitri 297
Doshi, Prashant 1025
Dou, Dejing 952
Drew, Touby 1534
Drogoul, Alexis 27
Dror, Eran 1175
Druon, Emmanuel 593
Duff, Simon 1033
Dugan, Christopher 1104
Dunin-Kęplicz, Barbara 1017
Durfee, Edmund 297, 874, 1058
Dutta, Partha 826

E

Eijk, Rogier van 899
Emami, G. 1580
Endriss, Ulle 362, 392, 866
Erdoğdu, Utku 329
Erete, Ikpeme 1266
Espinosa, Agustin 1107
Esteves, Sofia 65
Estivie, Sylvia 866

F

Faltings, Boi 1233, 1397
Fan, Xiaocong 34
Far, Behrouz H 408, 923
Fatima, Shaheen 1205
Fehler, Manuel 120, 1439
Feldhousen, M. 1580
Felner, Ariel 553
Ferdous, Raihana 1055
Fernández, Fernando 720
Fiddian, Nick J. 1459
Fielding, Daniel 996
Fitzpatrick, Stephen 1441
Fix, Julia 105
Flores, Roberto 1369
Fok, Chien-Liang 323
Fries, Benedikt 915
Fuentes Fernandez, Rubén 1303
Fukazawa, Yoshiaki 73
Fukuda, Kensuke 601, 682
Fukumoto, Taro 1461
Fullam, Karen 1241

G

Gal, Margo 1506
Gal, Ya'akov 370
Galanti, Daniel 10
Gams, Matjaz 457
Garcia, Alessandro 1332
Garcia, Ana 1107, 1282
García-Camino, Andrés.....223, 670
Gasser, Les 1378, 1381
Gati, Gergely 1441
Gedikli, Suat 42
Gemelli, Nathaniel 1445
Gerbrandy, Jelle 185, 193
Gerding, Enrico 1208
Germain, Bart Saint 1451
Gervasio, Melinda 1564
Ghidini, Chiara 217
Ghose, Aditya 377
Ghose, Debasish 471
Gilboa, Arnon 553
Gilpin, Andrew 545, 1127, 1453
Gini, Maria 1534
Giorgini, Paolo 265
Giovannucci, Andrea 893
Giroux, Sylvain 320
Glinton, Robin 1443
Gmytrasiewicz, Piotr 1025
Godo, Lluis 400
Goh, Angela 1260
Gómez-Sanz, Jorge 1303
Gonçalves, Bruno 65
Gordon, Geoffrey J 249
Górski, Janusz 1063
Goschnick, Steve 939
Governatori, Guido 513, 1311
Grando, Adela 1375
Gratch, Jonathan 985
Gray, Peter M. D. 1459
Gray, W. Alex 1459
Greenstadt, Rachel 1424
Gregori, Miguel Escrivá 1282
Grossi, Davide 1110
Guillory, Andrew 795

Gunupudi, Vandana 1116
Guo, An Yuan 326
Gupta, Nachi 58
Gutiérrez, Edgar 1542
Guyot, Paul 27

H

Hadeli, Karuna 1451
Halpern, Joseph 744
Hameurlain, Nabil 414
Hanachi, Chihab 1384
Hanak, David 1441
Harland, James 235, 1033
Harvey, Peter 377
He, Minghua 1159
Hendrickson, Jim 10
Hernández, Héctor 1542
Herrler, R. 1439
Herzig, Andreas 209
Himoff, Jonathan 1455, 1514
Hirayama, Katsutoshi 890
Hirotsu, Toshio 601
Hoek, Wiebe van der 153, 177, 185, 201
Hoen, Pieter Jan t 728
Holvoet, Tom 842, 1052, 1314, 1421
Honiden, Shinichi 27, 73, 949
Horling, Bryan 1098
Houk, Jim 1378
Hübner, Jomi Fred 1288, 1291
Hulstijn, Joris 688
Hung, Eugene 1087
Huynh, Trung Dong 1217

I

Isbell, Charles 338, 775, 795
Ishida, Toru 81, 123
Ishikawa, Fuyuki 949
Ishizuka, Mitsuru 955
Ito, Takayuki 435, 1151, 1183
Iwasaki, Atsushi 1135
Iwasaki, Hirotoshi 73

J

Jago, Mark..... 217, 1288
James, Geoff..... 676
Jamroga, Wojciech 232
Jayatilleke, Gaya..... 1465
Jennings, Nicholas R
…400, 426, 826, 834, 1159, 1208, 1217, 1353, 1459, 1463
Jiang, Hong.....1522, 1556
Jin, Jing.....1441
Johansson, Stefan..... 569
Johnston, Mark D. 1483
Jorritsma, Annemiek.....1506
Julian, Vicente.....1282
Jung, Yuna.....1329
Junges, Robert.....126, 1471
Jurca, Radu.....1233

K

Kakas, Antonis..... 384
Kamboj, Sachin..... 335
Kant, Jean-Daniel.....50
Karsai, Gabor.....1441
Karunatillake, Nishan.....426
Katz, Ron.....697
Kehagias, Dionisis.....1191
Kempen, Masja.....1320
Kephart, Jeffrey O.....1572
Kerschberg, L.....1580
Khalid, Mahboob Alam..... 915
Khan, Majid Ali.....411
Khan, Shakil..... 1337
Khandaker, Nobel.....1556
Kim, Minkoo.....1329
Kirchlechner, Bernhard..... 42
Klein, Mark.....435
Klugl, Franziska.....120, 1439
Klusch, Matthias.....915
Koenig, Sven.....281
Koes, Mary.....491
Kogo, Akihiro.....1323
Konikowska, Beata.....169
Kowalczyk, Ryszard.....355, 585
Kramer, James.....488
Kraus, Sarit.....257, 273, 697, 834, 1095
Kungas, Peep.....432
Kurihara, Satoshi.....601, 682
Kwon, YoungMin.....1297

L

Lakkaraju, Kiran.....1381
Lam, Dung..... 1317
Lapouchnian, Alexei..... 241
Larson, Kate.....1167
Lawton, James..... 1445
Lazaric, Alessandro..... 783
Lee, Jungtae..... 1329
Lesperance, Yves.....241, 1337
Lesser, Victor.....305, 326, 529, 622, 850, 1098
Lester, James.....961, 977
Leszczyna, Rafał 1063
Leung, Ho-fung.....946
Lewis, Michael.....1071, 1113
Likhachev, Maxim.....281
Lister, Kendall.....943
Littman, Michael.....313
Liu, Jiming.....646
Liu, Xuliu.....1556
Lockemann, Peter C.....1101
Logan, Brian.....217, 996, 1288
Lomuscio, Alessio.....145, 161, 201, 548
Long, C.....1580

Lopes, António 65
Lopes, Luis.....1294
López-Carmona, Miguel.....429
Lu, Chenyang.....323
Luck, Michael.....1459, 1549
Luke, Sean.....616, 760, 801, 804
Luo, Xiangyu.....220
Luo, Xudong.....1159
Lutz, Carsten.....137

M

Machado, Jaçanã 813
Maheswaran, Rajiv T.....577, 1441
Makino, Takaki.....789
Maldonado, Alexis.....42
Malhotra, V.....1580
Mallya, Ashok.....1361
Mamei, Marco.....485
Mancoridis, Spiros.....1475
Manisterski, Efrat.....257, 834
Mao, Wenji.....985
Mao, Yuxin.....1257
Marcinkiewicz, Marek.....705
Martinez, Kirk.....1353
Martínez-Miranda, Juan 114
Martinoli, Alcherio.....441
Massaguer, Daniel.....323, 1469
Masutani, Osamu.....73
Mateas, Michael.....775
Mathieu, Philippe.....593
Matskin, Mihhail.....432
Matsunaga, Kumiko.....1461
Matsuo, Tokuro.....1183
Matsutani, Toshihiro.....1135
Matthews, Robert..... 993
Maudet, Nicolas.....362, 866
Mayk, Israel.....1475
McBurney, Peter.....705, 1549
McDermott, Drew.....952
McNeese, Michael.....34
McQuiggan, Scott.....961
Mechitov, Kirill.....1297
Mehrotra, Sharad.....1469
Meisels, Amnon.....553
Mertens, Koenraad.....1421
Meshulam, Ram.....1095
Meyer, Andre.....1320
Meyer, John-Jules.....899, 1271, 1279, 1285
Meyer, Thomas.....226
Miao, Chunyan.....1260
Miguel Alberola, Juan.....1107
Miller, Tim.....1549
Mine, Tsunenori.....1323
Mitkas, Pericles.....1191
Miyashita, Hiromitsu.....1461
Modi, Pragnesh Jay.....1104, 1433, 1475
Moldt, Daniel.....105
Monderer, Dov.....819
Mongan, William M.....1475
Moraitis, Pavlos.....384
Moreau, Luc.....826
Morley, David.....858
Mott, Bradford.....977
Moyaux, Thierry.....132
Mukai, Takanori.....1461
Mulet, Luis.....1107
Munoz de Cote, Enrique.....783
Munroe, Steve.....1549
Myers, Karen.....858

N

Nair, Ranjit.....289
Nakasone, Arturo.....955
Nau, Dana.....244, 561
Neches, Bob.....1441
Negri, Alessandro.....907
Nelson, Mark.....775
Nguyen, Anh.....786
Nguyen, Hai.....795
Nimis, Jens.....1101
Nitta, Katsumi.....1461
Nitz, Kenneth.....1467
Niu, Jinzhong.....1143
Noriega, Pablo.....223, 1542
Norman, Timothy J.....1459
Nourbakhsh, Illah.....491
Novák, Peter 1009

O

Okamoto, Steven 630
Oliveira, Denise de.....1471
Ordóñez, Fernando.....273
Oren, Nir.....1459
Orgun, Mehmet.....220
Osman, Nardine.....238
Owens, Sean.....1443

P

Padgham, Lin 1001, 1049, 1465
Padhy, Paritosh.....1353
Padmanabhan, Vineet..... 513
Palanca, Javier..... 1282
Palmer, Doug.....676
Panait, Liviu.....760, 801, 804
Paquet, Sebastien.....736
Paranjape, Raman.....1430
Parkes, David C.....1151, 1397
Parsons, Simon.....129, 705, 1041, 1143
Paruchuri, Praveen..... 273
Parunak, Van.....111, 993
Pasquale, Joseph..... 1087
Pasquier, Philippe.....1369, 1372
Patel, Jigar.....1459

Patil, Pratik.....1490
Paulino, Hervé.....1294
Pavlíček, Dušan.....1447, 1498
Pavón, Juan 1303
Pearce, Jonathan.....577, 1424, 1467
Pěchouček, Michal 420, 1447, 1498
Peintner, Bart.....1467, 1564
Penczek, Wojciech.....169
Peng, Jing.....798
Perotto, Filipo.....807, 810
Perrussel, Laurent.....226
Petcu, Adrian..... 1397
Peysakhov, Maxim..... 1104
Pfeffer, Avi..... 370
Pham, Duc Quang..... 235
Phelps, Steve..... 705
Pighin, Fred..... 1490
Pini, Maria Silvia..... 685
Platt, Glenn..... 676
Poggi, Agostino..... 907
Polat, Faruk..... 329, 332
Poutré, Han La 728, 1506
Powers, Rob..... 752
Preece, Alun..... 1459
Prendinger, Helmut..... 955
Procaccia, Ariel..... 497, 505, 673
Pugh, Jim..... 441
Puppe, Frank.....120
Putten, Sander van der.....1506

R

Rabinovich, Zinovi 465, 679
Rahwan, Iyad.....347, 426
Raimondi, Franco.....161, 548
Ramchurn, Sarvapali.....400, 426
Ramírez, Victor Soto.....1542
Rand, William.....1457
Rathnasabapathy, Bharanee.....1025
Reches, Shulamit..... 1095
Reece, S..... 1463
Regli, William..... 1104, 1475
Rêgo, Leandro 744
Rehák, Martin 420, 1447
Restelli, Marcello..... 783
Richards, Debbie..... 767
Riemsdijk, M Birna van..... 1271, 1279, 1285
Roberts, David..... 338, 775
Roberts, S..... 1463
Robertson, David..... 238
Robles, Armando..... 1542
Robles, Marco Julio..... 1542
Robu, Valentin..... 1506
Rodríguez-Aguilar, Juan Antonio.....223, 670, 893
Rogers, Alex..... 1159, 1208, 1463
Rogers, Craig M..... 1441
Roman, Gruia-Catalin..... 323
Rosenschein, Jeffrey S.... 338, 465, 497, 505, 673, 679, 1202
Rossi, Francesca..... 685
Rotolo, Antonino..... 513
Royakkers, Lambèr 1110
Rzevski, George..... 1514

S

Saadati, Shahin 1467
Sadeh, Norman 1405
Saha, Sabyasachi 423
Sanchez, Romeo 1441
Sandholm, Tuomas 521, 537, 545, 1127, 1453
Santos, Emanuel 97
Sardina, Sebastian 1001
Sarne, David 257
Sato, Shinya 601, 682
Sattar, Abdul 220, 1055
Sauro, Luigi 185
Sauter, John 993
Sawamura, Hajime 1461
Scerri, Paul 630, 1071, 1113, 1443
Schelfthout, Kurt 1314
Schermerhorn, Paul 654
Scheutz, Matthias 488, 654
Scheve, Christian von 105
Schlegel, Tino 585
Schurr, Nathan 1490
Sen, Sandip 423, 609, 691, 1211, 1249, 1263, 1266
Şensoy, Murat 931
Serafini, Luciano 217
Shabtay, Amit 679
Shadbolt, Nigel R 1217
Shao, Jianhua 1459
Shehory, Onn 1175, 1530
Shen, Jiaying 529, 622
Shercliff, Gareth 1459
Shintani, Toramatsu 1183
Shoham, Yoav 752
Sibertin-Blanc, Christophe 1384
Sierra, Carles 223, 400, 670, 1225
Silaghi, Marius 1389
Silva, Bruno Castro da 810, 1471
Silva, Lavindra de 1001
Silva, Roberto da 108
Simari, Gerardo 244, 1041
Singh, Munindar 1326, 1345, 1361
Sinha, Arpita 471
Sislak, David 1447, 1498
Sklar, Elizabeth 129, 767, 1143
Skobelev, Petr 1514
Sliva, Amy 244
Smit, Annika 1320
Soares, Rodrigo 97
Soh, Leen-Kiat 1522, 1556
Somefun, Koye 417
Song, Insu 1311
Spaan, Matthijs 249
Starnes, I. 1580

Stathis, Kostas 792
Sterling, Leon 939, 943
Stockreisser, Patrick J. 1459
Strehl, Alexander 313
Stulp, Freek 482
Sturtevant, Nathan 713
Su, Kaile 220
Subrahmanian, Venkatramanan 244
Such, Jose M. 1107
Sugawara, Toshiharu 601, 682
Sujit, P B.....471
Sukthankar, Gita 638
Sullivan, Keith 801, 804
Sun, Bingjun 34
Sun, Christine 939
Sun, Shuang 34
Sundresh, Sameer 1297
Swarup, Samarth 1381
Sycara, Katia 491, 630, 638, 915, 1071, 1113, 1427, 1443
Szekely, Pedro ….....1441

T

Taillibert, Patrick 593
Tambe, Milind 273, 289, 577, 1413, 1424, 1467, 1490
Tang, Yuqing 129
Tate, Stephen 1116
Taveter, Kuldar 943
Teacy, W.T. Luke 1459
Tennenholtz, Moshe 819
Terrasa, Andres 1107
Thalmann, Daniel 89
Thangarajah, John 1033, 1049, 1465
Thévenin, Jean-Marc 226
Thiriot, Samuel 50
Thompson, Simon 1459
Tomaiuolo, Michele 907
Tomlinson, Bill ….....969
Torii, Daisuke 81
Torre, Leendert van der 229, 688
Toulis, Panos 1191
Toyama, Marcelo 108
Tožička, Jan …….....420
Treur, Jan 117
Troquard, Nicolas 209
Tumer, Kagan 58, 1079, 1119
Turci, Paola 907

U

Uller, Miroslav 1498
Undeger, Cagatay 332
Uribe, Tomás 1467, 1564

V

Valckenaers, Paul 1052, 1451
Varakantham, Pradeep 289
Vasconcelos, Wamberto 670
Velasco, Juan R 429
Zohar, Aviv 1202
Veloso, Manuela 449, 720
Venable, Kristen Brent 685
Venkatasubramanian, Nalini 323, 1469
Verbrugge, Rineke 1017
Verstraete, Paul 1451
Vincent, Regis 1449
Vlassis, Nikos 249
Vu, Thuc 752

W

Wakaki, Toshiko 1461
Walsh, Toby 685
Walton, Christopher 238, 1375
Wang, Jun 1378
Wang, Kewen 1055
Wanyama, Tom .....,.....408
Weng, Jianshu 1260
Weyns, Danny 842, 1314
Whalley, Ian N. 1572
Wijngaards, Niek 1320
Wilensky, Uri 1457
Winikoff, Michael 1465
Wobcke, Wayne 786
Wooldridge, Michael 153, 177, 185, 201, 1291
Woźna, Bożena 145
Wright, Robert 1445
Wu, Jianhui 874, 1058
Wu, Zhaohui 1257

X

Xiang, Yang 19
Xie, Xiao-Feng 646
Xu, Yang 1071, 1113

Y

Yagdar, Osher 1449
Yamins, Daniel 1122
Yang, Gu 449
Yang, Yan 1430
Yarden, Aner 1095
Yau, Man Lok 969
Yen, John 34
Yeung, Ching-Man Au 946
Yokoo, Makoto 289, 1135, 1389, 1413
Yolum, Pinar 931
Yorke-Smith, Neil 858, 1467, 1564
Yoshioka, Nobukazu 949
Yu, Bin 1071, 1443

Z

Zambonelli, Franco 485
Zhang, Haizheng 305
Zhang, Kun 19
Zhang, X. 1580
Zheng, Xiaoqing 1257
Zilberstein, Shlomo 341